Uma Arquitetura Escalável de Roteamento com Requisitos de Qualidade de Serviço

por

Luís Henrique Maciel Kosmalski Costa

UFRJ

Tese submetida para a obtenção do título de

Mestre em Ciências em Engenharia Elétrica

ao Programa de Pós-Graduação de Engenharia Elétrica

da COPPE/UFRJ

Uma Arquitetura Escalável de Roteamento com Requisitos de Qualidade de Serviço

Luís Henrique Maciel Kosmalski Costa

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

Aprovada por:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Prof. José Ferreira de Rezende, Dr.

Prof. José Marcos Silva Nogueira, Dr.

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 1998

COSTA, LUÍS HENRIQUE MACIEL KOSMALSKI

Uma Arquitetura Escalável de Roteamento com Requisitos de Qualidade de Serviço [Rio de Janeiro] 1998

XIV, 84 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 1998)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Roteamento com Requisitos de Qualidade de Serviço
2. Aplicações Multimídias
3. Algoritmos para Descoberta de Rotas

I. COPPE/UFRJ II. Título (Série)

Aos meus pais.

Agradecimentos

À minha família.

Ao meu orientador, Professor Otto Carlos Muniz Bandeira Duarte.

Ao membro externo da banca examinadora José Marcos Silva Nogueira.

Ao Professor e amigo José Ferreira de Rezende.

Aos amigos do GTA: Flavio, Renata, Baiano, Emerson, Belem, Hermes, Rubi, Antonio, Andrea, Adriana, Daniel, Fred, Bill, Alexandre, Marcos Tadeu, Ronaldo, Vidal, Sílvia, Marcial, Jonny, Paulo André, Arthur, Michel, Celio, Marcio, Spíndola, Gilberto, Iniciações Científica (Bernardo, Tadeu, Doc, Pedro, FredII), Nacao, Sérgio, Rodolfo, Jucele, Ezequiel, Duboc, Profs. Aloysio, Leão e Lanza.

Ao pessoal de potência: Evandro, Tony, Marcelo, Pedro, Luís Oscar.

Aos amigos da graduação: Cagy, Velho, Rosas, Wallace, Natalizi, Quadrados e Guilherme.

Ao pesquisador Kannan Varadhan, pela ajuda fundamental com o simulador.

Ao Professor Richard, pelo apoio nos congressos e no laboratório.

Aos funcionários: Wilson, Wanderlei, Fidel, Paulo Roberto, Bia, Solange, Patrícia.

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

Uma Arquitetura Escalável de Roteamento com Requisitos de Qualidade de Serviço

Luís Henrique Maciel Kosmalski Costa

Julho de 1998

Orientador: Otto Carlos Muniz Bandeira Duarte
Programa: Engenharia Elétrica

O roteamento baseado em requisitos de qualidade de serviço (QoS) é um componente fundamental da nova arquitetura de serviços integrados para a Internet. Um dos aspectos principais da implementação do roteamento com QoS é a garantia de sua escalabilidade.

Ao contrário dos mecanismos tradicionais, o roteamento com QoS deve levar em consideração várias métricas para cada enlace. A construção de algoritmos com diversas métricas e a mesma complexidade dos tradicionais é a principal dificuldade do roteamento com QoS. Os algoritmos podem ser definidos de duas formas diferentes. Numa das abordagens, leva-se em conta a dependência existente entre as métricas utilizadas. A outra abordagem consiste em tratar as restrições independentemente. Alguns autores consideram inviável, dada sua complexidade, a utilização de algoritmos considerando mais que duas restrições independentes.

Neste trabalho, um novo algoritmo de roteamento que considera três métricas é proposto, utilizando a abordagem que considera as métricas independentes aliada a uma métrica única combinada. Este algoritmo possui a vantagem de ser construído de forma matematicamente simples. Além disso, serve de base para um esquema de roteamento escalável, pois possui a mesma complexidade que os algoritmos tradicionais. A arquitetura proposta foi implementada dentro de um simulador de redes e comparada com um algoritmo comum para três métricas. O desempenho obtido foi melhor em média 25% que o algoritmo comum.

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

A Scalable QoS-based Routing Architecture

Luís Henrique Maciel Kosmalski Costa

July 1998

Supervisor: Otto Carlos Muniz Bandeira Duarte
Department: Electrical Engineering

Quality of Service (QoS) based routing is a main component of the new Integrated Services in the Internet Architecture. Guaranteeing scalability is one of the main issues in implementing a QoS-based routing scheme.

Unlike traditional mechanisms, QoS-based routing must consider multiple metrics for each link. Constructing algorithms with several metrics and the same complexity as classical algorithms is the main difficulty on QoS-based routing. These can be implemented in two ways. In the first approach, the metric dependency is taken into account. The other solution is to consider the metrics independently. Some authors consider the use of algorithms that take into account more than two metrics impossible because of its computational complexity.

In this work, a new routing algorithm that considers three metrics is proposed, using the approach that considers the metrics independently combined to a single mixed metric approach. This algorithm has the same complexity as traditional algorithms, being solvable in polinomial time. The proposed architecture was implemented inside a network simulator and compared to a simple approach that considers three metrics separately. The performance obtained was 25% better than the non-scalable three metrics solution.

Palavras-chave

Redes de Alta Velocidade

Garantia de Qualidade de Serviço

Roteamento com Requisitos de Qualidade de Serviço

Aplicações Multimídias

Algoritmos para Descoberta de Rotas

Glossário

QoS :	Qualidade de Serviço (*Quality of Service*);
IETF :	*Internet Engineering Task Force*;
IP :	*Internet Protocol*;
CAC :	Controle de Admissão de Conexão (*Connection Admission Control*);
ATM :	Modo de Transferência Assíncrono (*Asynchronous Transfer Mode*);
CBR :	Taxa Constante de Bits (*Constant Bit Rate*);
RSVP :	*Resource Reservation Protocol*;
NS-2 :	*Network Simulator version 2*;
OSPF :	*Open Shortest Path First*;
RIP :	*Route Information Protocol*;
DVRP :	*Distance-Vector Routing Protocol*;
NHRP :	*Next Hop Routing Protocol*;
IntServ :	*Integrated Services*;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo I

Introdução

A INTERNET vem sendo usada cada vez mais no transporte de informação multimídia, tal como transmissão de imagens, voz e vídeo. Com o aumento da capacidade de processamento dos computadores pessoais e o surgimento de novas aplicações multimídias, a Internet tende a receber uma quantidade crescente deste tipo de tráfego. Estes tipos de mídias possuem características de tráfego distintas do tráfego de dados para o qual a rede vinha sendo utilizada até então. Desta forma, o tráfego gerado pelas aplicações multimídias possui alguns requisitos de Qualidade de Serviço (QoS) que as aplicações anteriores, como a transferência de arquivos, não possuíam [1]. Por exemplo, a transmissão de vídeo em tempo real exige uma grande quantidade de banda passante enquanto a transmissão de áudio interativo exige um atraso bem limitado. No entanto, estes requisitos de QoS não são tão importantes para uma transferência de arquivos, onde por outro lado a perda de pacotes é um fator crítico, pois o arquivo recebido é inútil se incompleto ou com erros.

Este tipo de utilização, no entanto, não é o uso para o qual o protocolo IP (*Internet Protocol*) foi projetado, há mais de vinte anos. A arquitetura de rede da Internet foi projetada para o envio da informação usando a disciplina de serviço de melhor esforço, ou seja, sem quaisquer garantias de vazão, atraso ou outro requisito de QoS. A rede se compromete a enviar a informação sem qualquer garantia, de acordo com os recursos disponíveis no momento. O controle da informação que foi recebida e a eventual retransmissão ficam a cargo da camada superior, ou da aplicação.

Para o suporte de tráfego de tempo real na Internet, algumas novas características e serviços devem ser adicionados. Novas classes de serviço precisam ser oferecidas, além do envio de melhor esforço, para prover alguma garantia de atraso e vazão às aplicações. É necessário um mecanismo que permita às aplicações sinalizar seus requisitos de QoS à rede, e novos mecanismos de gerenciamento de tráfego nas estações e nos roteadores.

O IETF (*Internet Engineering Task Force*) possui diversos Grupos de Trabalho relacionados com a definição de uma nova arquitetura para a Internet. O Grupo IntServ (*Integrated Services*) é responsável pela definição de uma arquitetura de Serviços Integrados para a Internet, que inclui a proposta de novas classes de serviço a serem providas [2, 3]. Existem outros grupos relacionados com a definição de diferentes elementos desta nova arquitetura. Existe um grupo relacionado com o projeto de um protocolo através do qual podem ser realizadas reservas de recursos, o RSVP (*Resource ReSerVation Protocol*) [4, 5, 6]. O RSVP deverá ser utilizado pelas aplicações para solicitar à rede a QoS necessária de acordo com suas caracterítsticas de tráfego. O protocolo é responsável por criar um estado "soft" descrevendo os recursos alocados nos roteadores por onde determinado fluxo estará trafegando. Existe também um Grupo de Trabalho, o ISSLL (*Integrated Services over Specific Link Layers*) relacionado com a garantia de requisitos de QoS sobre diferentes camadas de enlace.

Outra área que merece atenção, e para a qual foi recentemente criado um Grupo de Trabalho específico, consiste na definição de uma arquitetura de roteamento que considere os requisitos de QoS dos fluxos transportados na descoberta das rotas. No esquema de roteamento clássico, dados uma origem e um destino, todos os pacotes transmitidos seguem a mesma rota, a menos que ocorram mudanças nos nós da rede que forcem a uma alteração das tabelas de rotas, ou nos casos em que o protocolo implementa o balanceamento de carga. No roteamento baseado em requisitos de QoS, a rota para determinado fluxo, dadas sua origem e destino, não é sempre a mesma. Esta depende também das características do tráfego gerado por cada fluxo.

Sendo assim, uma conexão relativa a uma teleconferência, por exemplo, pode utilizar um caminho diferente na rede do utilizado por uma transmissão de arquivos. Na figura I.1, possivelmente a transferência de arquivos seria realizada através da rota 1, e a teleconferência através da rota 2. Isto considerando-se que a teleconferência possui requisitos de atraso e banda passante muito mais restritivos que uma transferência de arquivos. Se um protocolo tradicional como o RIP (*Route Information Protocol*)[7] ou o DVRP (*Distance-Vector Routing Protocol*)[8]. estivessem sendo utilizados, os pacotes seguiriam sempre a rota 1 pois é a que possui o menor número de saltos.

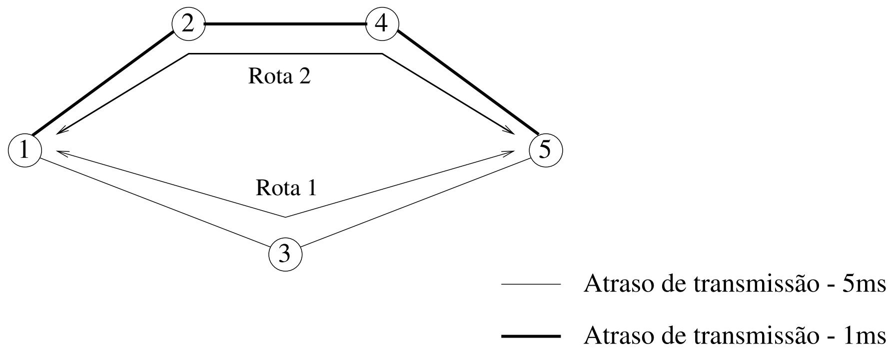


Figura I.1: Rotas com diferentes características de QoS.

Nos esquemas de roteamento tradicionais, as rotas são decididas de acordo com uma métrica associada a cada enlace da rede. Esta métrica pode ser alterada, de acordo com a política administrativa de cada organização. Isto ocorre por exemplo com o protocolo OSPF (*Open Shortest Path First*)[9]. No entanto, no momento do cálculo de rotas, o algoritmo leva em conta apenas uma restrição para cada arco no grafo representativo da rede. Já no roteamento baseado em requisitos de QoS, os algoritmos devem considerar múltiplas métricas para cada arco no grafo [10]. Existirão múltiplas métricas para cada enlace, por exemplo uma representativa do atraso de transmissão, uma para a banda passante disponível, outra para a variação de atraso (*jitter*) esperada, etc.. Estas métricas devem ser consideradas simultaneamente. Além disso, as tabelas de roteamento deverão ser construídas de forma a considerar os requisitos de QoS dos fluxos transmitidos. Uma primeira alternativa poderia ser a manutenção de tabelas com as rotas utilizadas por cada fluxo, o que no entanto exigiria tabelas de roteamento bem maiores que as atuais.

O mais importante é que o esquema adotado deve possuir uma complexidade que não o torne inviável com o crescimento da quantidade de nós da rede. A esta característica dá-se o nome de escalabilidade do algoritmo de roteamento. Esta característica toma importância cada vez maior dado o ritmo de crescimento atual da Internet. Dentro da garantia de escalabilidade do algoritmo de roteamento, existem dois pontos que se destacam: a complexidade do algoritmo implementado para a descoberta de rotas e o tamanho das tabelas de roteamento utilizadas.

De uma forma geral, os algoritmos para a descoberta de rotas dadas várias métricas podem ser definidos de duas formas diferentes. Numa das abordagens, leva-se em conta a dependência existente entre as métricas utilizadas [11]. Por exemplo, o atraso medido fim-a-fim estará relacionado com a banda passante disponível na rota utilizada, e também com a probabilidade de perda de pacotes, considerando-se um mecanismo com retransmissão de pacotes. Esta solução leva a uma maior complexidade na definição das relações entre as métricas utilizadas, embora exija poucas modificações nos algoritmos de roteamento tradicionais.

A outra abordagem consiste em tratar independentemente as métricas. Neste caso, é imprescindível garantir que as métricas utilizadas sejam independentes entre si. Esta solução transfere a complexidade do problema para a construção de algoritmos eficientes, de maneira que o tempo gasto no cálculo das rotas não inviabilize o mecanismo [12]. O número de passos gastos pelo algoritmo iterativo na descoberta de rotas quando são consideradas várias métricas não deve ser maior, de preferência, que o gasto pelos algoritmos tradicionais que consideram apenas uma métrica. Alguns autores consideram a utilização de algoritmos considerando mais que duas restrições independentes inviável, dada sua complexidade [12, 11]. No caso de uma caracterização de tráfego através de duas restrições, e não mais, algoritmos simples podem ser construídos, como mostrado em [12]. Além disso, uma das métricas neste caso deve ser a representativa da banda passante.

Na abordagem em que as métricas são consideradas independentes, existem duas formas de calcular as rotas. A primeira, considerando as métricas separadamente, maneira para a qual não se conhece um algoritmo que resolva o problema em tempo polinomial, para três ou mais métricas. Uma outra alternativa seria definir uma métrica única combinada. Esta consiste em uma função matemática que é gerada a partir das diferentes métricas utilizadas. O algoritmo de descoberta de rotas consideraria, desta forma, não as múltiplas métricas, mas apenas a métrica única combinada. Neste caso, poderiam ser utilizados os algoritmos de roteamento tradicionais com pequenas modificações, sem aumentar sua complexidade computacional, ou seja, o número de passos necessário para a descoberta de rotas.

Neste trabalho, a abordagem que considera as métricas de forma independente é utilizada. A solução apresentada em [12] para o cálculo de rotas para duas restrições, sendo uma delas a banda passante, é o ponto de partida para a construção de algoritmos que consideram três métricas. Esta solução é utilizada em conjunto com a adoção de uma métrica única combinada de outras duas métricas. Algumas modificações são realizadas nos algoritmos, de maneira que os diferentes requisitos de QoS estejam garantidos. Desta forma, um novo algoritmo de roteamento é proposto. Este algoritmo considera três métricas, diferentemente dos propostos até então, com a vantagem de ser construído de forma matematicamente simples. Além disso, serve de base a um esquema de roteamento escalável, pois possui a mesma complexidade que os algoritmos tradicionais, sendo solúvel em tempo polinomial. Isto porque é construído baseado nestes algoritmos, sem qualquer alteração que leve a um aumento do número de iterações necessário para a descoberta das rotas. A utilização de três métricas para cada rota garante a caracterização de um maior número de aplicações que com apenas duas métricas e, conseqüentemente, uma melhor garantia de QoS.

Para garantir a escalabilidade adota-se a estratégia de definir rotas para classes de serviço ao invés de fluxos individuais. Sendo assim, dadas as características de tráfego de cada fluxo, este deve ser mapeado na classe de serviço que melhor o representar. Desta forma, são construídas tabelas de roteamento pequenas que permitem uma consulta rápida. Esta foi a solução utilizada na arquitetura proposta.

O mecanismo proposto foi implementado utilizando um simulador de redes, o NS-2 (*Network Simulator version 2*), construído pela Universidade de Berkeley. Este simulador é um programa de domínio público, que está em constante atualização, e para o qual são feitas contribuições de pesquisadores de centros de pesquisa do mundo inteiro. Através da utilização do simulador, foi realizada uma comparação do algoritmo proposto com outro algoritmo no qual as três métricas são consideradas separadamente, o que levaria *a priori* a um número de iterações maior. Além disso, dentro do simulador, foram feitas todas as modificações que serão necessárias na implementação de uma arquitetura real de roteamento com requisitos de QoS. Ou seja, não só o algoritmo foi implementado, mas também foram realizadas as alterações necessárias na tabela de roteamento e no classificador de pacotes IP. O desempenho da arquitetura proposta pôde então ser avaliado. Os resultados obtidos mostraram uma vantagem de cerca de 25%, em média, em relação ao algoritmo de controle utilizado.

No capítulo II deste trabalho, são analisados os principais requisitos que o roteamento com QoS deve respeitar, além do relacionamento deste com o protocolo de reserva de recursos, o RSVP. No capítulo III são apresentadas as principais características relacionadas com o algoritmo de roteamento, considerando os requisitos de QoS mais comuns. No capítulo IV são feitas algumas considerações que permitem a construção de algoritmos com três requisitos de QoS, apresentados no capítulo V, onde é descrita a implementação realizada e os resultados obtidos. O capítulo VI apresenta algumas considerações finais e propostas para trabalhos futuros.

Capítulo II

Arquitetura de Serviços Integrados

PARA a provisão de garantias de QoS para as novas aplicações multimídias na Internet, é necessário o desenvolvimento de uma nova arquitetura de rede. Dentro desta arquitetura, diversos elementos devem ser definidos, sendo um dos mais importantes o roteamento baseado nos requisitos de QoS dos fluxos transmitidos. Neste capítulo, é feita uma breve apresentação da arquitetura para Serviços Integrados na Internet proposta pelo IETF [2]. Em seguida são apresentados os principais pré-requisitos para o mecanismo de roteamento baseado em QoS e os problemas encontrados na relação entre este mecanismo e o protocolo de reserva de recursos, RSVP.

II.1 Serviços Integrados na Internet

A arquitetura de roteamento com requisitos de QoS proposta neste trabalho baseia-se no modelo de Serviços Integrados na Internet definida pelo Grupo IntServ do IETF [2]. A arquitetura propõe-se a suportar, além do serviço de melhor esforço, aplicações de tempo real e controle de compartilhamento dos enlaces de transmissão.

Desta forma, a arquitetura de Serviços Integrados define uma função nos roteadores responsável por fornecer diferentes requisitos de QoS para os diversos fluxos, denominada controle de tráfego. A funcionalidade deste controle é definida por seis componentes, como pode ser visto na figura II.1:

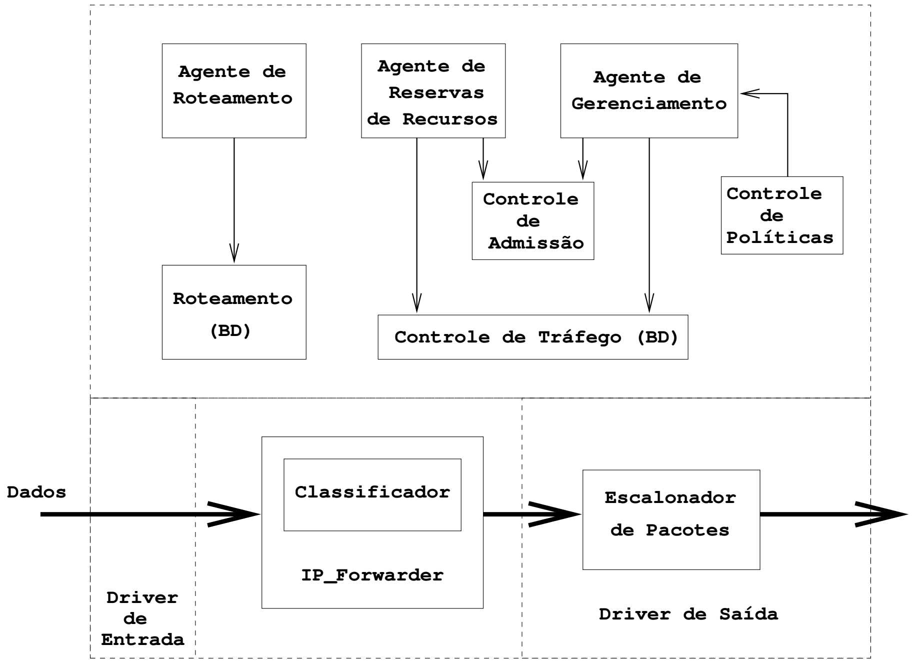


Figura II.1: Arquitetura de Serviços Integrados na Internet.

- **controle de admissão**: implementa o algoritmo de decisão utilizado pelo roteador para determinar se a qualidade de serviço requisitada por um novo fluxo pode ser provida;

- **controle de políticas**: mecanismo responsável por tomar decisões de caráter administrativo na aceitação de fluxos pelos roteadores;

- **classificador**: para propósitos de controle de tráfego, cada pacote deve ser mapeado em uma classe de tráfego. Todos os pacotes de uma determinada classe devem receber o mesmo tratamento do escalonador de pacotes. O classificador está localizado dentro do módulo de transmissão IP (*IP_Forwarder*);

- **escalonador de pacotes**: responsável por gerenciar a transmissão de pacotes de diferentes fluxos utilizando um conjunto de filas, cada uma representativa de determinada classe de tráfego. Para cada interface de saída existirá um conjunto destas filas;

- **reserva de recursos**: processo necessário para criar e manter o estado específico de cada fluxo nas estações e nos roteadores ao longo do caminho de transmissão utilizado. Este componente é representado basicamente pelo protocolo de reserva de recursos (no caso, o RSVP[4]);

- **roteamento baseado em requisitos de QoS**: processo que deve interagir com o RSVP para determinar as rotas a serem seguidas na transmissão dos fluxos, de acordo com seus requisitos de QoS.

II.2 Requisitos do Protocolo de Roteamento

Considerando a arquitetura da Internet, a natureza do tráfego gerado, as classes de serviço que devem ser providas e os requisitos dos usuários, podem ser definidos diversos pré-requisitos para a construção de um protocolo de roteamento baseado em requisitos de QoS. A seguir são apresentados diversos requisitos, divididos nos grupos principais: métricas e rotas, robustez, eficiência, prioridades, inter-domínios, políticas e escalabilidade. Outros requisitos, como o suporte à mobilidade e segurança, podem ser definidos [13].

Quanto a métricas e rotas, é desejável que o protocolo de roteamento suporte múltiplas métricas para os caminhos (como por exemplo atraso, banda passante, variação de atraso etc.). O cálculo da rota deverá ser realizado de acordo com os requisitos de cada aplicação para cada uma destas características de QoS. Desta forma, diversas classes de serviço podem ser definidas, de acordo com as características de cada tipo de aplicação.

A definição das métricas que serão utilizadas para mapear os requisitos de QoS das aplicações deve ser realizada de acordo com as classes de serviço suportadas [14, 15]. O algoritmo de roteamento deve ser capaz de calcular dinamicamente rotas satisfazendo os requisitos de QoS. Diz-se, neste caso, que o algoritmo considera múltiplas restrições. Deve oferecer suporte de QoS para vários tipos de fluxos, incluindo ponto-a-ponto (*unicast*) e ponto-a-multiponto (*multicast*) e, além disso, deve suportar receptores heterogêneos em termos dos requisitos de QoS [16, 17].

O protocolo deve ser robusto, ou seja, possuir as habilidades de detectar e responder a mudanças nas capacidades de QoS dos enlaces e de continuar o roteamento mesmo diante de múltiplas falhas na rede. O tempo de adaptação às mudanças da rede deve ser o menor possível.

Em se tratando de eficiência, deseja-se que o protocolo seja capaz de rotear fluxos de curta duração com a menor sobrecarga de processamento possível, além de utilizar os recursos da rede de forma eficiente, através de balanceamento de carga, e controles de admissão de chamadas globais [18]. A sobrecarga de processamento devida ao roteamento deve ser mínima. Deve ser possível o controle dos recursos, ou seja, a capacidade de limitar o consumo de recursos da rede pelas diferentes classes de tráfego.

Deve existir um mecanismo para estabelecer prioridades de transmissão entre os fluxos e para permitir que tráfegos de alta prioridade sejam estabelecidos com precedência em relação aos menos prioritários. Um fluxo de alta prioridade deve ser aceito com uma maior probabilidade que um fluxo de baixa prioridade.

Os requisitos para o roteamento entre domínios diferentes constituem um aspecto importante que tem gerado bastante polêmica [19]. Mais especificamente, a natureza da informação trocada na fronteira entre dois domínios administrativos deve ser bem determinada. A capacidade de introdução de restrições de acordo com políticas administrativas no roteamento do fluxo de informação na vizinhança entre duas organizações é indispensável.

O algoritmo utilizado deve ser capaz de lidar com o aumento do número de nós e de enlaces da rede sem se tornar inviável em termos da capacidade de processamento das máquinas, assim como de sua capacidade de memória. Assim, o protocolo deve ser escalável, pré-requisito indispensável quando se considera a dimensão de uma rede como a Internet. Ou seja, as tabelas de roteamento mantidas não podem crescer descontroladamente com o tamanho das redes. Nas seções seguintes, os aspectos mais importantes relacionados com a escalabilidade do protocolo de roteamento são explorados.

II.2.1 Determinação da qualidade de serviço e reserva de recursos

O Grupo de Trabalho em Serviços Integrados [2, 20, 21, 22] do IETF vem se concentrando no gerenciamento local de recursos nos roteadores. A noção de controle de admissão [23] foi introduzida, permitindo ao roteador a decisão de aceitação ou rejeição de um novo fluxo, baseado na disponibilidade local de recursos. O controle de admissão requer o conhecimento da QoS disponível em todos os enlaces conectados ao roteador. É uma questão em aberto, no entanto, como os valores de QoS são calculados para enlaces de diferentes tipos. Este problema está sendo discutido no domínio do Grupo de Trabalho ISSLL do IETF.

Este problema ocorre, por exemplo, no caso de um roteador conectado a uma rede não *broadcast* de múltiplo acesso, como a rede ATM (*Asynchronous Transfer Mode*)[24, 25, 26]. Nesta situação, o roteador deverá ter diversas escolhas para o próximo nó, com caminhos correspondentes através da rede ATM, cada um dos quais possivelmente com diferente disponibilidade de QoS. As questões que se impõem são como o roteador pode determinar quais são as opções de roteamento através da rede ATM, qual é a disponibilidade de recursos em cada um destes caminhos, e que valores de QoS devem ser informados pelo roteador à rede ATM, no caso do roteamento baseado em reserva de recursos ser implementado na Internet. Como uma ilustração destes problemas, os padrões atualmente definidos para o roteamento IP sobre ATM, como o NHRP (*Next Hop Routing Protocol*), permitem a seleção de um único ponto de saída da rede ATM para um datagrama IP. Com requisitos de QoS nos fluxos IP, pode não existir um caminho entre o roteador de entrada e o roteador de saída definido pelo NHRP que suporte a exigência de QoS.

II.2.2 Propagação da informação

A arquitetura de roteamento baseada no estado do enlace requer que os roteadores informem através de difusão o *status* dos recursos locais (como a faixa de transmissão disponível e os atrasos medidos) e as informações relacionadas com a topologia local para todos os outros roteadores da rede. A difusão da informação de *status* de um roteador para os outros tende a ser um processo dispendioso em termos de sobrecarga do sistema de comunicação e das capacidades de processamento dos roteadores. Até então, o roteamento intra-domínio tem sido baseado em métricas fixas para os enlaces. Neste contexto, a eficiência na propagação da informação não era um pré-requisito, pois haviam poucas modificações na topologia da rede provocando poucas trocas de mensagens pelo protocolo de roteamento. Por esse motivo, os algoritmos de roteamento como o implementado pelo OSPF (*Open Shortest Path First*) utilizam-se de inundação (cada nó transmite informação para todos os nós da sub-rede) com um período de *refresh* como forma de propagar as atualizações.

Para o roteamento baseado em QoS eficiente, no entanto, mecanismos de inundação não são adequados para a distribuição da informação. No caso do roteamento baseado em requisitos de QoS, a informação do *status* dos enlaces é muito volátil, ou seja, seus valores podem mudar significativamente em curtos intervalos de tempo com a admissão de fluxos de tempo real. A maneira como essa informação é quantificada de forma a diminuir a freqüência das atualizações é uma questão importante. Quanto menos precisa for a informação, menor a chance de um roteador fonte encontrar um caminho ideal. Por outro lado, os custos da manutenção de informação muito acurada em cada roteador podem ser muito altos. Técnicas alternativas para reduzir a sobrecarga da difusão, como o envio baseado em árvores e a utilização de redes ativas [27, 28, 29], foram propostas.

II.2.3 Controle de recursos

Dada a existência de múltiplas classes de serviço, é necessário planejar uma rede para suportar as demandas de tráfego previstas para cada classe. Neste sentido, um administrador de rede pode particionar logicamente os recursos dos roteadores e dos enlaces entre as diversas classes de serviço. Este particionamento rigoroso, porém, resultará em uso ineficiente dos recursos da rede, pois durante determinados períodos de tempo pode haver carga excessiva de tráfego de uma classe de serviço e ociosidade de recursos em outra classe. É desejável, portanto, permitir que recursos não utilizados sejam alocados a tráfegos que os necessitem. Este compartilhamento deve ser feito de forma controlada, de maneira a impedir que tráfego de uma determinada classe de serviço possa ter acesso a mais recursos do que fora projetada a rede para esta classe por períodos de tempo prolongados. Esta é uma tarefa do mecanismo de roteamento. Pode ser realizada através de preempção, de maneira não contraditória à atribuição de prioridades entre os fluxos ativos. O projeto de um esquema de compartilhamento de recursos apropriado e sua incorporação ao roteamento baseado em QoS são importantes. Em particular, as sobrecargas impostas pela implementação do particionamento lógico de recursos e compartilhamento dinâmico constituem uma questão a ser estudada.

II.2.4 Fluxos de curta duração

A princípio, o roteamento baseado em requisitos de QoS incorpora sobrecarga de processamento durante o estabelecimento do fluxo, se considerarmos que as rotas sejam calculadas para cada novo fluxo. É importante garantir que este tempo gasto a mais não seja desproporcional ao tempo de duração das sessões. O problema tende a se tornar mais grave quando é utilizada como enlace uma rede com circuitos virtuais, como a rede ATM, onde vários circuitos virtuais chaveados (*Switched Virtual Circuit* - SVC) de curta duração resultam em maiores sobrecargas devidas ao estabelecimento de conexão. Mesmo sem o roteamento baseado em QoS, as sobrecargas devidas ao estabelecimento de fluxo e à construção do *soft state* através do RSVP são adicionadas a cada sessão. A diminuição da sobrecarga devida ao roteamento no estabelecimento de fluxos deve ser buscada. Uma das formas possíveis pode ser a utilização de *caches* com as rotas recentemente computadas, de maneira que novos fluxos para os mesmos destinos não exijam novo cálculo dos caminhos. Outra forma de eliminar o problema é a definição de rotas para os diferentes tipos de tráfego, representados por diversas classes, por exemplo. De acordo com as características de cada fluxo, pode-se determinar a classe a que o fluxo pertence e a rota a ser utilizada.

No caso de uma rede não orientada a conexão, como a Internet, o cálculo de rotas pode ser feito para classes de tráfego diferentes, em vez de sob demanda para cada fluxo alocado. Isto diminui a sobrecarga de processamento e, muito importante, o tamanho das tabelas de roteamento a serem construídas e atualizadas.

II.2.5 Aceitação de fluxos ponto-a-ponto

O cálculo de um caminho possível para um fluxo pode ser feito de maneira simples: o roteador elimina primeiramente todos os nós e enlaces impraticáveis, ou seja, os que não podem dar suporte aos requisitos de QoS determinados, da sua representação local da topologia da rede e encontra um caminho fim-a-fim na rede remanescente. Porém este tipo de algoritmo resulta na aceitação de um número menor de fluxos do que seria possível de outra forma. Este tipo de admissão de fluxos corresponde ao trabalho realizado pelo Grupo de Trabalho em Serviços Integrados do IETF; neste caso, um fluxo passa pelo controle de admissão à medida em que é aceito por cada roteador ao longo do caminho. Enquanto este tipo de controle é necessário para administrar os recursos em um roteador, não leva em conta qualquer visão global do consumo dos recursos pelos diversos fluxos individuais.

A eficiência no roteamento poderia ser obtida através de uma camada de controle de admissão a mais. Este controle de admissão de mais alto nível pode considerar o consumo de recursos de cada fluxo em relação aos recursos disponíveis ao longo de um caminho de forma a determinar se a aceitação deste fluxo é desejável globalmente. Sendo assim, um fluxo individual pode ser rejeitado mesmo se existir um caminho possível para roteá-lo, se a aceitação deste fluxo puder resultar num declínio geral do número de fluxos carregados pela rede. A determinação desta camada de controle de admissão de mais alto nível, com justiça para todos os fluxos que desejem utilizar a rede, é uma área de pesquisa de interesse atual.

II.2.6 Roteamento dinâmico

A característica dinâmica do protocolo de roteamento merece atenção especial na definição de uma arquitetura baseada em requisitos de QoS. A complexidade do algoritmo para a descoberta de rotas será refletida no tempo gasto para que o protocolo de roteamento se adapte às mudanças ocorridas na rede.

Quando um enlace ou um nó na rede falham, o protocolo de roteamento é o responsável por modificar as tabelas de rotas nos nós para que os fluxos de dados possam continuar a serem transmitidos. Enquanto esta adaptação não ocorrer, pacotes serão perdidos, ou deixarão de ser enviados, uma vez que as tabelas de roteamento estarão inconsistentes. Deste ponto de vista, quanto mais simples for o algoritmo para a descoberta de rotas, mais eficiente será o mecanismo de roteamento na adaptação às mudanças da rede. O algoritmo utilizado deve recalcular as rotas com o menor número de iterações possível. No caso do roteamento considerar os requisitos de QoS das aplicações, a construção de algoritmos eficientes torna-se um problema. Isto porque o algoritmo deve considerar não uma mas várias métricas.

O cálculo de rotas sujeito a múltiplas restrições de QoS em um fluxo é um problema complexo. Para algumas combinações de requisitos de QoS, o problema é NP-completo, como mostrado em [12]. No entanto, alguns algoritmos foram propostos considerando dois requisitos de QoS, sendo um deles obrigatoriamente a faixa de transmissão, o outro podendo ser por exemplo o atraso de propogação. Estes algoritmos podem ser incorporados aos mecanismos de roteamento. No capítulo seguinte são apresentados três algoritmos baseados nesta proposta.

II.3 A Relação entre o RSVP e o Roteamento

Dentre as diretivas de projeto adotadas para o RSVP, as mais importantes são o suporte a receptores heterogêneos, que devem poder realizar reservas de recursos de acordo com suas necessidades e a possibilidade de estas reservas, baseadas nos requisitos do usuário final, serem agregadas ao longo da rede, de forma a obter uma melhor utilização de recursos [4].

Desta forma, são definidos diversos estilos de reserva, que regulam a maneira como as reservas são combinadas ao longo dos caminhos. O RSVP possui características particulares importantes no que diz respeito ao roteamento. O RSVP é um protocolo unidirecional, orientado ao receptor. Inicialmente, uma fonte transmite mensagens *path* através de uma árvore de distribuição *multicast*, para um grupo de receptores inicial. As reservas são realizadas posteriormente, pelos receptores interessados, através de mensagens *reserve* enviadas na direção da fonte através do caminho definido pelas mensagens *path*. Este mecanismo está representado na figura II.2. Nesta figura, a estação SH (*Source Host*) é a geradora do tráfego *multicast*, e RH (*Receiver Host*) é um dos receptores, que após a recepção da mensagem *path* envia uma mensagem *reserve* na direção da fonte. As reservas são mantidas através da definição de um *soft state*, sendo descartadas se não atualizadas periodicamente.

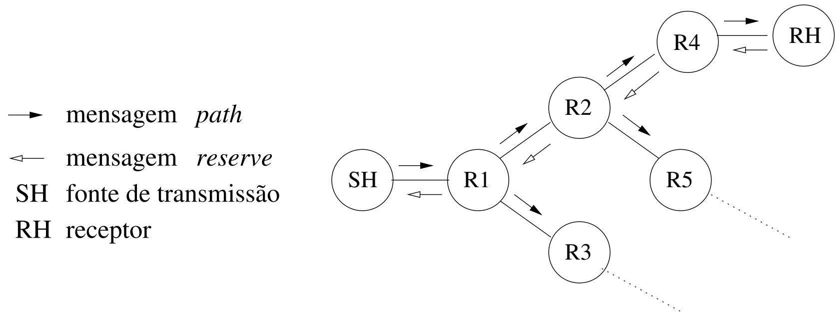


Figura II.2: Funcionamento básico do RSVP.

II.3.1 Fluxos ponto-a-ponto e ponto-a-multiponto

O modelo de roteamento baseado em requisitos de QoS possui algumas implicações na sinalização. A QoS desejada para um fluxo deve ser especificada antes que um caminho possa ser calculado. Em contrapartida, no RSVP, um caminho precisa existir antes que uma reserva possa ser realizada. O RSVP utiliza-se de mensagens *path* para notificar aos receptores de uma sessão as características do tráfego gerado pela fonte, e para estabelecer um estado para o caminho do fluxo através dos roteadores. Com o roteamento baseado em requisitos de QoS, a última função não é útil uma vez que o caminho do fluxo é calculado baseado apenas nas reservas dos receptores.

Utilizando o modelo atual do RSVP com o roteamento baseado em QoS, o estabelecimento de um fluxo ponto-a-ponto poderia ser realizado pela transmissão de uma mensagem *path* enviada da fonte ao receptor através de um canal de melhor esforço, por exemplo, e uma mensagem *reserve* enviada através de um caminho calculado por um roteador próximo ao receptor de forma a acomodar os requisitos de QoS do fluxo.

Com relação a fluxos ponto-a-multiponto, o caminho para este tipo de fluxo dentro das especificações atuais do RSVP é a árvore de distribuição do IP Multicast a partir da fonte. Para receber um determinado fluxo, uma estação deve primeiramente aderir ao respectivo grupo *multicast*, e por conseqüência entrar na árvore *multicast* de uma determinada fonte. Para requisitar uma certa QoS para o fluxo, cada receptor deve enviar uma mensagem *reserve* na direção da raiz da árvore, reservando recursos nos enlaces e roteadores através do caminho. O conjunto de receptores de um fluxo *multicast* é dinâmico, e transparente para a fonte de transmissão. O protocolo de roteamento ponto-a-multiponto dinamicamente mantém uma árvore de distribuição por fonte, na qual as folhas são o conjunto atual de receptores. Este protocolo estabelece a árvore ponto-a-multiponto independentemente dos requisitos de QoS dos fluxos subseqüentemente roteados através desta. Sendo assim, não existe garantia de que uma folha terá sucesso na reserva de recursos de acordo com os requisitos de QoS desejados.

Uma solução para o roteamento ponto-a-multiponto baseado em requisitos de QoS é separar o fluxo de mensagens *path* do RSVP do fluxo de dados real de cada fonte. Neste caso, um protocolo de roteamento ponto-a-multiponto básico proveria a árvore de distribuição para as mensagens *path* para todo receptor participante do grupo. A recepção de uma mensagem *path* por um roteador iria disparar um algoritmo para, incrementalmente, determinar uma nova árvore para o fluxo de dados, ou seja, através da adição e remoção de ramos da árvore existente.

Outra solução poderia ser determinar a árvore *multicast* baseado na informação da própria mensagem *path*. Isto significa que a árvore é criada baseada no conjunto de receptores conhecido inicialmente e assumindo a alocação máxima de recursos por todos os receptores. A heterogeneidade, por sua vez, pode ser conseguida através do ajuste das reservas de acordo com a informação contida nas mensagens *reserve*.

O dinamismo no grupo de receptores pode ser administrado, incrementalmente, recalculando as árvores toda vez que mensagens *path* de *refresh* forem geradas pela fonte. A desvantagem desta solução está no fato das árvores de distribuição *multicast* serem calculadas baseadas nos níveis de reserva de recursos máximos, tornando possível que árvores realizáveis não sejam computadas, quando seus níveis de reserva reais forem mais baixos que os níveis máximos usados na tentativa inicial.

Resumidamente, o modelo de sinalização definido pelo RSVP e o mecanismo de roteamento baseado em requisitos de QoS não se encaixam perfeitamente. A resolução destas incompatibilidades e o custo destas soluções são questões em aberto. Na seção seguinte é apresentada uma proposta recente para o gerenciamento dos caminhos considerando QoS com o protocolo RSVP.

II.3.2 Gerenciamento de caminhos com o RSVP

Rotas baseadas em requisitos de QoS podem ser realizadas utilizando versões estendidas de alguns algoritmos de roteamento, como os baseados no estado do enlace, implementado pelo OSPF [30]. No entanto, o uso de algoritmos de roteamento baseado em QoS possui algumas implicações a mais do que quando utilizam-se algoritmos convencionais.

Primeiramente, um mecanismo específico precisa ser usado para identificar fluxos com requisitos de QoS, de maneira que possam ser escalonados para o algoritmo de roteamento baseado em QoS correspondente. Nesta proposta, assume-se que o protocolo RSVP é o responsável por esta tarefa. Mais especificamente, as mensagens *path* do RSVP são utilizadas para disparar o roteamento baseado em QoS. Em segundo lugar, devido às variações na disponibilidade de recursos na rede, rotas entre os mesmos endereços de origem e destino, com os mesmos requisitos de QoS, podem variar, dependendo de quando o pedido de reserva é feito. No entanto, é preciso assegurar que tais mudanças não sejam sempre refletidas nos caminhos existentes, de forma a evitar oscilações nestes caminhos, limitando as mudanças a casos onde a configuração anterior torne-se inadequada.

Como resultado, alguma informação deve ser associada com uma rota baseada em QoS para determinar sua validade, ou seja, decidir se o algoritmo de roteamento deve ser acionado para determinar uma nova rota potencialmente melhor ou se a rota atual é adequada e deve ser mantida. Para tanto, pode-se utilizar uma convenção como definida a seguir. Diz-se que um caminho está "marcado" quando o seu estado significa que o roteamento baseado em QoS não precisa ser acionado novamente, enquanto um caminho é considerado "desmarcado" em caso contrário. A questão que se impõe é saber quando uma rota deve ser marcada ou desmarcada. Deseja-se ainda que este processo seja feito de acordo com o protocolo de reserva utilizado, o RSVP. Desta maneira, no gerenciamento a ser descrito, a marcação e a desmarcação de rotas deve ser realizada consistentemente com os estados *soft* do RSVP, exigindo mudanças mínimas nas regras deste protocolo.

É necessário observar que certas modificações são inevitáveis, principalmente na interface entre o RSVP e o protocolo de roteamento (*Routing Support for Resource Reservations* - RSRR) [31]. Especificamente, o roteamento baseado em QoS requer, além dos endereços de origem e destino, no mínimo, conhecimento sobre as características de tráfego do fluxo (TSpec), e possivelmente também o tipo de serviço e outros parâmetros [32]. Assume-se que o TSpec do transmissor, pelo menos, seja adicionado aos endereços de origem e destino na RSRR. No mecanismo de gerenciamento de caminhos apresentado é feito o mínimo de mudanças na RSRR, e não se modificam os formatos das mensagens do RSVP, embora esta opção pudesse trazer benefícios [32].

Durante o processamento das mensagens *path*, o RSVP aciona o roteamento (baseado em QoS) para obter o próximo salto de envio destas mensagens. Como mencionado, o TSpec do transmissor é disponível para o roteamento, podendo o algoritmo selecionar o próximo salto ao longo do caminho para o destino que tenha condições de suportar a QoS especificada. Desta forma, o encaminhamento da mensagem *path* através deste próximo salto deve aumentar as chances de sucesso das mensagens de reserva a serem produzidas subseqüentemente, através do caminho reverso do percorrido pelas mensagens *path*.

Dentro do contexto de roteamento nó-a-nó descrito, duas questões são levantadas em termos da marcação/desmarcação de rotas com requisitos de QoS. A detecção de ciclos fechados (*loops*) pode ser feita através de inconsistências nas rotas retornadas pelo roteamento em diferentes nós. Tais inconsistências são tipicamente transientes, mas é importante que a marcação de rotas não resulte na formação de ciclos fechados permanentes. Outra questão diz respeito ao pedido de uma nova rota no caso de falhas. Um exemplo é a falha de uma reserva devido à mensagem *reserve* ter chegado substancialmente depois de a rota baseada em QoS ter sido inicialmente selecionada. Outras falhas incluem defeitos usuais nos enlaces, e em geral é importante permitir ao roteamento baseado em QoS ser um tanto imune a falhas e escolher uma melhor rota, se esta estiver disponível.

O gerenciamento de caminhos apresentado trata das duas questões através de duas regras de projeto. Primeiramente, o que é marcado é um caminho tomado por um fluxo RSVP específico e não a rota, calculada pelo algoritmo de roteamento. Sendo assim, a marcação e desmarcação são operações de responsabilidade do RSVP, e como resultado, são completamente independentes do algoritmo de roteamento específico sendo utilizado. Em segundo lugar, a marcação/desmarcação de caminhos é temporária, sendo submetida ao mecanismo de estado *soft* existente no RSVP. Ou seja, a marcação e desmarcação de caminhos é feita através dos mecanismos de temporização e *refresh* do estado da reserva existentes no RSVP. Estas mudanças são realizadas apenas em função da informação atualmente utilizada pelo RSVP.

Regras para o gerenciamento de caminhos

O estado de um caminho, como definido pelo RSVP, consiste em um *flag* utilizado para indicar se o caminho está atualmente marcado ou não. Um caminho marcado significa que o roteamento baseado em QoS não precisa ser acionado para obtenção de um próximo salto para o envio de uma mensagem *path* de *refresh*. As regras para a marcação/desmarcação de caminhos são:

- os caminhos são marcados durante o processamento de mensagens *path*;

- os caminhos são desmarcados quando:
    1. estados de caminho correspondentes são removidos (por estouro de temporizador ou explicitamente);
    2. alguns dos parâmetros recebidos nas mensagens *path* se modificam;
    3. um erro devido a falha no controle de admissão local é detectado após o recebimento de uma mensagem *reserve*;
    4. uma mensagem de erro é recebida;
    5. a notificação da falha de um enlace local pertencendo ao caminho é recebida.

As regras acima se traduzem em algumas pequenas modificações nas regras atuais do RSVP. Estas podem ser agrupadas em dois conjuntos, o primeiro deles, reunindo mudanças na interface entre o roteamento e o RSVP, como definida na RSRR [31]. A principal mudança diz respeito à inclusão do parâmetro TSpec na requisição de uma rota ao mecanismo de roteamento. O segundo conjunto de mudanças se refere às relacionadas com as regras de processamento das mensagens, em termos da marcação/desmarcação de caminhos:

Mensagens *Path* - Quando recebe a primeira mensagem *path*, o RSVP determina que não existe um estado do caminho para o fluxo. Este aciona então o roteamento baseado em QoS para obter o próximo salto através do "melhor"caminho disponível. Este próximo passo é armazenado como parte do estado do caminho com sua *flag* de marcação verdadeira.

Quando recebe uma mensagem *path* de *refresh*, o RSVP checa se as mudanças no estado do caminho são relevantes para o roteamento baseado em QoS. Se não há mudanças e o próximo passo atual está marcado, este será utilizado para enviar adiante a próxima mensagem *path*. Se o estado do caminho mudou ou o próximo salto atual está desmarcado, o RSVP consulta o roteamento novamente para obter um novo próximo salto a ser usado no envio da mensagem *path*. De forma semelhante, no momento em que uma mensagem *path* deve ser enviada, o RSVP checa se o próximo salto atual está marcado ou não. Se estiver, é utilizado no envio desta mensagem, se não, o roteamento é outra vez consultado.

Mensagens *Reserve* - A única mudança necessária é para o caso de a reserva de recursos falhar. De acordo com a especificação atual, uma mensagem RESV_ERR com o código de erro "falha de controle de admissão" é enviada no sentido dos receptores. No entanto, algum processamento adicional é necessário para a seleção de um melhor caminho, se este existir. Este começa com a desmarcação do próximo salto atual, e prossegue em uma de duas formas: a tentativa, ou não, de um reparo local no caminho com requisitos de QoS.

No caso de um reparo local ser tentado, o RSVP consulta novamente a tabela de roteamento baseado em QoS local. Se um próximo salto diferente for retornado, ou seja, a reserva pode ter sucesso desta vez, então o reparo local é tentado através da marcação do novo próximo salto e o envio de uma mensagem *path* através da nova rota. Se no entanto o mesmo próximo salto for retornado, então o reparo local falhou. Neste caso, ou quando o reparo local não é tentado, o próximo salto atual é desmarcado no estado do caminho. Além disso, uma mensagem PATH_ERR é enviada no sentido da fonte com um código de erro de "falha de caminho", indicando a indisponibilidade de recursos. A recepção desta mensagem provoca a desmarcação do caminho no roteador acima deste nó, na direção da fonte. Isto assegura que o roteamento será acionado quando a próxima mensagem *path* for gerada.

Notificação de Mudança de Rota - Uma notificação de mudança de rota é acionada quando o roteamento baseado em QoS detecta que um enlace utilizado atualmente por um caminho com requisitos de QoS falhou. Quando recebe tal notificação, o RSVP imediatamente desmarca o próximo salto atual. Como no caso da falha de uma reserva, o RSVP pode primeiramente tentar o reparo local, isto é, acionar o roteamento para a descoberta de um novo próximo salto. Se um próximo salto diferente for retornado, o RSVP utiliza-o na substituição do próximo salto anterior, marca-o, e gera uma mensagem *path* na direção deste novo próximo nó. Se o roteamento responder que não há rota disponível para o destino, ou se o reparo local não for tentado, o RSVP envia na direção da fonte uma mensagem PATH_ERR com código de erro indicando "falha de enlace".

Mensagem de Erro de Caminho - Como mencionado anteriormente, a única modificação no processamento da mensagem PATH_ERR, é o reconhecimento do código de erro de falha de caminho com requisitos de QoS, e a desmarcação do próximo salto associado. Esta desmarcação força o consulta ao roteamento no processamento da próxima mensagem *path*.

II.3.3 Impacto do roteamento baseado em QoS no caminho de dados

A utilização do roteamento baseado em requisitos de QoS afeta apenas a escolha do caminho para os dados e não a maneira como estes dados são realmente enviados. Apesar disso, existe um aspecto importante a ser observado. Enquanto mensagens *path* são encaminhadas imediatamente através do próximo salto retornado pelo roteamento baseado em QoS, o mesmo não ocorre necessariamente com pacotes de dados. Isto se deve ao fato de poderem ocorrer ciclos fechados temporários nos caminhos com requisitos de QoS. O envio de mensagens *path* através de caminhos que contenham ciclos fechados provoca um impacto mínimo nos roteadores e é, na verdade, útil na detecção e eliminação de tais ciclos. Porém, dependendo do quão rápido os ciclos fechados possam ser solucionados, pode ser interessante que o envio de pacotes de dados seja retardado até que a ausência de ciclos fechados possa ser verificada.

Como conseqüência, é proposto que a modificação que resultará no envio de pacotes através do ciclo seja atrasada até o momento da recepção de uma mensagem *reserve*. Como discutido mais adiante, a recepção de uma mensagem *reserve* implica a não existência de ciclos fechados no caminho com requisitos de QoS.

Tratamento e prevenção de ciclos fechados

É importante detectar a ocorrência de ciclos fechados e desmarcar o caminho com requisitos de QoS selecionado (próximo salto), de forma que os ciclos possam ser eventualmente eliminados quando os nós tenham convergido para uma informação consistente.

A seguir, é feita uma descrição de como as regras de processamento descritas anteriormente são eficientes na detecção e eliminação de ciclos fechados. Esta é realizada através de dois cenários de ciclos fechados, que respectivamente ilustram a utilização da identificação de mudanças nos campos PHOP (*previous hop - salto anterior*) e IP TTL (IP *time-to-leave*) [33] para detecção de diferentes ocorrências de ciclos fechados.

Cenário 1:

Seja a situação descrita na figura II.3. Assume-se que o roteador R2 (e possivelmente outros ao longo de RX...RY) possui informação de roteamento (métrica) inconsistente com a informação de R1. O roteador R1 identificou R2 como o próximo salto com a melhor QoS para o destino RH, baseado no fato de que o caminho com requisitos de QoS seria R1-R2-R3-...-RH. No entanto, devido à inconsistência na informação sobre as métricas, R2 identifica RX como o próximo salto no melhor caminho com requisitos de QoS para RH. Sendo assim, a mensagem *path* enviada para RX eventualmente retorna a R2 através de RY. Como resultado, um ciclo fechado temporário é criado e ficará em efeito enquanto a informação de próximo salto incorreta for mantida marcada. No entanto, uma vez que a mensagem *path* de RY para R2 chega com um valor de PHOP diferente, este dispara a desmarcação da informação de próximo salto em R2. O roteamento baseado em QoS é, portanto, acionado novamente a próxima vez que R2 tiver que enviar uma mensagem *path* para o fluxo, de maneira que o ciclo fechado é eventualmente (uma vez que a informação em R2 se torne consistente) removido.

É importante notar que a rapidez na remoção do ciclo fechado é diretamente dependente dos valores dos temporizadores do RSVP e também da maneira como

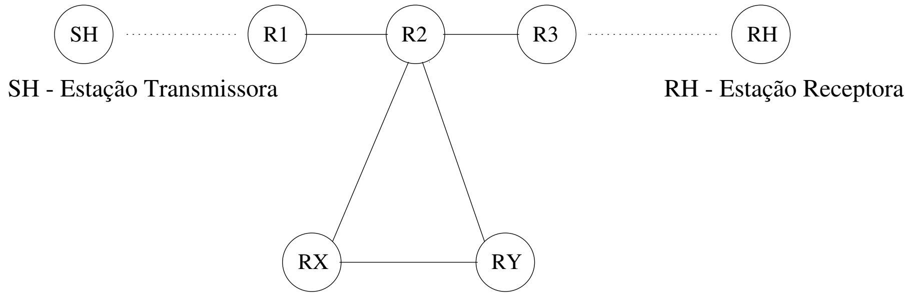


Figura II.3: Ciclo fechando na máquina causadora do erro.

estes são sincronizados entre os diversos roteadores. Por exemplo, R2 pode receber uma mensagem *path* de RY logo após ter recebido uma mensagem *path* de R1. Isto não afeta a desmarcação do próximo salto (RX), mas significa que RY será mantido como o salto anterior para o fluxo, pelo menos até o próximo *refresh* de R1 (se a informação de roteamento de R2 estiver agora consistente, as mensagens *path* serão encaminhadas para R3 e não para RX, de maneira que RY não receberá mensagens *path* de *refresh*, e, assim, também não vai enviá-las). No entanto, isto não vai criar um ciclo fechado na reserva, uma vez que as mensagens *reserve* retornam para R2 de R3, R2 determina que RX não é o próximo salto correto para este fluxo.

Cenário 2:

O cenário 1 mostrou o caso onde o roteador que causou o ciclo fechado era também o nó onde o ciclo se fechava. Seja agora a situação apresentada na figura II.4. O cenário 2 considera o caso no qual o ciclo não se fecha no roteador que o provocou em primeiro lugar.

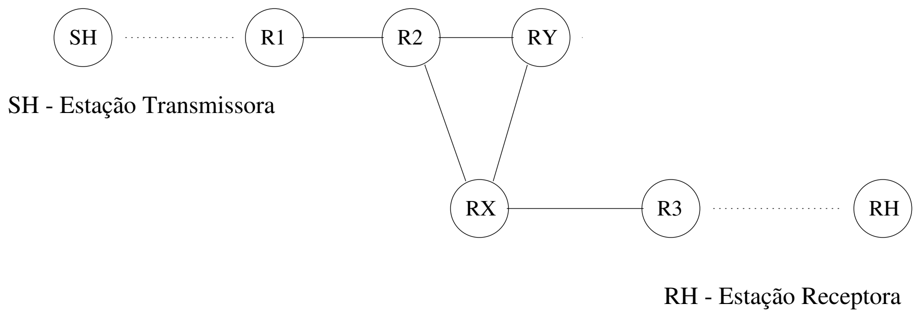


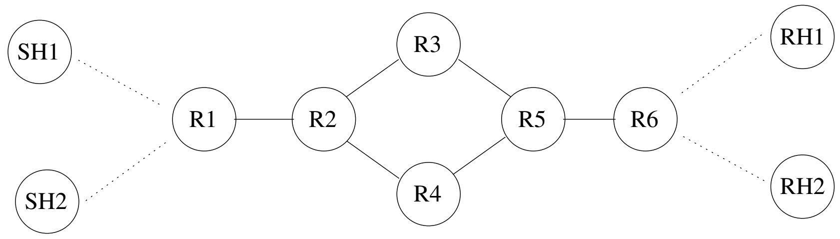
Figura II.4: Ciclo fechando em uma máquina diferente.

Neste caso, o caminho com requisitos de QoS correto de SH para RH é a rota SH...R1-R2-RX-R3...RH. No entanto, o roteador RX possui informação inconsistente, identificando RY como o melhor próximo salto. Isto eventualmente cria um ciclo fechando em R2, se RY identificar R2 como o melhor próximo salto para RH. Nesta situação, RX não identificará qualquer diferença no campo PHOP, e portanto este critério não pode ser utilizado para desmarcar a informação de próximo salto em RX. Porém, RX receberá mensagens *path* com diferentes valores de IP TTL, e este fato disparará a desmarcação do caminho de forma que quando a informação de roteamento em RX for atualizada, este poderá descobrir o próximo salto correto, R3.

Tratamento de condições de competição na alocação de recursos

O problema de condições de competição, ou seja, a existência de diversos caminhos competindo pelos mesmos recursos, é intrínseco à natureza distribuída das decisões do roteamento baseado em requisitos de QoS. O problema é agravado pelo atraso potencial entre a seleção de um caminho e o momento real da reserva dos recursos, ou seja, do recebimento de mensagens *reserve*. Conseqüentemente, é importante assegurar que o roteamento baseado em QoS e os mecanismos associados para o estabelecimento de caminhos com requisitos de QoS sejam suficientemente flexíveis para tratar tais condições de competição e mudar os caminhos se e quando for necessário.

Seja a topologia mostrada na figura II.5, onde o enlace entre R2 e R3 pode acomodar apenas um dos dois fluxos em questão por vez, e que o segmento R1-R2-R3-R5-R6 está marcado para ambos os fluxos antes que RH1 ou RH2 tenham gerado qualquer mensagem *reserve*. Em seguida, RH1 é o primeiro a gerar uma mensagem *reserve*, e esta reserva obtém sucesso no caminho inteiro. Quando RH2 gera sua própria mensagem *reserve*, sua reserva terá sucesso em R6, R5 e R3, mas falhará em R2. De acordo com as regras de processamento descritas anteriormente, R2 irá então desmarcar seu estado de próximo salto além de enviar uma mensagem RESV_ERR na direção do receptor. É importante notar que os estados do caminho e da reserva permanecem intactos no segmento nesta direção.



SH1 envia uma mensagem path para estabelecer o fluxo para RH1.
SH2 envia uma mensagem path para estabelecer o fluxo para RH2.

Figura II.5: Uma situação de competição por recursos.

Se em seguida R2 tentar o reparo local, vai acionar o roteamento baseado em QoS. Se o mesmo próximo salto, R3, for retornado, R2 conclui que o reparo local falhou e envia uma mensagem PATH_ERR na direção do transmissor com o código de erro de falha do caminho com requisitos de QoS, com o valor indicativo de QoS requisitada não disponível. A recepção desta mensagem PATH_ERR dispara a desmarcação do estado de próximo salto em todos os nós na direção do transmissor, ou seja, em R1 e SH2. Isto por sua vez vai definir o acionamento do roteamento baseado em QoS no momento do próximo *refresh* do caminho. Isto assegura que, se em algum tempo, recursos se tornarem disponíveis em R4 e esta informação for passada ao roteamento, então o fluxo entre SH2 e RH2 será roteado através deste caminho, isto é, R2 identificará R4 como seu próximo salto para este fluxo. Como resultado, uma mensagem *path* será enviada na interface para R4 de maneira que mensagens *reserve* vindas de R5 serão eventualmente enviadas para R4, e a reserva de recursos poderá obter sucesso. Um processo semelhante iria ocorrer se o roteamento baseado em QoS tivesse retornado R4 como o melhor próximo salto, quando R2 tentara realizar o reparo local.

Deve ser notado que R2 não enviou uma mensagem PATH_TEAR (de apagamento de um caminho) para R3 na ocasião da falha de reserva. Existem diversas razões para este procedimento. Primeiramente, o caminho através de R3 pode realmente ser o melhor atualmente disponível, e qualquer reserva feita com sucesso sobre este caminho até o momento provavelmente não deve ser desfeita (uma mensagem PA-

TH_TEAR acarretaria isso) a menos que o receptor o deseje. Em segundo lugar, manter as reservas também evita o problema de ter as reservas nos enlaces de R5 para R6 e de R6 para RH2 utilizadas por outros fluxos, enquanto o fluxo de SH2 para RH2 está em processo de troca de caminhos. Em vez disso, as reservas existentes no segmento na direção do receptor (R5-R6...RH2) serão reutilizadas, e os estados do caminho e da reserva para o fluxo em R3 serão descartados com o estouro de um temporizador. Por outro lado, manter as reservas pode afetar o o roteamento baseado em QoS. Isto porque este roteamento não saberá *a priori* que em alguns enlaces as reservas realmente disponíveis para o fluxo são maiores do que é atualmente informado. Porém, isto deve ser aliviado pelo fato de que os recursos informados são sempre imprecisos e também porque o roteamento retornará sempre o melhor caminho possível, mesmo que este não reconheça que os recursos necessários estão disponíveis neste caminho.

Pode-se concluir das análises realizadas nesta seção que diversas questões precisam ainda ser resolvidas no relacionamento entre os protocolos de roteamento baseado em requisitos de QoS e o protocolo de reserva de recursos, o RSVP. No entanto, algoritmos para a descoberta de rotas para o roteamento com QoS podem ser definidos independentemente destes problemas, que estão localizados na interface entre estes dois elementos da arquitetura de Serviços Integrados na Internet.

Capítulo III

Algoritmos de Roteamento

Os esquemas de roteamento pela origem (*source routing*) e roteamento salto-a-salto (*hop-by-hop routing*) estão presentes na maior parte das arquiteturas de roteamento para redes de transmissão de dados. O roteamento salto-a-salto é a forma mais comum de roteamento de propósito geral nas redes atuais, enquanto o roteamento pela origem é utilizado principalmente para o diagnóstico da rede e para realização de rotas com políticas especiais.

No esquema de roteamento pela origem, a rota de envio é calculada sob demanda para cada fluxo na fonte da transmissão e listada explicitamente no cabeçalho de cada pacote. Os pacotes são encaminhados de acordo com a informação em seu cabeçalho. Uma vez que o cálculo é realizado para cada requisição, de maneira centralizada, o roteamento pela origem é flexível. No entanto, a fonte necessita de informação relativa a todos os enlaces envolvidos no caminho, e os pacotes possuem um cabeçalho longo.

Já no roteamento salto-a-salto, os pacotes são encaminhados salto a salto a cada roteador. Cada nó possui uma tabela de roteamento com o próximo salto para todos os destinos, sendo em geral esta tabela atualizada periodicamente em resposta a mudanças na topologia da rede. Quando um pacote chega a um roteador, este precisa apenas checar na tabela qual a porta de saída (próximo salto) para o destino desejado. Em relação ao roteamento pela origem, o cabeçalho dos pacotes pode ser menor, uma vez que não carrega a descrição completa da rota a ser seguida. O roteamento salto-a-salto pode ser implementado através de algoritmos completamente distribuídos.

Estes dois esquemas de roteamento possuem características específicas que se encaixam nas necessidades do roteamento baseado em requisitos de QoS. Os requisitos de QoS das aplicações são bastante diversos, sendo difícil especificar um conjunto de parâmetros que sirva de base para a maioria das aplicações. Deste ponto de vista, o roteamento pela origem seria bastante apropriado, uma vez que o cálculo de rotas é realizado de acordo com cada fluxo, sob demanda. Por outro lado, o roteamento salto-a-salto permite a implementação de algoritmos distribuídos e possui as vantagens de menores sobrecarga de processamento e tempo gasto na inicialização. Existe também um tempo de cálculo durante a inicialização. Além disso, o roteamento pela origem é evitado por questões ligadas à segurança, sendo filtrado em *firewalls* de diversas organizações [13].

Neste capítulo, primeiramente é feita uma análise dos fatores de decisão na escolha da métrica ou conjunto de métricas adequado para a implementação de protocolos de roteamento baseado em requisitos de QoS. Em seguida, são discutidas diferentes maneiras de comsiderarem-se as métricas. Este tratamento pode ser realizado através da criação de uma medida que englobe as diferentes métricas, ou mediante a análise das métricas separadamente, por meio de algoritmos específicos.

III.1 Análise das Métricas Possíveis

As métricas escolhidas para o roteamento com requisitos de QoS definem basicamente a forma como a rede será representada. Esta decisão influirá não somente na complexidade do algoritmo de roteamento a ser implementado, mas também no conjunto de requisitos de QoS das aplicações que tal esquema será capaz de suportar. Diversos aspectos devem ser levados em conta na escolha da métrica ou do conjunto de métricas que constituirão o fator de decisão de roteamento [12].

Qualquer métrica escolhida deve permitir algoritmos eficientes para o cálculo das rotas, de forma a garantir a escalabilidade do protocolo de roteamento, fator indispensável quando leva-se em consideração o tamanho de uma rede como a Internet. De preferência, a complexidade dos algoritmos propostos para o roteamento baseado em requisitos de QoS não deve ser maior que a dos algoritmos utilizados nos protocolos de roteamento atuais.

As métricas utilizadas devem representar as características da rede, de forma a suportar requisitos de QoS básicos. Os parâmetros de QoS negociados deverão inevitavelmente ser mapeados na métrica (ou conjunto de métricas) usada pelo protocolo de roteamento na decisão de rotas. Desta forma, a escolha da métrica é fator determinante dos tipos de requisitos de QoS que poderão ser suportados. Por exemplo, se o atraso de transmissão e o custo forem as métricas, todos os requisitos de QoS das aplicações terão que ser mapeados em termos de custo e atraso. Neste tipo de métrica, um requisito do tipo probabilidade de perda de pacotes não poderia ser considerado.

No caso de algoritmos para múltiplas restrições, as métricas utilizadas devem ser disjuntas, sem qualquer informação redundante. Isto porque a informação redundante pode introduzir inter-dependência entre as métricas, o que torna impossível o cálculo de cada métrica independentemente. O cálculo de rotas torna-se mais complexo com a introdução de algoritmos recursivos entre as métricas.

III.2 Métrica Única Combinada

Algoritmos para o cálculo de rotas baseados em uma única métrica são bastante conhecidos, sendo implementados pela maioria dos protocolos de roteamento atuais. A solução mais natural seria, então, o mapeamento de diferentes requisitos de QoS do usuário em uma única medida, através da qual a descoberta de rotas poderia ser realizada utilizando algoritmos tradicionais.

Para a definição de uma tal métrica, poderia ser criada uma função geradora, que receba como parâmetros de entrada um conjunto de requisitos de QoS do usuário e forneça na saída uma medida única. Por exemplo, poderia ser definida a função de saída para cada caminho como sendo proporcional à faixa de transmissão disponível e inversamente proporcional ao atraso de propagação e à probabilidade de perda de pacotes:

$$f(c) = \frac{B(c)}{D(c) \times L(c)},$$

onde c é um determinado caminho, $B(c)$ é sua banda passante, $D(c)$ o atraso de transmissão associado a este caminho e $L(c)$ a probabilidade de perda de pacotes correspondente. O caminho com o maior valor de $f(c)$ seria o melhor de acordo com esta medida.

A métrica única combinada, no entanto, não poderia ser utilizada na prática, pois não contém informação suficiente para garantir que os requisitos de QoS do usuário estão sendo respeitados individualmente. Garantir que $f(c)$ tem um valor menor que determinado limite, ou o menor possível, não garante que cada uma das três métricas esteja abaixo (ou acima) dos valores desejados.

Outro problema advém das diferentes regras de composição para parâmetros de QoS diversos. Para dois caminhos diferentes, a métrica do caminho total será a soma das duas individuais, se esta representar o atraso de transmissão por exemplo. Se, no entanto, a característica de QoS for a banda passante, a métrica do caminho total será a menor das duas. Desta forma, a função definida acima não preserva nenhuma das duas regras de composição.

III.3 Métricas Múltiplas

A utilização de métricas múltiplas é capaz de modelar a rede de maneira mais precisa, porém o problema de encontrar um caminho dadas diversas restrições pode se tornar complexo [12]. A quantidade de diferentes requisitos de QoS é um fator de acréscimo da complexidade dos algoritmos. Além disso, os requisitos mais importantes dependem diretamente da aplicação em questão.

A complexidade dos algoritmos de roteamento será dependente da regra de composição das métricas, sendo três as regras de composição mais comuns: a aditiva, a multiplicativa e a côncava. Na aditiva, as métricas dos arcos componentes do caminho devem ser somadas para gerar a métrica do todo. Na multiplicativa, a métrica resultante é o produto das métricas dos enlaces componentes do caminho. Na regra do mínimo, também chamada de côncava, a métrica final possui o menor valor entre todas as métricas dos enlaces componentes da rota.

Os parâmetros mais comuns utilizados como requisitos de QoS do usuário são o atraso máximo suportado pela aplicação, a variação de atraso (*jitter*) admitida, o custo do serviço, a probabilidade de perda de pacotes máxima e a banda passante mínima necessária. O atraso, a variação de atraso e o custo seguem a regra de composição aditiva. A banda passante segue a regra de composição do mínimo. A regra de composição para a probabilidade de perda de pacotes é um pouco mais complexa. Seja c o caminho cuja métrica será calculada, $d(c)$ a métrica deste caminho, $d(i, j)$ a métrica associada ao arco entre o nó i e o nó j. Desta forma a regra de composição da probabilidade de perda será dada como a seguir:

$$d(c) = 1 - ((1 - d(i, j) \times (1 - d(j, k)) \times ... \times (1 - d(l, m))).$$

No entanto, se a probabilidade de perda for transformada na probabilidade de transmissão com sucesso, a métrica equivalente possui uma regra de composição multiplicativa. Em [12] é realizado um estudo acerca da complexidade dos algoritmos para n métricas de forma aditiva e de forma multiplicativa, mostrando que estes problemas são NP-completos. Além disso, é demonstrado que a combinação de n métricas aditivas com m métricas multiplicativas também não é solúvel por algoritmos em tempo polinomial.

Como conseqüência, pode-se concluir que dos requisitos de QoS citados anteriormente, atraso, *jitter*, custo, probabilidade de perda de pacotes e banda passante, as únicas combinações tratáveis seriam da banda passante com uma das outras métricas. Isto porque a banda passante possui uma regra de composição côncava, enquanto as outras são aditivas ou multiplicativas. Neste caso, podem ser construídos algoritmos para descoberta de rotas dadas duas restrições de QoS.

III.4 Roteamento com Duas Restrições de Qualidade de Serviço

A seguir, são apresentados algoritmos para o cálculo de rotas dadas duas restrições de QoS, sendo uma delas a faixa de transmissão, a outra podendo ser o atraso de propagação, o *jitter* ou o custo do enlace, uma vez que todas possuem regras de composição similares.

Na análise a seguir, a banda passante considerada deve ser entendida como a taxa de transmissão residual disponível para os novos fluxos em cada enlace. A métrica banda passante é então definida como a menor banda passante disponível em todos os arcos componentes de um determinado caminho.

O atraso possui dois componentes básicos, o atraso nas filas e o atraso de propagação propriamente dito. É importante notar que o atraso nas filas é função da menor banda passante e das características do tráfego. Desta forma, o atraso utilizado como métrica deve considerar apenas a componente devida ao tempo de propagação, uma vez que a correspondente ao tempo gasto nas filas já está considerada na métrica faixa de transmissão. Desta forma pode-se garantir a independência entre as métricas utilizadas.

A banda passante mínima de um determinado caminho pode ser entendida como a sua largura, enquanto que o atraso pode ser tomado como o comprimento do caminho. Desta forma, o problema a ser resolvido pelo algoritmo de roteamento passa a ser encontrar um caminho na rede mais curto que determinada restrição, e mais largo que um certo valor mínimo.

Para muitas aplicações, em especial as de tempo real, o atraso fim-a-fim é um dos requisitos de QoS mais importantes. A utilização da banda passante e do atraso de propagação como métricas constitui para estes algoritmos um compromisso entre a complexidade e a melhor caracterização dos requisitos de QoS do usuário.

III.4.1 Algoritmo para roteamento pela origem

Nesta seção, é apresentado um algoritmo para o esquema de roteamento pela origem que leva em conta como restrições a faixa de transmissão mínima necessária e o atraso máximo admitido. Seja $G = (N, E)$ um grafo representante da rede, com N nós e E enlaces. A cada arco (i, j) são associados dois números reais, b_{ij} como a banda passante disponível e a_{ij} como o atraso de propagação neste enlace. Para um arco (i, j) inexistente no grafo, será considerado que $b_{ij} = 0$ e $a_{ij} = \infty$. Dado um caminho $c = (i, j, k, ..., l, m)$, a largura do caminho, $w(c)$, será dada por $min[b_{ij}, b_{jk}, ..., b_{lm}]$ e seu comprimento será a soma dos atrasos de propagação, $l(p) = a_{ij} + a_{jk} + ... + a_{lm}$. Dadas duas constantes B para a faixa de transmissão mínima necessária e A para o atraso máximo aceitável, o problema se reduz a encontrar um caminho c entre i e m tal que $w(c) \geq B$ e $l(c) \leq A$.

O algoritmo consiste primeiro em descartar todos os arcos cuja $b_{ij} < B$. Todos os caminhos possíveis dentro do grafo resultante estarão desta forma respeitando o requisito de banda passante. O algoritmo deve então apontar um caminho que respeite o atraso máximo permitido, se ele existir. O algoritmo a seguir encontra um caminho entre o nó 1 e o nó m com banda passante não menor que B e atraso não maior que A. A_i é considerado o comprimento estimado do caminho entre o nó 1 e o nó i.

`Passo 1:` $\forall i, j : a_{ij} = \infty$ `se` $b_{ij} < B$.

`Passo 2:` $L = \{1\}; \forall i, i \neq 1 : A_i = a_{1i}$.

`Passo 3:` `Encontre` $k \notin L \mid A_k = \min_{i \notin L} A_i$.

`Se` $A_k > A$`, o caminho não pode ser encontrado e o algoritmo termina.`

`Se` L `contém o nó` m`, um caminho foi encontrado e o algoritmo termina.`

$L := L \cup \{k\}$.

`Passo 4:` $\forall i \notin L : A_i := \min[A_i, A_k + a_{ki}]$.

```
Vá para o passo 3.
```

O passo 1 elimina os arcos cuja faixa de transmissão é menor que B. Os passos 2-4 encontram o caminho de atraso mínimo para o nó m através do algoritmo de Dijkstra. O algoritmo termina se o nó m for incluído no caminho ou se o atraso ultrapassar o limite antes de alcançar o nó m. Cada passo no algoritmo requer da ordem de N operações, e no pior caso os passos são repetidos $N-1$ vezes. Sendo assim o número de operações total no pior caso é $O(N^2)$.

III.4.2 Algoritmos para roteamento salto-a-salto

Uma vez que o esquema de roteamento salto-a-salto pré-calcula entradas na tabela para todos os destinos possíveis, este deve suportar todos os pedidos de recursos possíveis. Usualmente, os algoritmos de roteamento salto-a-salto calculam o melhor caminho para todos os destinos. Com uma única métrica, a melhor rota pode ser encontrada facilmente. Com mais de uma métrica, no entanto, o melhor caminho, com todos os seus parâmetros possuindo valores ótimos, pode sequer existir. Sendo assim, torna-se necessária primeiramente a definição de uma prioridade entre as diferentes métricas para a escolha da melhor rota para um destino.

A precedência entre a banda passante e o atraso é de certa forma dependente da aplicação. De uma maneira geral, entretanto, o atraso nas filas é mais dinâmico e sensível ao tráfego, sendo a banda passante freqüentemente mais crítica para a maioria das aplicações multimídias. No caso de não haver banda passante suficiente, o atraso nas filas, e provavelmente a taxa de perdas, serão altos. Em contraste, se o atraso de propagação não puder ser satisfeito, o atraso total será maior, mas o acréscimo será previsível e estável. Sendo assim, falhar em qualquer dos dois requisitos aumentará o atraso total do sistema, no entanto as conseqüências da falta de banda passante podem ser mais graves. Sendo assim, nos dois algoritmos apresentados a seguir é dada precedência à faixa de transmissão necessária em detrimento do atraso de propagação admitido.

III.4.3 Algoritmo por vetor-distância

Nesta seção, é apresentado um algoritmo baseado em vetores-distância que considera duas restrições de QoS na descoberta de rotas, a banda passante disponível e o atraso de propagação. Algoritmos baseados em vetores-distância são implementados atualmente por protocolos como o RIP e o DVRP.

Considere-se $G = (N, E)$ o grafo representante da rede, definido como na seção III.4.1. Além disso, b_{ij} e a_{ij} são definidos como anteriormente. Seja o nó 1 a fonte da transmissão e h o número de arcos saindo do nó fonte. Sejam $B_i^{(h)}$ e $A_i^{(h)}$ a largura e o comprimento do caminho mais curto e mais largo escolhido entre o nó 1 e o nó i com h saltos. Por convenção, $B_1^{(h)} = \infty$ e $A_1^{(h)} = 0$ para todo h. Esta seria a "rota"do nó 1 para ele mesmo. Os algoritmos para descoberta do caminho mais largo e mais curto podem ser produzidos pela adição de um teste de menor comprimento quando forem encontrados caminhos com igual largura. O algoritmo baseado em vetores-distância é apresentado a seguir:

`Passo 1:` `Início:` $\forall i \neq 1 : h = 0$ `e` $B_i^{(0)} = 0$

`Passo 2:` `Encontre` $K = \{k \mid w(1, ..., k, i) = \max_{1 \leq j \leq N}[\min[B_j^{(h)}, bji]]\}$, $i \neq 1$.

`Passo 3:` `Se` $\#K > 1$, `encontre` $k \in K \mid l(1, ..., k, i) = \min_{1 \leq j \leq N}[A_j^{(h)} + a_{ji}]$, $i \neq 1$.

`Passo 4:` $B_i^{(h+1)} = w(1, ..., k, i)$ `e` $A_i^{(h+1)} = l(1, ..., k, i)$.

`Passo 5:` `Se` $h \geq E$, `o algoritmo termina.`

`Se não,` $h = h + 1$. `Vá para o passo 2.`

O passo 2 encontra o caminho de maior faixa de transmissão do nó 1 para cada nó i. Se mais de um caminho com a maior faixa de transmissão for encontrado, o passo 3 escolhe o caminho com o menor comprimento (atraso). O passo 4 atualiza a largura e o comprimento para o caminho mais largo e mais curto do nó 1 para cada nó i. O algoritmo continua até a rota ser encontrada ou h ultrapassar o número de enlaces presente no grafo.

III.4.4 Algoritmo baseado no estado dos enlaces

Nesta seção, é apresentado um algoritmo baseado no estado dos enlaces para duas restrições. Este tipo de algoritmo é implementado por protocolos como o OSPF.

Considere-se $G = (N, E)$ o grafo representante da rede, definido como na seção III.4.1. Além disso, b_{ij} e a_{ij} são definidos como anteriormente. Seja o nó 1 a fonte de dados, B_i e A_i a largura e o comprimento do caminho mais curto e mais largo do nó 1 para o nó i. Por convenção, $B_1 = \infty$ e $A_1 = 0$. O algoritmo baseado no estado dos enlaces é o seguinte:

`Passo 1:` `Início:` $\forall i \neq 1 : L = \{1\}, B_i = b_{1i}$ `e` $A_i = a_{1i}$.

`Passo 2:` `Encontre` $K = \{k \mid k \in L, B_k = \max_{i \notin L}[B_i]\}$.

`Passo 3:` `Se` $\#K > 1$`, encontre` $k \in K \mid l(1, ..., k, i) = \min_{j \in k}[A_{(1,...,j,i)}]$.

$L := L \cup \{k\}$. `Se` L `contém todos os nós, o algoritmo termina.`

`Passo 4:` $\forall i \notin L$, $B_i := \max[B_i, \min[B_k, b_{ki}]]$.

`Vá para o passo 2.`

O passo 2 encontra os nós com a faixa de transmissão máxima entre os nós provisoriamente marcados. Se mais de um nó for encontrado, o passo 3 escolhe o nó cuja rota possui a métrica mais curto e o marca permanentemente. O passo 4 atualiza os nós marcados provisoriamente em torno do novo nó permanentemente marcado.

Os dois algoritmos para o cálculo do caminho mais curto e mais largo são escaláveis. Nas duas versões o número de operações necessárias em cada iteração é proporcional às versões correspondentes dos algoritmos para o cálculo do caminho mais curto. Sendo assim, o tempo de processamento dos dois algoritmos é igual ao dos algoritmos para descoberta do caminho mais curto tradicionais. Estes algoritmos serão utilizados no capítulo seguinte na construção de outros algoritmos que consideram três métricas, em vez de duas. No capítulo seguinte a proposta destes algoritmos é realizada, para no capítulo V ser realizada a medição do desempenho do esquema.

Capítulo IV

Construção de Algoritmos com Três Restrições

NO capítulo anterior, foram apresentadas as principais dificuldades encontradas na definição de um protocolo de roteamento com requisitos de QoS. Além disso, foram discutidos os principais problemas no que diz respeito ao relacionamento entre o mecanismo de roteamento baseado em requisitos de QoS com o protocolo de reserva de recursos, o RSVP. Considerando as questões em aberto sobre este relacionamento e o enfoque dado a este trabalho, os algoritmos de roteamento apresentados neste capítulo não levam em conta a maneira como a relação com o protocolo RSVP está definida. Esta abstração é válida, uma vez que do ponto de vista da descoberta de caminhos com determinados requisitos de QoS, o RSVP não possui influência sobre o roteamento.

Quanto à construção de algoritmos com múltiplas restrições, um dos problemas principais na definição de uma arquitetura de roteamento com QoS escalável, alguns aspectos merecem destaque. A utilização de uma métrica única combinada seria descartada, devido às dificuldades impostas pelas diferentes regras de composição intrínsecas de cada métrica e pela dificuldade em se garantir que cada métrica é respeitada individualmente. Em seguida, dadas as considerações sobre a independência entre as métricas [12], foram apresentados três algoritmos que consideram duas métricas, sendo uma delas a faixa de transmissão, que possui uma regra de composição côncava. A outra métrica poderia ser o atraso, o *jitter*, ou a probabilidade de perda.

Outro aspecto importante quanto à escalabilidade do protocolo de roteamento diz respeito ao tamanho das tabelas de roteamento. No roteamento baseado em requisitos de QoS, a decisão de rotas é realizada de acordo com as características de cada fluxo. Uma solução natural seria a definição de tabelas de roteamento com as rotas utilizadas por cada fluxo. A utilização de um esquema deste tipo, no entanto, levaria à construção e manutenção de tabelas muito grandes, devido à quantidade de nós de cada sub-rede e à quantidade de usuários conectados a cada nó.

Uma solução de compromisso entre a precisão da rota utilizada pelo fluxo de acordo com seus requisitos de QoS e o tamanho das tabelas pode ser a manutenção nos roteadores de várias rotas possíveis para um destino. Estas rotas podem ser representativas de grupos de aplicações de acordo com a classe de requisitos de QoS do tráfego transportado. O tamanho das tabelas de roteamento é certamente maior que o utilizado pelos protocolos atuais, porém é menor do que no caso em que é adotada a solução de uma rota por fluxo.

Considerando as observações realizadas, um esquema híbrido pode ser proposto que serve de base a uma arquitetura de roteamento com QoS escalável [34, 35]. Neste mecanismo, podem ser utilizados três requisitos de QoS das aplicações. Esta solução constitui um compromisso entre a simplicidade do algoritmo utilizado e a precisão da definição da rota, de acordo com os requisitos de QoS de um determinado fluxo. Algumas modificações são realizadas nos algoritmos apresentados no capítulo anterior, através da utilização cuidadosa de uma métrica única combinada. Esta métrica leva em consideração o atraso de propagação e a probabilidade de perda de pacotes nos enlaces, que em conjunto com a banda passante torna o algoritmo capaz de descobrir rotas de acordo com as características de uma maior variedade de aplicações.

IV.1 Combinação de Métricas

A utilização da métrica única combinada é uma alternativa interessante uma vez que os algoritmos de roteamento já implementados poderiam ser utilizados sem maiores modificações. No entanto, existem dois empecilhos importantes. O primeiro deles diz respeito à definição de uma função matemática de combinação das métricas que seja compatível com as regras de composição de cada uma. Outro problema consiste em, dada uma métrica que é a combinação de duas ou mais métricas, poder-se garantir que uma determinada rota está garantindo os requisitos de QoS para cada uma das métricas. Em outras palavras, no caso geral, a escolha de um caminho que respeite um limite para a métrica única combinada não garante que este caminho respeite cada uma das métricas isoladamente.

As principais métricas consideradas até então são a faixa de transmissão, o atraso de propagação, a variação de atraso e a probabilidade de perda. A faixa de transmissão possui uma regra de composição côncava. O atraso de propagação e o *jitter* possuem ambos regra de composição aditiva (assim como o custo dos enlaces). Já a probabilidade de perda possui uma regra de composição mais complexa, como mostrado na seção III.3.

No entanto, a probabilidade de perda pode ser considerada de outra forma, como sendo a probabilidade de transmissão com sucesso. Neste caso, a regra de composição passa a ser multiplicativa. Seja c o caminho cuja métrica será calculada, $S(c)$ a probabilidade de transmissão com sucesso neste caminho, $S(i,j)$ a métrica associada ao arco entre o nó i e o nó j, a regra de composição da probabilidade de transmissão com sucesso será a da equação IV.1.

$$S(c) = (S(i,j) \times S(j,k) \times ... \times S(l,m)). \qquad \text{(IV.1)}$$

Se for tomado o logaritmo desta métrica, sua regra de composição passa a ser aditiva. Desta forma, a combinação desta métrica com outras métricas aditivas passa a ser viável, do ponto de vista da composição de métricas. Pode-se definir a probabilidade de transmissão com sucesso logarítmica:

$$Slog(c) = log(S(c)) = (log(S(i,j)) + log(S(j,k)) + ... + log(S(l,m))). \qquad \text{(IV.2)}$$

É importante notar os efeitos que a transformação logarítmica acarreta. A composição aditiva é desejável, mas outro aspecto deve ser observado. O valor desta métrica, por se tratar de uma probabilidade, estará sempre entre 0 e 1. Quando é tomado o logaritmo na base 10, este intervalo é transformado no intervalo de $-\infty$ a 0.

Números com valor absoluto tão grande podem constituir um problema para o cálculo das métricas. Mas se o intervalo considerado para a probabilidade de transmissão com sucesso for reduzido, para entre 0.1 e 1 por exemplo, o intervalo de transformação pode ser restrito, ficando entre -1 e 0. Esta restrição é bastante razoável, uma vez que dificilmente a probabilidade de transmissão com sucesso de um caminho será tão baixa, da ordem de 10%. Isto considerando-se topologias onde as rotas não possuam um número de saltos muito grande. Pode-se então considerar que o valor absoluto da probabilidade logarítmica estará sempre entre 0 e 1. A mesma faixa pode ser esperada para a probabilidade de transmissão com sucesso de um caminho composto de diversos enlaces. Esta probabilidade será sempre menor que a menor das métricas dos enlaces envolvidos, ficando mais próxima de zero.

IV.2 Métrica Única para Atraso e Probabilidade de Perda

Dadas as restrições mencionadas anteriormente, pode ser definida uma métrica única combinada para a probabilidade de perda e o atraso de propagação. A métrica consiste da soma do atraso de propagação, dado em microssegundos, com a probabilidade de transmissão com sucesso logarítmica. Tomando a unidade de atraso de transmissão como o microssegundo, pode-se restringir esta medida a números inteiros (se conveniente, pode-se utilizar uma unidade menor). Esta medida é compatível com os atrasos esperados na Internet.

A métrica única combinada é a soma da probabilidade de transmissão com sucesso logarítimica com o atraso de transmissão. A regra de composição das duas métricas é aditiva, não causando problemas, sendo a mesma regra de composição da métrica única combinada.

Na definicão da métrica única combinada da probabilidade de transmissão com sucesso logarítmica e do atraso de propagação de acordo com as hipóteses citadas, o outro problema da utilização deste tipo de métrica é eliminado. A primeira métrica (probabilidade de transmissão com sucesso logarítmica) será sempre representada por números reais entre 0 e 1, e a outra (atraso de propagação) sempre por números inteiros. Desta forma, na métrica total de um caminho poder-se-á sempre distingüir a parte devida a uma e a outra métrica. Esta é outra das desvantagens que eram impostas à utilização de uma métrica única combinada, podendo desta forma ser contornada. Além disso, para as duas métricas que estão sendo combinadas, o melhor caminho é o que possui o menor valor para a métrica. Sendo assim, definir a métrica única combinada como a soma das duas métricas componentes é válido.

Com a definição de uma métrica única combinada para o atraso e a probabilidade de perda, pode-se, através de um esquema semelhante ao utilizado anteriormente, construir algoritmos que levem em conta também a faixa de transmissão envolvida. A idéia é a mesma, ou seja, realizar uma busca prévia no grafo representativo da sub-rede eliminando os enlaces que não possam prover a faixa de transmissão requisitada. Desta forma, estes arcos são eliminados do grafo. Em seguida, um algoritmo semelhante aos tradicionais, com pequenas modificações, pode ser utilizado. Estas mudanças irão garantir, como mostrado na seção seguinte, que as duas métricas que foram combinadas sejam respeitadas individualmente, a última das desvantagens associadas à utilização de uma métrica única combinada.

IV.3 Novos Algoritmos

Nesta seção, são apresentadas modificações nos algoritmos para roteamento pela origem e para roteamento salto-a-salto mostrados no capítulo anterior, de forma a levar em consideração a utilização da métrica única combinada definida para o atraso de transmissão e a probabilidade de perda, que em conjunto com a banda passante definem um esquema de roteamento dadas três métricas.

IV.3.1 Roteamento pela origem

Considere-se em primeiro lugar o algoritmo para roteamento pela origem. Seja $G = (N, E)$ um grafo representante da rede, com N nós e E enlaces. A cada arco (i, j) são associados dois números reais, b_{ij} como a banda passante disponível, $Slog(i, j)$ a probabilidade de transmissão com sucesso logarítmica, e um número inteiro, $D(i, j)$, o atraso de transmissão. Além disso é definido u_{ij} como a métrica única combinada (equação IV.3) resultante da soma do atraso de propagação, um inteiro, com a probabilidade de transmissão com sucesso logarítmica, um real entre 0 e 1.

$$u_{i,j} = Slog(i, j) + D(i, j). \tag{IV.3}$$

Para um arco (i, j) inexistente no grafo, é considerado que $b_{ij} = 0$ e $u_{ij} = \infty$. Dado um caminho $c = (i, j, k, ..., l, m)$, a largura do caminho, $w(c)$, é dada por $min[b_{ij}, b_{jk}, ..., b_{lm}]$ e seu comprimento é a soma das métricas de cada um dos enlaces, $l(p) = u_{ij} + u_{jk} + ... + u_{lm}$. Sejam três constantes, B para a faixa de transmissão mínima necessária, A para o atraso de propagação máximo admitido e P para a probabilidade de transmissão com sucesso logarítmica máxima (que corresponde a uma probabilidade de perda de pacotes máxima). Desta forma, pode-se definir U como o valor máximo aceitável da métrica única combinada soma de A com P. O problema se reduz a encontrar um caminho c entre i e m tal que $w(c) \geq B$ e $l(c) \leq U$, como originalmente.

`Passo 1:` $\forall i, j : u_{ij} = \infty$ `se` $b_{ij} < B$.

`Passo 2:` $L = \{1\}; \forall i, i \neq 1 : U_i = u_{1i}$.

`Passo 3:` `Encontre` $k \notin L \mid U_k = \min_{i \notin L} U_i$.

`Se` $U_k > U$`, o caminho não pode ser encontrado e o algoritmo termina.`

`Se` L `contém o nó` m`, um caminho foi encontrado e o algoritmo termina.`

$L := L \cup \{k\}$.

`Passo 4:` $\forall i \notin L : U_i := \min[U_i, U_k + u_{ki}]$.

`Vá para o passo 3.`

O algoritmo como mostrado acima não garante que a probabilidade de transmissão com sucesso será maior que a desejada (ou logaritmicamente, menor). No entanto, como o atraso está representado por um número inteiro, seu limite estará sempre respeitado. Porém quando dois caminhos possuirem um mesmo atraso, está garantido que a probabilidade de perda do enlace escolhido será sempre a menor.

Considere-se por exemplo que o caso em que o limite para o atraso (A) seja 6 e o limite para a probabilidade de transmissão com sucesso logarítmica (P) seja 0.5. O limite máximo para a métrica única combinada é $U = 6.5$. Dados dois caminhos de mesma banda passante e métrica única combinada com valores 4.6 e 5.4, o algoritmo acima elegeria o primeiro caminho como a melhor rota, pois ambos respeitam o limite U, e além disso $4.6 < 5.4$. Esta rota respeita o limite de atraso A, no entanto não está de acordo com o limite de P para a probabilidade de transmissão com sucesso logarítmica. O outro caminho está respeitando os dois limites, embora possua um valor maior para a métrica única combinada. Este outro caminho seria portanto uma melhor rota.

Uma pequena modificação pode ser incluída no algoritmo, de forma que a probabilidade de transmissão com sucesso seja garantida. O requisito de probabilidade de perda se traduz em um valor (absoluto) máximo para o logaritmo da probabilidade de transmissão com sucesso. Ou seja, a soma desta métrica para todos os enlaces de um caminho deve ser menor que um limite, definido pelo requisito de QoS da aplicação.

Considere-se $Slog(i, j)$ como o logaritmo da probabilidade de transmissão com sucesso no enlace entre os nós i e j, e P como o seu limite máximo. Dado um caminho c com x saltos, se $Slog(i, j)$ para cada caminho for menor que P/x, $Slog(c)$ será menor que P. No entanto, no momento em que um caminho está sendo calculado, o seu tamanho não é conhecido ainda. Uma primeira idéia seria então restringir $Slog(ij)$ a P/C, onde C é o tamanho do maior caminho no grafo representativo da sub-rede. Limitando-se $Slog(ij)$ sempre a P/C, a cada passo, garante-se que para o caminho c, $Slog(c)$ será menor que P. Porém este limite pode ser muito restritivo. O caminho entre dois nós pode ser no caso geral menor que C, sendo assim sua probabilidade de perda estará mais restrita que o necessário.

Um esquema alternativo pode relaxar um pouco esta restrição. A cada passo do algoritmo, define-se um "saldo" de saltos (SS) e um "saldo" de probabilidade logarítmica (SP). O saldo de saltos consiste na diferença entre o tamanho do maior caminho no grafo, C, e o número de saltos do caminho no passo atual do algoritmo. O saldo de probabilidade logarítmica é a diferença entre o limite exigido e a probabilidade de transmissão com sucesso logarítmica total do caminho construído até o momento. A probabilidade logarítmica máxima para o próximo enlace, a cada passo, não será mais P/C, mas SP/SS (cujo valor inicial é P/C). Este valor pode ser maior que P/C, caso o SP seja menor que $SS \times (P/C)$. Assim, a probabilidade de se encontrar uma rota no grafo da sub-rede aumenta. Desta forma, o algoritmo deve ser modificado:

`Passo 1:` $\forall i, j : u_{ij} = \infty$ `se` $b_{ij} < B$.

$SS = C; SP = P.$

`Passo 2:` $L = \{1\}; \forall i, i \neq 1 : U_i = u_{1i}.$

`Passo 3:` `Encontre` $k \notin L \mid (U_k = \min_{i \notin L} U_i)$ `e` $((U_k - \lfloor U_k \rfloor) < (SP/SS)).$

`Se` $U_k > U$`, o caminho não pode ser encontrado e o algoritmo termina.`

`Se` L `contém o nó` m`, um caminho foi encontrado e o algoritmo termina.`

$L := L \cup \{k\}.$

$SS := SS - 1.$

$SP = SP + (U_k - \lfloor U_k \rfloor).$

`Passo 4:` $\forall i \notin L : U_i := \min[U_i, U_k + u_{ki}].$

`Vá para o passo 3.`

Sendo assim, a restrição quanto à probabilidade de transmissão com sucesso continua sendo mais rígida, para caminhos menores que o caminho máximo, que a restrição necessária realmente. Esta restrição é no entanto mais relaxada, e garante que ambas as métricas serão respeitadas sem aumentar a complexidade do algoritmo proposto. Esta imprecisão na descoberta de rotas constitui uma solução de compromisso com a complexidade do algoritmo utilizado.

O algoritmo proposto leva o mesmo número de passos para encontrar uma rota que o algoritmo original para duas métricas [12]. A complexidade computacional adicionada pelo cálculo da função combinada das duas métricas é pequena, considerando-se que seu valor pode ser obtido antes que o algoritmo iterativo seja executado.

IV.3.2 Roteamento por vetor-distância

Considere-se agora o algoritmo para roteamento salto-a-salto baseado em vetores-distância. As mesmas considerações para o grafo $G = (N, E)$ feitas na seção anterior permanecem válidas.

Seja o nó 1 a fonte da transmissão e h o número de arcos saindo do nó fonte. Sejam $B_i^{(h)}$ e $U_i^{(h)}$ a largura e o comprimento do caminho mais curto e mais largo escolhido entre o nó 1 e o nó i com h saltos. Agora, $B_i^{(h)}$ será a métrica representante da banda passante disponível no enlace e $U_i^{(h)}$ a métrica única combinada que representa o atraso de propagação e a probabilidade de perda de pacotes de cada enlace, como definidos anteriormente. Por convenção, $B_1^{(h)} = \infty$ e $U_1^{(h)} = 0$ para todo h ("rota" do nó 1 para ele próprio). Os algoritmos para descoberta do caminho mais largo e mais curto podem ser produzidos pela adição de um teste de menor comprimento quando forem encontrados caminhos com igual largura. Nesse caso, o menor comprimento significa a métrica única combinada menor.

Para garantir que ambas as métricas, o atraso e a probabilidade de perda, são satisfeitas é adicionado mais um teste. Seja P o limite para o logaritmo da probabilidade de transmissão com sucesso, e $Slog(ij)$ o logaritmo da probabilidade de transmissão com sucesso no enlace ij. Quando houver empate entre caminhos coma mesma banda, deve-se ter $Slog(ij) < P$. Assim, tomando-se a métrica única combinada, o atraso será mínimo, e a probabilidade de perda estará respeitada. O algoritmo baseado em vetores-distância modificado para a utilização da métrica única combinada será como a seguir:

`Passo 1:` `Início:` $\forall i \neq 1 : h = 0$ `e` $B_i^{(0)} = 0$

`Passo 2:` `Encontre` $K = \{k \mid w(1,...,k,i) = \max_{1 \leq j \leq N}[\min[B_j^{(h)}, bji]]\}$`,` $i \neq 1$.

`Passo 3:` `Se` $\#K > 1$`, encontre` $k \in K \mid l(1,...,k,i) = \min_{1 \leq j \leq N}[U_j^{(h)} + u_{ji}]$ `e` $Slog(1,...,k,i) < P$`,` $i \neq 1$.

`Passo 4:` $B_i^{(h+1)} = w(1,...,k,i)$ `e` $U_i^{(h+1)} = l(1,...,k,i)$.

`Passo 5:` `Se` $h \geq A$`, o algoritmo termina.`

`Se não,` $h = h + 1$. `Vá para o passo 2.`

O passo 2 encontra o caminho de maior faixa de transmissão do nó 1 para cada nó i. Se mais de um caminho com a maior faixa for encontrado, o passo 3 escolhe o caminho com o menor comprimento (neste caso, representado pela métrica única). Neste passo garante-se também o requisito de probabilidade de perda de pacotes, através do teste descrito anteriormente. O passo 4 atualiza a largura e o comprimento para o caminho mais largo e mais curto do nó 1 para cada nó i. O número de passos gasto pelo algoritmo é o mesmo do algoritmo mostrado no capítulo anterior para duas métricas.

IV.3.3 Roteamento baseado no estado dos enlaces

Considere-se agora o algoritmo para roteamento salto-a-salto baseado no estado dos enlaces. As mesmas considerações que foram realizadas na seção IV.3.1 para o grafo $G = (N, E)$ são feitas neste caso. Seja o nó 1 a fonte de dados, B_i e U_i a largura e o comprimento do caminho mais curto e mais largo do nó 1 para o nó i, de acordo com a utilização da métrica única combinada. Por convenção, $B_1 = \infty$ e $U_1 = 0$. O algoritmo baseado no estado dos enlaces pode ser modificado, sendo apresentado a seguir:

`Passo 1:` `Início:` $\forall i \neq 1 : L = \{1\}, B_i = b_{1i}$ `e` $U_i = u_{1i}$.

`Passo 2:` `Encontre` $K = \{k \mid k \in L, B_k = \max_{i \notin L}[B_i]\}$.

`Passo 3:` `Se` $\#K > 1$`, encontre` $k \in K \mid l(1, ..., k, i) = \min_{j \in k}[U_{(1,...,j,i)}]$ `e` $lp(1, ..., k, i) < P$.

$L := L \cup \{k\}$. `Se` L `contém todos os nós, o algoritmo termina.`

`Passo 4:` $\forall i \notin L$, $B_i := \max[B_i, \min[B_k, b_{ki}]]$.

`Vá para o passo 2.`

O passo 2 encontra os nós com a faixa de transmissão máxima entre os nós provisoriamente marcados. Se mais de um nó for encontrado, o passo 3 elege o nó cujo caminho é o mais curto e o marca permanentemente. O passo 4 atualiza os nós marcados provisoriamente em torno do novo nó permanentemente marcado.

IV.4 Considerações Finais

Os três algoritmos para o cálculo do caminho mais curto e mais largo apresentados neste capítulo são escaláveis. Nas três versões o número de operações necessárias em cada iteração é igual ao das versões correspondentes dos algoritmos para o cálculo do caminho mais largo e mais curto com duas restrições. Sendo assim, o tempo de processamento dos três algoritmos é igual ao dos algoritmos de caminho mais curto tradicionais.

Desta forma, garante-se a escalabilidade do mecanismo de roteamento proposto, em conjunto com a adoção de rotas por tipo de tráfego, em vez de rotas para cada fluxo. O esquema proposto apresenta como vantagem a utilização de três métricas, em contraposição aos algoritmos propostos na literatura [12] para duas métricas, sem no entanto aumentar o número de passos necessários no cálculo de uma rota. Além disso, o cálculo da função representativa da métrica única combinada pode ser realizado antes do algoritmo iterativo para cálculo de rotas. A utilização de três métricas, em especial da banda passante dos enlaces, seu atraso de transmissão e probabilidade de perda abrange grande parte das aplicações multimídias conhecidas atualmente. Os três algoritmos podem ser utilizados na construção de arquiteturas de roteamento com QoS escaláveis.

No capítulo seguinte a implementação do algoritmo proposto para roteamento salto-a-salto baseado em vetores-distância é apresentada, para em seguida ser realizada a medição do desempenho do esquema, comparada a uma solução onde as três métricas seriam consideradas separadamente.

Capítulo V

Implementação no Simulador NS

A ARQUITETURA proposta foi implementada através da utilização do simulador NS-2 (*Network Simulator Version 2*)[36]. O simulador NS-2 é um *software* de livre distribuição feito por um grupo de instituições liderado pela Universidade de Berkeley. É um simulador orientado por eventos que dá suporte à simulação de protocolos de transporte, roteamento e protocolos *multicast*. Na próxima seção, é realizada uma breve introdução ao simulador, para em seguida ser descrita a implementação da arquitetura proposta e feita a apresentação dos resultados obtidos.

V.1 O Simulador NS-2

O simulador NS-2 é escrito na linguagem C++, orientada a objeto, e utiliza como interface para comandos e configuração a linguagem OTcl (*Object Tool Command Language*), uma versão orientada a objeto da linguagem *script* Tcl. O *software* OTcl é desenvolvido pelo MIT (*Massachussets Institute of Technology*).

Além disso, são distribuídos em conjunto com o simulador os programas *xgraph* e *nam*. O primeiro é um gerador de gráficos para o ambiente X11. Este programa é útil para o desenho de gráficos representativos de diversos eventos durante a simulação, como os momentos de geração de pacotes nos nós, os momentos onde houve perda de pacotes nos diferentes enlaces, gerados a partir do arquivo de saída criado pelo simulador. Já o *nam* (*Network AniMator*) é um programa que recebe como entrada um arquivo de saída específico do simulador NS-2 e realiza uma "animação" com os diversos nós da rede e com a geração e transmissão de pacotes durante o tempo da simulação. O programa possui uma interface gráfica semelhante a de um vídeo-cassete, útil para ter uma idéia geral do funcionamento da topologia de rede simulada e detectar erros no roteamento dos pacotes.

V.1.1 Funcionamento básico

O simulador é invocado através do interpretador `ns`, uma extensão do *shell* `otclsh`. Uma simulação é construída através de um *script* OTcl. A classe `Simulator` é a interface principal para o mecanismo de simulação. Através dos métodos definidos para esta classe, a topologia da rede pode ser definida, as fontes e destinos de tráfego são criados e configurados, a simulação é disparada e as estatísticas decorrentes são coletadas. Através desta interface diferentes ações durante o tempo de simulação podem ser programadas.

O primeiro passo em uma simulação é criar uma instância da classe `Simulator`. As instâncias de objetos no `ns` são criadas através da chamada `new`, e destruídas através do método `delete`. Uma topologia de rede é composta de uma maneira geral por três tipos de blocos funcionais: nós, enlaces e agentes. A classe `Simulator` possui métodos para criar e configurar cada um destes elementos. Nós são criados através do método `node`, que automaticamente designa um endereço único para cada nó. Os enlaces são criados conectando nós na topologia através dos métodos `simplex-link` e `duplex-link`, que criam respectivamente enlaces de transmissão unidirecionais e bidirecionais. Os agentes são os objetos que controlam a simulação de fato. Os agentes podem ser entendidos como processos ou entidades de transporte que são executadas em nós, sejam roteadores ou estações. As fontes e destinos de tráfego e módulos de roteamento dinâmico são exemplos de agentes no simulador NS-2.

Os agentes são criados através da instanciação de objetos da sub-classe `Agent`.

Uma vez que os agentes são criados, devem ser conectados a nós através do método `attach-agent` da classe `Simulator`. Cada agente recebe um número de porta único entre os agentes de um nó. Assim, os pacotes gerados por este nó estarão identificados unicamente pelo endereço do nó e número de porta do agente. Este número é análogo ao número de porta utilizado pelos protocolos TCP/UDP. Cada agente possui parâmetros de configuração associados que podem ser modificados através de métodos próprios à sua classe. Por exemplo, para uma fonte CBR (*Constant Bit Rate*) a taxa de transmissão de pacotes pode ser escolhida.

Os eventos podem ser escalonados no simulador através dos procedimentos `at` e `rtmodel-at` da classe *Simulator*, ocorrendo em pontos espscíficos do tempo de simulação. Essas chamadas constituem um mecanismo flexível para a construção de simulações, através do qual fontes de tráfego podem começar/parar de transmitir em determinados momentos, estatísticas podem ser coletadas, rotas podem ser conhecidas, falhas e recuperações dos enlaces podem ser programadas e certas ações de reconfiguração podem ser tomadas.

A simulação é iniciada através do método `run`, e termina apenas quando não existem mais eventos a serem processados. Neste momento, a chamada ao método `run` retorna e o *script* OTcl pode terminar, ou executar outra simulação após alguma reconfiguração.

V.1.2 Estrutura do nó

Existe um procedimento no `ns` que permitir configurar os nós como *multicast* ou *unicast*. A implementação realizada neste trabalho foi baseada em um dos protocolos de roteamento *unicast*.

Cada nó *unicast* no simulador NS-2 possui a estrutura mostrada na figura V.1. Conceitualmente, existe uma fila de entrada de pacotes que estará conectada aos enlaces de recepção. O pacote deve em seguida passar por um classificador de endereços (`Addr Classifier`). Se o pacote for destinado a este nó, é passado ao classificador de portas (`Port Classifier`), que em seguida o envia ao agente correspondente.

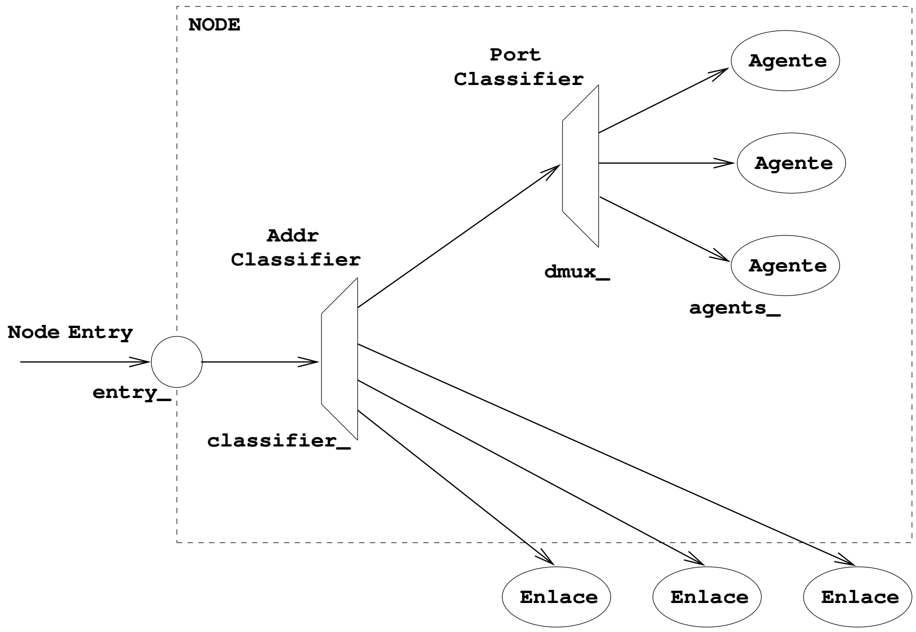


Figura V.1: Estrutura do nó no simulador NS-2.

Já no caso em que o pacote não possui como destino este nó e deve ser roteado, o classificador de endereços é o responsável por consultar a tabela de rotas e decidir por qual enlace de saída o pacote deve ser transmitido. Na figura V.1 estão representados também os nomes das variáveis associadas a cada elemento dentro da classe `Node`. Cada variável possui uma referência para os respectivos blocos do nó.

Quanto ao roteamento, são importantes na classe `Node` os métodos `add-route`, `add-routes` e `delete-routes`, usadas pelo simulador para manipular as tabelas de roteamento dos nós. A tabela de roteamento contém, para cada endereço, a referência para a entrada da fila de um dos enlaces de transmissão. No caso em que o destino é o próprio nó, a tabela contém a referência para o classificador de portas (`dmux_` na figura V.1).

No caso do roteamento dinâmico, cada nó possui associado a si uma instância da classe `rtObject`. Uma referência para este objeto é guardada na variável `rtObject_` da classe `node`. A rotina `init-routing` é responsável pela criação deste objeto. A classe `rtObject` contém os métodos responsáveis pelo cálculo das rotas e sua instalação no módulo de transmissão. Além disso, outro processo importante no roteamento dinâmico é o procedimento `intf-changed`, disparado quando ocorrem mudanças nos enlaces conectados a um determinado nó.

V.1.3 Protocolos de roteamento no NS-2

No simulador NS-2, os pacotes são enviados através da rota que representa o caminho mais curto entre a origem e o destino. A métrica utilizada é a soma dos custos dos enlaces que compõem o caminho atravessado. Como padrão, o custo de cada enlace é 1, o que significa que neste caso a métrica equivale ao número de saltos da rota. No entanto, o custo de cada enlace pode ser modificado através do método `cost` chamado no *script* da simulação.

Como padrão, o protocolo de roteamento implementado é estático, o que signfica que o estado dos nós e enlaces da topologia não podem ser alterados durante o tempo de simulação. Existem porém métodos para configurar uma topologia dinâmica onde enlaces ou nós podem falhar em momentos específicos da simulação.

A arquitetura de roteamento do simulador NS-2 possui três protocolos de roteamento *unicast*. Dois deles são baseados no algoritmo de Dijkstra, enquanto o terceiro implementa o algoritmo distribuído baseado em vetores-distância de Bellman-Ford. Este é o algoritmo utilizado por protocolos de roteamento em uso atualmente como o RIP e o DVRP [8, 37].

No entanto, apenas um dos protocolos de roteamento implementados no simulador NS-2 é dinâmico. Este foi nomeado no simulador como roteamento DV (*Distance-Vector*). Neste esquema, é possível determinar momentos em que determinado enlace da topologia irá falhar e momentos no qual será reabilitado. Isto é possível de forma determinística ou probabilística, através do procedimento `rtmodel`. Este mecanismo foi utilizado para produzir perdas de pacotes, em determinados momentos, que servirão como medida da eficiência do algoritmo de roteamento.

V.2 Implementação

A implementação da arquitetura de roteamento baseado em requisitos de QoS proposta foi realizada através da modificação do código do simulador NS-2. O mecanismo de roteamento utilizado foi o *Distance-Vector*, por ser o único protocolo dinâmico no NS-2. Como será visto mais adiante, o esquema dinâmico foi utilizado para medir o desempenho da arquitetura proposta. Esta implementação envolveu a alteração de diferentes partes do simulador.

Primeiramente, foi necessário alterar a implementação dos enlaces de transmissão. Cada enlace é caracterizado, dentro do mecanismo de roteamento do NS-2, através de um custo. Este custo é utilizado como a métrica considerada nas decisões de roteamento tomadas pelo protocolo. As alterações realizadas permitiram que a cada enlace da topologia sejam associados três custos, em vez de um. No esquema implementado, convencionou-se que o custo 0 representa o atraso de propagação de cada enlace, o custo 1, a faixa de transmissão disponível e o custo 2 a probabilidade de perda de pacotes do enlace.

A seguir, foi necessária a alteração do classificador de pacotes implementado na camada de rede do NS-2. O classificador é responsável por checar o endereço de destino de cada pacote, consultar a tabela de roteamento e de acordo com este endereço decidir qual a porta de saída pela qual o pacote deve ser enviado. Foram definidas três classes de serviço para a arquitetura proposta. Na implementação original, há uma variável do tipo *array* que armazena as rotas, *slot_*. Na arquitetura proposta, foram adicionadas mais duas variáveis, formando uma tabela com três linhas por destino.

Desta forma, para cada destino na tabela de rotas de um nó, existirá não uma entrada, mas três, cada uma delas correspondendo a uma classe de serviço. Assim, a rota para cada destino pode ser diferente de acordo com cada classe de serviço. As classes de serviço foram construídas de uma forma simples, apenas para que os algoritmos pudessem ser definidos. Na primeira classe de serviço, o atraso é considerado como a métrica mais importante. Sendo assim, a rota considerada melhor para esta classe de serviço é a que possuir o menor atraso dentre as rotas entre dois nós. Para rotas com igual atraso, a que possuir a maior banda passante é escolhida, e de forma semelhante a probabilidade de perda de pacotes é utilizada como o próximo critério de desempate. Na segunda, a banda passante máxima é prioritária no cálculo de rotas, e as outras duas métricas são utilizadas como critério de desempate. Já para a terceira classe de serviço, o mais importante seria a confiabilidade, sendo assim a probabilidade de perda de pacotes é minimizada na descoberta de rotas e após as outras duas métricas.

No entanto, as maiores alterações foram feitas na implementação do esquema de roteamento DV. Duas soluções distintas foram implementadas. Em ambas, as três métricas foram consideradas independentes entre si. Na primeira solução, cada métrica é minimizada separadamente. A ordem em que estas métricas são consideradas depende da classe de serviço em questão. Por exemplo, para a classe de serviço 0, o mais importante é o atraso de propagação. Sendo assim, é definida como a rota para um destino aquele caminho que em primeiro lugar apresentar o menor atraso. Depois, é considerado o caminho que possua mais banda passante dentre os que possuem o menor atraso, e em seguida o que possui a menor probabilidade de perda de pacotes. A ordem de busca pelas métricas é alterada de acordo com o que é mais importante para as outras duas classes de seviço. Esta solução seria a mais direta, porém não escalável uma vez que a quantidade de passos necessários para a descoberta de rotas será maior que a dos algoritmos tradicionais. Este algoritmo, referenciado por *3M* (*Three Metric algorithm*) daqui em diante, serviu como o algoritmo de controle para que o desempenho dos algoritmos propostos pudesse ser avaliado.

A segunda solução implementada é baseada na utilização de uma métrica única combinada como proposto nos dois capítulos anteriores. Esta métrica única considera os requisitos de atraso de propagação e perda de pacotes nos enlaces. O algoritmo será referenciado por SMM (*Single Mixed Metric algorithm*). As rotas descobertas pelos algoritmos *3M* e SMM não são, necessariamente, as mesmas nas duas soluções. Isto devido ao fato da utilização da métrica única combinada ter sido realizado através de uma aproximação, como foi mostrado em IV.3. No entanto é garantido que estas rotas estão de acordo com os limites determinados para as três métricas de acordo com cada classe de serviço.

O mecanismo de roteamento do NS-2 não possui um suporte de tempo real. Em outras palavras, o tempo gasto no cálculo das rotas não é levado em consideração na simulação. Desta forma, foram necessárias algumas modificações em seu código de maneira a medir a eficiência dos dois esquemas.

Dado que o esquema de roteamento utilizado é dinâmico, quando existe a falha de um enlace ou nó na rede o protocolo deve se adaptar e alterar as tabelas de rotas. Durante este tempo, podem existir perdas de pacotes devido às rotas estarem inconsistentes. De forma a forçar estas perdas, o código do NS-2 foi alterado. Na implementação original, quando um enlace falha, os nós conectados a este enlace são avisados, recalculam suas rotas, atualizam suas tabelas de roteamento e enviam mensagens aos outros nós alertando sobre as mudanças ocorridas. No entanto, o tempo gasto na descoberta das rotas não é computado. Desta forma não ocorrem perdas de pacotes, salvo aquelas devidas aos pacotes que estavam em trânsito no enlace que falhou no momento em que ocorreu a falha.

Sendo assim, foram realizadas alterações de forma que a atualização das rotas não seja realizada instantaneamente após o seu cálculo, quando um enlace falha. Foi introduzida uma variável nos algoritmos iterativos de descoberta de rotas, que armazena o número de passos gasto neste processo. Com o valor desta variável, é produzido um atraso na atualização das rotas, proporcional ao número de passos gastos. Desta forma, durante algum tempo, pacotes serão enviados por um enlace que falhou e serão perdidos. Este tempo com rotas inconsistentes é diretamente proporcional ao tempo gasto pelos algoritmos para calcular as rotas, sendo uma medida da eficiência destes algoritmos.

V.3 Resultados Obtidos

Para a medição das perdas de pacotes experimentadas quando cada um dos algoritmos de roteamento é utilizado, foram construídas diferentes topologias. Foi utilizada uma única fonte de tráfego CBR, de forma que o número de pacotes perdidos, dado um intervalo de tempo com rotas inconsistentes, é bem determinado (a fonte utilizada é determinística). Foi escolhido sempre um par fonte-destino para o qual os dois mecanismos de roteamento tenham escolhido a mesma rota. Em seguida, cada um dos enlaces componentes deste caminho são desligados. O roteamento deve então se adaptar. As perdas de pacotes experimentadas por cada um dos esquemas de roteamento, para cada enlace em falha, foram medidas. Foram realizados testes com duas topologias, uma delas com quinze nós e a outra com dez nós.

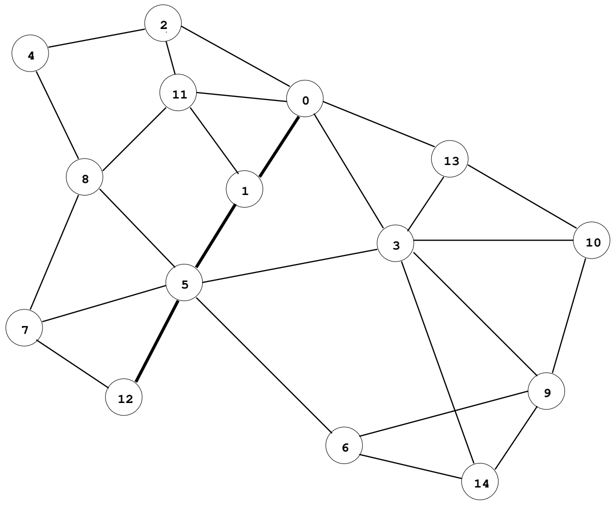


Figura V.2: Topologia de teste com 15 nós.

A primeira topologia utilizada, com quinze nós, está representada na figura V.2. Uma fonte de tráfego CBR foi colocada enviando pacotes do nó 0 para o nó 12. A rota utilizada pelo tráfego nos dois algoritmos é $0-1-5-12$. A topologia foi testada, então, para o caso em que cada um dos três enlaces da rota escolhida entre os dois nós falha. Os experimentos foram realizados para as duas versões do simulador, uma delas implementando o algoritmo proposto (SMM), a outra implementando a busca métrica por métrica (*3M*).

Como pode ser visto na tabela V.1, os resultados obtidos pelo algoritmo SMM são bem melhores que os obtidos pelo algoritmo *3M*. Através destas medidas, pode ser comprovada a escalabilidade do mecanismo proposto, além dos problemas que devem ser experimentados pelo algoritmo *3M* quando o número de nós se torna

Tabela V.1: Percentuais de pacotes perdidos com 15 nós.

	Algoritmo		Ganho Percentual
	Três metricas (*3M*)	Métrica única (*SMM*)	%
Link 0-1	13.1	9.0	31
Link 1-5	15.3	11.8	24
Link 5-12	14.9	8.9	40

grande.

Em seguida foram realizados testes com uma topologia menor, com dez nós. A topologia utilizada está representada na figura V.3. Uma fonte de tráfego CBR foi colocada enviando pacotes do nó 9 para o nó 4. A rota utilizada pelo tráfego nos dois algoritmos é $9-3-5-8-4$. A topologia foi testada, então, para o caso em que cada um dos quatro enlaces da rota escolhida entre os dois nós falha. Neste caso, porém, o mesmo teste foi realizado com dois grupos de valores de métricas distintos. No primeiro, as métricas são bastante semelhantes entre si. No outro grupo, estas são bastante diferenciadas. O objetivo deste teste é mostrar o efeito que a quantidade de rotas com métricas iguais tem no desempenho dos dois algoritmos.

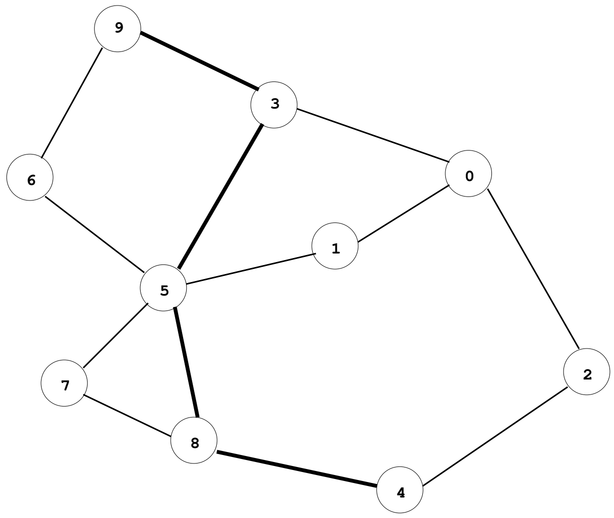


Figura V.3: Topologia de teste com 10 nós.

Os experimentos foram novamente realizados para as duas versões do simulador.

No caso da busca métrica por métrica ($3M$), quando ocorre um empate no melhor valor para a primeira métrica, a decisão passa para a métrica seguinte e para a próxima em caso de novo empate. Sendo assim, pode-se esperar que nas topologias com muitas rotas semelhantes o algoritmo exaustivo possua um desempenho pior.

Tabela V.2: Percentuais de pacotes perdidos com 10 nós: métricas diferentes.

	Algoritmo		Ganho Percentual
	Três métricas ($3M$)	Métrica única (SMM)	%
Link 9-3	10.1	8.6	15
Link 3-5	2.2	2.2	0
Link 5-8	4.1	4.1	0
Link 8-4	9.4	6.4	32

Tabela V.3: Percentuais de pacotes perdidos com 10 nós: métricas semelhantes.

	Algoritmo		Ganho Percentual
	Três métricas ($3M$)	Métrica única (SMM)	%
Link 9-3	7.9	4.9	38
Link 3-5	11.1	8.1	27
Link 5-8	12.0	7.1	41
Link 8-4	16.1	11.2	30

Esta é a conclusão que pode ser retirada da comparação entre as tabelas V.2 e V.3. No caso em que as métricas são muito diferentes entre si, os algoritmos tendem a um desempenho semelhante. Isto porque ocorrem poucos empates entre as rotas nos diversos valores de métricas. No entanto, é importante observar que o desempenho do algoritmo proposto (SMM) é sempre no mínimo igual ao do algoritmo exaustivo ($3M$), sendo o número de passos para que o algoritmo proposto descubra determinada rota sempre menor ou igual que algoritmo $3M$.

É importante notar que o algoritmo SMM gasta o mesmo número de passos na descoberta de rotas que o gasto por um algoritmo considerando duas métricas, como pôde ser concluído do capítulo anterior. Isto se deve à utilização da métrica única combinada. Além disso, quando comparado ao número de passos exigido pelo algoritmo *3M*, o gasto é sempre menor ou igual. Isto se reflete nos resultados obtidos. O desempenho medido para o algoritmo SMM é sempre no mínimo igual ao do algoritmo *3M*. Isto ocorre apenas para topologias pequenas, onde as métricas sejam bastante diferentes. Quando o tamanho da topologia aumenta, o desempenho do algoritmo SMM é sempre superior ao do algoritmo *3M*. Neste caso, o número de rotas possíveis entre uma determinada origem e destino é maior, favorecendo a solução proposta. É importante observar que a vantagem média entre todos os experimentos realizados para o algoritmo SMM foi de 25%.

Capítulo VI

Conclusões

O ROTEAMENTO baseado em requisitos de QoS deve ser considerado como um componente fundamental na provisão de novas classes de serviço na Internet, que possam dar suporte à variedade de novas aplicações multimídias que vêm surgindo. Em se tratando da escalabilidade da arquitetura de roteamento com requisitos de QoS, dois aspectos importantes dizem respeito ao tamanho das tabelas de roteamento utilizadas e à complexidade dos algoritmos para descoberta de rotas.

A solução adotada na arquitetura proposta neste trabalho para as tabelas de roteamento é a manutenção de entradas para cada classe de serviço que seja provida. Sendo assim, de acordo com o tipo do tráfego roteado, a entrada correspondente na tabela é retornada. Com isso o tamanho das tabelas é reduzido, proporcionando consultas mais rápidas.

Neste trabalho, através da combinação de algoritmos apresentados na literatura que consideram duas métricas independentemente, utilizando uma delas como uma métrica única combinada de outras duas, foi mostrado que podem ser construídos algoritmos simples para a construção de rotas dadas três restrições. Foi realizada a proposta de três algoritmos, para roteamento pela origem e roteamento salto-a-salto, neste caso baseado em vetores-distância e no estado dos enlaces. Estes algoritmos levam em consideração como métricas a banda passante, a probabilidade de erro e o atraso de transmissão. Os algoritmos propostos possuem a mesma complexidade dos algoritmos propostos por outros autores para tráfegos com dois requisitos de QoS. Nos algoritmos propostos, uma métrica que possui regra de composição multiplicativa é combinada com uma de composição aditiva, com o auxílio de operações matemáticas simples. As operações adicionadas não alteram o número de passos necessários aos algoritmos para calcular as rotas, preservando sua complexidade computacional. Estes cálculos podem inclusive ser realizados antes da parte iterativa do algoritmo.

Os algoritmos propostos para três restrições garantem, portanto, a escalabilidade da arquitetura de roteamento. Isto se deve ao fato destes algoritmos serem solúveis em tempo polinomial, pois gastam o mesmo número de passos na descoberta de rotas que os algoritmos propostos na literatura que consideram apenas dois requisitos de QoS. A utilização de três requisitos de QoS como métricas para o esquema de roteamento, por outro lado, possui a vantagem de caracterizar bem a maior parte das aplicações multimídias existentes, certamente um conjunto bem maior que o das aplicações que podem ser caracterizadas com apenas dois requisitos de QoS. Os três requisitos de QoS escolhidos neste trabalho são bem representativos para diversas aplicações encontradas hoje em dia.

A arquitetura proposta neste trabalho foi implementada no simulador NS-2. A utilização deste *software* mostrou diversas vantagens e desvantagens. O emprego de um programa que está em desenvolvimento traz alguns problemas quanto à instalação, dependente da plataforma utilizada. Além disso, o programa contém ainda alguns erros de programação, ou incompatibilidades de acordo com a versão das bibliotecas de apoio utilizadas. Desta forma, mesmo antes do desenvolvimento da arquitetura proposta, algumas modificações precisaram ser feitas no código do simulador.

O uso do NS-2 trouxe no entanto muitas vantagens. O simulador implementa uma topologia de rede conceitualmente completa, onde os nós possuem o comportamento de roteadores ou estações. A implementação de diversos tipos de enlaces de transmissão já foi realizada, além de várias fontes de tráfego. A estrutura interna de cada nó é semelhante à da camada de transmissão IP, o que é bastante interessante, pois para a implementação do roteamento com QoS dentro do simulador foram alterados diferentes módulos, de maneira semelhante ao que deveria ser feito numa implementação real. Além disso, no simulador são implementados algoritmos de roteamento semelhantes aos utilizados por protocolos de roteamento como o RIP, DVRP e OSPF.

A implementação serviu também para avaliar o desempenho esperado da arquitetura proposta. O algoritmo baseado em vetores-distância para três métricas proposto foi construído, e comparado à solução semelhante, não escalável, que trata as métricas separadamente. O desempenho obtido pelo mecanismo foi satisfatório. A vantagem média foi de 25%, e em nenhum caso o esquema proposto possui desempenho inferior à versão de controle.

Como principal conclusão deste trabalho, pôde ser observado que a definição de protocolos de roteamento baseados em requisitos de QoS é uma questão importante na definição de uma arquitetura de Serviços Integrados para a Internet. A implementação de protocolos baseados nos algoritmos apresentados neste trabalho mostra-se uma área promissora. A simplicidade do mecanismo proposto é importante, sendo crucial para uma implementação real.

Como trabalhos futuros, podem ser realizadas implementações dos mecanismos propostos para os protocolos de roteamento baseado no estado dos enlaces e de roteamento pela origem. Dentro do NS-2, a avaliação do desempenho como realizada neste trabalho exigiria a implementação de versões dinâmicas destes protocolos. Uma implementação real de protocolos de roteamento também pode ser realizada. O estudo do roteamento com requisitos de QoS considerando a solução IP sobre ATM é uma área interessante relacionada com este tema. Além disso, o estudo de algoritmos para roteamento *multicast* com QoS pode ser feita, além da análise mais detalhada da interface entre o protocolo de roteamento e o RSVP.

Referências Bibliográficas

[1] D. A. M. Villela, L. H. M. K. Costa, C. V. N. de Albuquerque, M. D. Nunes e O. C. M. B. Duarte, "Desempenho de uma aplicação de transferência de arquivos em um sistema de comunicação industrial", in *XI Congresso Brasileiro de Automática*, São Paulo, pp. 1745–1750, setembro 1996.

[2] R. Braden, D. Clark e S. Shenker, *RFC1633: Integrated Services in the Internet Architecture: An Overview.* Network Working Group, junho 1994.

[3] D. Hull, A. Shankar, K. Nahrstedt e J. W. S. Liu, "An end-to-end QoS model and management architecture", in *IEEE Workshop on Middleware for Distributed Real-time Systems and Services*, San Francisco, pp. 82–89, dezembro 1997.

[4] L. Zhang, S. Deering, D. Estrin, S. Shenker e D. Zappala, "RSVP: A new resource reservation protocol", *IEEE Network Magazine*, setembro 1993.

[5] L. Delgrossi, R. G. Herrtwichi, C. Vogt e L. C. Wolf, "Reservation protocols for internetworks: A comparison of ST-II and RSVP", in *Proceedings of Fourth International Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster, UK, novembro 1993.

[6] D. J. Mitzel, D. Estrin, S. Shenker e L. Zhang, "An architectural comparison of ST-II and RSVP", in *Proceedings of IEEE INFOCOM*, 1994.

[7] G. Malkin, *RFC1721: RIP Version 2 Protocol Analysis.* Network Working Group, novembro 1994.

[8] C. Huitema, *Routing in the Internet.* Prentice Hall, 1996.

[9] J. Moy, *RFC1583: OSPF version 2.* Network Working Group, março 1994.

[10] S. Chen e K. Nahrstedt, "On finding multi-constrained paths", in *IEEE International Conference on Communications*, Atlanta, junho 1998.

[11] Q. Ma e P. Steenkiste, "Quality of service routing for traffic with performance guarantees", in *IFIP Fifth International Workshop on Quality of Service*, New York, pp. 115–126, maio 1997.

[12] Z. Wang e J. Crowcroft, "Quality of service routing for supporting multimedia applications", *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, setembro 1996.

[13] F. S. G. Oliveira, A. de Castro Pinto Pedroza e O. C. M. B. Duarte, "Suporte para segurança e mobilidade no IPv6", in *II Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos*, Fortaleza, pp. 233–244, novembro 1997.

[14] B. Rajagopalan e R. Nair, *Quality of Service (QoS)-Based Routing in the Internet - Some Issues.* Bellcore and Ascon Nexen, outubro 1996. Internet-draft.

[15] E. Crawley, R. Nair, B. Rajagopalan e H. Sandick, *A Framework for QoS-Based Routing in the Internet.* Gigapacket Networks, Arrowpoint, NEC USA e IBM, julho 1997. Internet-draft.

[16] J. Moy, *RFC1584: Multicast Extensions to OSPF.* Network Working Group, março 1994.

[17] J. Moy, *RFC1585: MOSPF: Analysis and Experience.* Network Working Group, março 1994.

[18] R. Guerin e A. Orda, "QoS-based routing in networks with innacurate information: Theory and algorithms", in *IEEE INFOCOM'97*, Kobe, abril 1997.

[19] M. Steenstrup, *RFC1477: IDPR as a proposed standard.* Network Working Group, julho 1993.

[20] S. Berson e L. Berger, *IP Integrated Services with RSVP over ATM.* ISI and Fore Systems, março 1997. Internet-draft.

[21] D. J. Mitzel, D. Estrin, S. Shenker e L. Zhang, "A study of reservation dynamics in integrated services packet networks", *IEEE Network Magazine*, pp. 18–26, jan-fev 1995.

[22] P. P. White, "RSVP and integrated services in the internet: A tutorial", *IEEE Communications Magazine*, pp. 100–106, may 1997.

[23] H. G. Perros e K. M. Elsayed, "Call admission control schemes: a review", *IEEE Communications Magazine*, novembro 1996.

[24] M. Laubach, *RFC1577: Classical IP over ATM.* Network Working Group, janeiro 1994.

[25] M. Borden, E. Crawley, B. Davie e S. Batsell, *RFC1821: Integration of Real-time Services in an IP-ATM Network Architecture.* Network Working Group, agosto 1995.

[26] J. Heinanen, *RFC1483: Multiprotocol Encapsulation over ATM Adaptation Layer 5.* Network Working Group, julho 1993.

[27] L. H. M. K. Costa, F. P. Junqueira e O. C. M. B. Duarte, "Uma arquitetura de roteamento com qualidade de serviço baseada em redes ativas", in *II Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos*, Fortaleza, pp. 257–268, novembro 1997.

[28] D. Tennenhouse et al, "A Survey of Active Network Research", *IEEE Communications Magazine*, pp. 80–86, janeiro 1997.

[29] S. Bhattacharjee, K. Calvert e E. W. Zegura, "An Architecture for Active Networking", in *High Performance Networking, White Plains, NY*, abril 1997.

[30] R. Guerin, S. Kamat, A. Orda, T. Przygienda e D. Williams, *QoS Routing Mechanisms and OSPF Extensions.* IBM, Technion and Fore Systems, março 1997. Internet-draft.

[31] D. Zappala, *RSRR: A Routing Interface for RSVP.* USC/ISI, novembro 1996. Internet-draft.

[32] R. Guerin, S. Kamat e S. Herzog, *QoS Path Management with RSVP.* IBM, março 1997. Internet-draft.

[33] E. Allison Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib e L. Zhang, *Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment.* USC/ISI, Cisco Systems, Harvard, UUNET Technologies, Sun Microsystems, Intel Corporation e UCLA, março 1997. Internet-draft.

[34] L. H. M. K. Costa e O. C. M. B. Duarte, "Um mecanismo escalável de roteamento com qualidade de serviço", in *XVI Simpósio Brasileiro de Redes de Computadores*, Rio de Janeiro, p. poster, maio 1998.

[35] L. H. M. K. Costa e O. C. M. B. Duarte, "Um mecanismo de roteamento para aplicações multimídias com três requisitos de qualidade de serviço", in *XVI Seminário Integrado de Hardware e Software*, Belo Horizonte, agosto 1998.

[36] K. Fall e K. Varadhan, *NS notes and documentation.* University of California at Berkeley/Lawrence Berkeley National Laboratory, novembro 1997. Available at http://www-mash.cs.berkeley.edu/ns.

[37] D. Waitzman, C. Partridge e S. Deering, *RFC1075: Distance Vector Multicast Routing Protocol.* Network Working Group, novembro 1988.

Apêndice A

Código do Roteamento com QoS

```
##########################################################################
#                                                                        #
# Primeira versao do Roteamento com QoS                                  #
# Metrica Unica Combinada                                                #
#                                                                        #
##########################################################################

Class rtObject

rtObject set maxpref_   255          ;# quanto menor, maior a preferencia
                                     ;# de cada protocolo

# Limites aceitaveis para as Classes de Servico

rtObject set max_delay   15          ;# atraso maximo, em ms,
                                     ;# 15 experimentalmente

rtObject set min_bw      1           ;# banda passante minima, em mbps,
                                     ;# 1 experimentalmente

rtObject set max_lp 0.3              ;# prob. de perda maxima, sem un.,
                                   ;# 0.3 experimentalmente (menos de 30%)

rtObject set max_unic   15.3         ;# metrica unica maxima
                                   ;# 15 + 0.3 (corres. a perda de 0.5)

# Definicao dos valores das diferentes metricas
# para o caso de rota inalcancavel

rtObject set unreach_delay 32
rtObject set unreach_bw 0
rtObject set unreach_lp 1.0
```

```
rtObject set unreach_unic 32.99

rtObject set contador_  1    ;# variavel que armazena o numero de
                             ;# passos gasto no calculo de rotas

...

# Rotina que escolhe e instala as melhores rotas entre todos os
# protocolos, de acordo com sua preferencia (preference_).

rtObject instproc compute-routes {} {
    # choose the best route to each destination from all protocols
    $self instvar ns_ node_ rtProtos_ nullAgent_
    $self instvar nextHop_ rtpref_ metric_ rtVia_
    $self instvar metric1_ metric2_ metricU_
    $self instvar contador_

    set UNREACH_DELAY [$class set unreach_delay]
    set UNREACH_BW    [$class set unreach_bw]
    set UNREACH_LP    [$class set unreach_lp]
    set UNREACH_UNIC  [$class set unreach_unic]

    set protos ""
    set changes 0

    foreach p [array names rtProtos_] {
      if [$rtProtos_($p) set rtsChanged_] {
incr changes
$rtProtos_($p) set rtsChanged_ 0
      }
      lappend protos $rtProtos_($p)
    }
    if !$changes return

    set changes 0

# aqui produzimos o atraso na atualizacao das rotas
    set t0 [$ns_ now]
    set tempo [expr $t0 + (0.02 * $contador_)]

    foreach dst [$ns_ all-nodes-list] {
      foreach CoS {0 1 2} {
if {$dst == $node_} continue
  ;# no'=destino, passe para o proximo
set nh ""
set pf [$class set maxpref_]
set mt $UNREACH_DELAY
set mt1 $UNREACH_BW
set mt2 $UNREACH_LP
set mtU $UNREACH_UNIC
```

```
set rv ""
set dst_CoS [concat $dst $CoS]     ;# destino = destino + CoS
foreach p $protos {                ;# para cada protocolo
  set pnh [$p set nextHop_($dst_CoS)]   ;#
  if { $pnh == "" } continue   ;# nao havia rota, prox. prot.

  set ppf [$p set rtpref_($dst_CoS)]   ;# pref do protocolo
  set pmt [$p set metric_($dst_CoS)]   ;# metrica 0  do prot
  set pmt1 [$p set metric1_($dst_CoS)] ;# metrica 1 do prot
  set pmt2 [$p set metric2_($dst_CoS)] ;# metrica 2 do prot
  set pmtU [$p set metricU_($dst_CoS)] ;# metrica U do prot

# se rota aceitavel...
  if {$CoS==0} {
    set pmtdummy $pmt
    set mtdummy $mt
  }
  if {$CoS==1} {
    set pmtdummy $pmt1
    set mtdummy $mt1
  }
  if {$CoS==2} {
    set pmtdummy $pmt2
            set mtdummy $mt2
  }

  if {$CoS==0} {  ;# Classe de Servico 0: privilegia o atraso
    if {$ppf < $pf || ($ppf == $pf && $pmtdummy < $mtdummy) \
            || $mtdummy >= $UNREACH_DELAY } {
      set nh  $pnh                    ;# proximo hop
      set pf  $ppf                    ;# pref da rota
      set mt  $pmt                    ;# metrica da rota
      set mt1  $pmt1                  ;# metric1 da rota
      set mt2  $pmt2                  ;# metric2 da rota
      set mtU  $pmtU                  ;# metricU da rota
      set rv  $p                      ;# protocolo
    }
        };# CoS 0

  if {$CoS==1} {      ;# Classe de Servico 1: banda passante
    if {$ppf < $pf || ($ppf == $pf && $pmtdummy > $mtdummy) \
            || $mtdummy <= $UNREACH_BW } {
      set nh  $pnh                    ;# proximo hop
      set pf  $ppf                    ;# pref da rota
      set mt  $pmt                    ;# metrica da rota
      set mt1  $pmt1                  ;# metric1 da rota
      set mt2  $pmt2                  ;# metric2 da rota
      set mtU  $pmtU                  ;# metricU da rota
      set rv  $p                      ;# protocolo
    }
```

```
        };# CoS 1

  if {$CoS==2} {       ;# Classe de Servico 2: prob. de perda
    if {$ppf < $pf || ($ppf == $pf && $pmtdummy < $mtdummy) \
                || $mtdummy >= $UNREACH_LP} {
      set nh  $pnh                   ;# proximo hop
      set pf  $ppf                   ;# pref da rota
      set mt  $pmt                   ;# metrica da rota
      set mt1  $pmt1                 ;# metric1 da rota
      set mt2  $pmt2                 ;# metric2 da rota
      set mtU  $pmtU                 ;# metric2 da rota
      set rv  $p                     ;# protocolo
    }
        };# CoS 2

       } ;# foreach protos

# se rota inexistente... apague as rotas existentes
if { $nh == "" } {
  # no route...  delete any existing routes
  if { $nextHop_($dst_CoS) != "" } {
    #$node_ delete-routes [$dst id] $nextHop_($dst_CoS) $nullAgent_ $CoS
    set nextHop_($dst_CoS)  $nh
    set rtpref_($dst_CoS)   $pf
    set metric_($dst_CoS)   $mt
    set metric1_($dst_CoS)  $mt1
    set metric2_($dst_CoS)  $mt2
    set metricU_($dst_CoS)  $mtU
    set rtVia_($dst_CoS)    $rv
    incr changes
  }
} else {
    if { $rv == $rtVia_($dst_CoS) } {
      # Current protocol still has best route.
      # See if changed
      if { $nh != $nextHop_($dst_CoS) } {
      #$node_ delete-routes [$dst id] $nextHop_($dst_CoS) $nullAgent_ $CoS
      set nextHop_($dst_CoS) $nh
      $ns_ at $tempo "$node_ add-routes [$dst id]  $nextHop_($dst_CoS) $CoS"
      incr changes
      }

# se houve alguma mudanca nas metricas da rota...

      if { ($metric_($dst_CoS) != $mt) || \
      ($metric1_($dst_CoS) != $mt1) \
|| ($metric2_($dst_CoS) != $mt2) } {
        set metric_($dst_CoS) $mt
        set metric1_($dst_CoS) $mt1
        set metric2_($dst_CoS) $mt2
```

```
        set metricU_($dst_CoS) $mtU
        incr changes
      }
      if { $pf != $rtpref_($dst_CoS) } {
        set rtpref_($dst_CoS) $pf
      }
    } else {
if { $rtVia_($dst_CoS) != "" } {
  set nextHop_($dst_CoS) [$rtVia_($dst_CoS) set nextHop_($dst_CoS)]
  set rtpref_($dst_CoS)  [$rtVia_($dst_CoS) set rtpref_($dst_CoS)]
  set metric_($dst_CoS)  [$rtVia_($dst_CoS) set metric_($dst_CoS)]
  set metric1_($dst_CoS) [$rtVia_($dst_CoS) set metric1_($dst_CoS)]
  set metric2_($dst_CoS) [$rtVia_($dst_CoS) set metric2_($dst_CoS)]
  set metricU_($dst_CoS) [$rtVia_($dst_CoS) set metricU_($dst_CoS)]
}

if {$rtpref_($dst_CoS) != $pf || $metric_($dst_CoS) != $mt || \
$metric1_($dst_CoS) != $mt1 || $metric2_($dst_CoS) != $mt2 } {
  # Then new prefs must be better, or
  # new prefs are equal, and new metrics are lower
  #$node_ delete-routes [$dst id] $nextHop_($dst_CoS) $nullAgent_ $CoS
  set nextHop_($dst_CoS) $nh
  set rtpref_($dst_CoS)  $pf
  set metric_($dst_CoS)  $mt
  set metric1_($dst_CoS)  $mt1
  set metric2_($dst_CoS)  $mt2
  set metricU_($dst_CoS)  $mtU
  set rtVia_($dst_CoS)   $rv
  $ns_ at $tempo "$node_ add-routes [$dst id] $nextHop_($dst_CoS) $CoS"
  incr changes
}
    }
}
      };# foreach CoS
    };# foreach dest

    foreach proto [array names rtProtos_] {
$rtProtos_($proto) send-updates $changes
    }

    $self flag-multicast $changes
}

...

proc minimo {m1 m2} {
    if {$m1 <= $m2} {
return $m1
    } else {
return $m2
```

```
    }
}

proc prob {m1 m2} {
    return [expr 1 - ((1-$m1)*(1-$m2)) ]
}

proc unica {m1 m2} {
    set a [expr log10(1 - $m2)]
# log. da prob. de transm. c/ sucesso
    return [expr $m1 - $a]
}

...

############################################################################
#                                                                          #
# Distance Vector Route Computation                                        #
#                                                                          #
############################################################################

Agent/rtProto/DV instproc compute-routes {} {
    $self instvar ns_ ifs_ rtpref_ metric_ nextHop_ nextHopPeer_
    $self instvar peers_ rtsChanged_ multiPath_
    $self instvar metric1_ metric2_ metricU_ node_
    $self instvar rtObject_

    set MAX_DELAY  [rtObject set max_delay]
    set MIN_BW          [rtObject set min_bw]
    set MAX_LP          [rtObject set max_lp]
    set MAX_UNIC         [rtObject set max_unic]

    set MAXPREF  [rtObject set maxpref_]

    set UNREACH_DELAY  [rtObject set unreach_delay]
    set UNREACH_BW  [rtObject set unreach_bw]
    set UNREACH_LP  [rtObject set unreach_lp]
    set UNREACH_UNIC  [rtObject set unreach_unic]
    set rtsChanged_ 0

    foreach dst [$ns_ all-nodes-list] {

    set counter 1      ;# variavel que vai conter o numero de passo
               ;# gasto na descoberta das rotas

#
#   Classe de Servico 0 - Privilegia o Atraso primeiramente
#
      set CoS 0
      if {$CoS==0} {
```

```
catch "unset nh"
      set p [lindex $nextHopPeer_([concat $dst $CoS]) 0]
      if {$p != ""} {
set metric_([concat $dst $CoS])  [$p metric? [concat $dst $CoS]]
set metric1_([concat $dst $CoS]) [$p metric1? [concat $dst $CoS]]
set metric2_([concat $dst $CoS]) [$p metric2? [concat $dst $CoS]]
set metricU_([concat $dst $CoS]) [$p metricU? [concat $dst $CoS]]
set rtpref_([concat $dst $CoS])  [$p preference? [concat $dst $CoS]]
      }

      set pf  $MAXPREF
      set mt  $UNREACH_DELAY
      set mt1 $UNREACH_BW
      set mt2 $UNREACH_LP
      set mtU $UNREACH_UNIC
      set nh(0) 0

      foreach nbr [array names peers_] {
set pmt  [$peers_($nbr) metric? [concat $dst $CoS]]
set pmt1 [$peers_($nbr) metric1? [concat $dst $CoS]]
set pmt2 [$peers_($nbr) metric2? [concat $dst $CoS]]
set pmtU [$peers_($nbr) metricU? [concat $dst $CoS]]
set ppf [$peers_($nbr) preference? [concat $dst $CoS]]

# if peer metric not valid continue
# if peer pref higher continue
# if peer pref lower set to latest values
# else peer pref equal
# if peer metric higher continue
# if peer metric lower set to latest values
#  else peer metrics equal append latest values

          set pmtdummy $pmt
          set mtdummy  $mt

   if { $pmtU < 0 || $pmtU >= $UNREACH_UNIC || $pmt1 < $MIN_BW || \
$ppf > $pf || $pmtU > $mtU || $pmt2 > $MAX_LP } {
  continue
        }
   if { $ppf < $pf || $pmtU < $mtU } { ;# metrica do peer melhor
  set pf $ppf
  set mtdummy $pmtdummy
  set mt $pmt
  set mt1 $pmt1
          set mt2 $pmt2
          set mtU $pmtU
          unset nh    ;# because we must compute *new* next hops
}                  ;# senao, metrica igual
set nh($ifs_($nbr)) $peers_($nbr)
set counter [expr $counter + 1] ;# incremento do numero de passos
```

```
        }    ;#foreach nbr

  $rtObject_ set contador_ $counter
        set metric_dummy $metric_([concat $dst $CoS])

        catch "unset nh(0)"

        if { $pf == $MAXPREF && $mtdummy == $UNREACH_DELAY } continue
          if { $pf > $rtpref_([concat $dst $CoS]) || \
      ($metric_dummy >= 0 && $mtdummy >          \
      $metric_dummy ) }                      \
    continue
          if {$mtdummy >= $MAX_DELAY} {
            set mtdummy $UNREACH_DELAY
    set mt $UNREACH_DELAY
    set mt1 $UNREACH_BW
    set mt2 $UNREACH_LP
    set mtU $UNREACH_UNIC
          }

        incr rtsChanged_
        if {$pf < $rtpref_([concat $dst $CoS]) ||                \
$mtdummy < $metric_([concat $dst $CoS]) } {
    set rtpref_([concat $dst $CoS]) $pf
    set metric_([concat $dst $CoS]) $mt
    set metric1_([concat $dst $CoS]) $mt1
    set metric2_([concat $dst $CoS]) $mt2
    set metricU_([concat $dst $CoS]) $mtU
    set nextHop_([concat $dst $CoS]) ""
    set nextHopPeer_([concat $dst $CoS]) ""
    foreach n [array names nh] {
lappend nextHop_([concat $dst $CoS]) $n
lappend nextHopPeer_([concat $dst $CoS]) $nh($n)
if !$multiPath_ break;
    }
    continue
        }

        set rtpref_([concat $dst $CoS]) $pf
        set metric_([concat $dst $CoS]) $mt
        set metric1_([concat $dst $CoS]) $mt1
        set metric2_([concat $dst $CoS]) $mt2
        set metricU_([concat $dst $CoS]) $mtU
        set newNextHop ""
        set newNextHopPeer ""
        foreach rt $nextHop_([concat $dst $CoS]) {
  if [info exists nh($rt)] {
    lappend newNextHop $rt
    lappend newNextHopPeer $nh($rt)
    unset nh($rt)
```

```
  }
        }

        set nextHop_([concat $dst $CoS]) $newNextHop
        set nextHopPeer_([concat $dst $CoS]) $newNextHopPeer
        if { $multiPath_ || $nextHop_([concat $dst $CoS]) == "" } {
  foreach rt [array names nh] {
    lappend nextHop_([concat $dst $CoS]) $rt
    lappend nextHopPeer_([concat $dst $CoS]) $nh($rt)
    if !$multiPath_ break
          }
        }
    };# fim de CoS 0

#
#   Classe de Servico 1 - Privilegia a banda passante
#
      set CoS 1
      if {$CoS==1} {
catch "unset nh"
        set p [lindex $nextHopPeer_([concat $dst $CoS]) 0]
        if {$p != ""} {
  set metric_([concat $dst $CoS])  [$p metric? [concat $dst $CoS]]
  set metric1_([concat $dst $CoS]) [$p metric1? [concat $dst $CoS]]
  set metric2_([concat $dst $CoS]) [$p metric2? [concat $dst $CoS]]
  set metricU_([concat $dst $CoS]) [$p metricU? [concat $dst $CoS]]
  set rtpref_([concat $dst $CoS])  [$p preference? [concat $dst $CoS]]
        }

        set pf $MAXPREF
        set mt  $UNREACH_DELAY
        set mt1 $UNREACH_BW
        set mt2 $UNREACH_LP
        set mtU $UNREACH_UNIC
        set nh(0) 0

        foreach nbr [array names peers_] {
  set pmt  [$peers_($nbr) metric? [concat $dst $CoS]]
  set pmt1 [$peers_($nbr) metric1? [concat $dst $CoS]]
  set pmt2 [$peers_($nbr) metric2? [concat $dst $CoS]]
  set pmtU [$peers_($nbr) metricU? [concat $dst $CoS]]
  set ppf [$peers_($nbr) preference? [concat $dst $CoS]]

  # if peer metric not valid continue
  # if peer pref higher continue
  # if peer pref lower set to latest values
  # else peer pref equal
  # if peer metric higher continue
  # if peer metric lower set to latest values
  #  else peer metrics equal append latest values
```

```
          set pmtdummy $pmt1
          set mtdummy  $mt1

  if { $pmtdummy == 0 || $pmtdummy <= $MIN_BW || \
  $ppf > $pf || $pmtdummy < $mtdummy } \
    continue
  if { $ppf < $pf || $pmtdummy > $mtdummy } {
#quanto mais banda melhor
    set pf $ppf
    set mtdummy $pmtdummy
    set mt $pmt
    set mt1 $pmt1
            set mt2 $pmt2
            set mtU $pmtU
            unset nh    ;# because we must compute *new* next hops
  }
  set nh($ifs_($nbr)) $peers_($nbr)
  set counter [expr $counter + 1]
# incremento do numero de passos
        }    ;#foreach nbr

# Buscamos agora pela segunda metrica - unica combinada
           catch "unset nh(0)"
  catch "unset nh_iguais"
    set met_min 32.99
  set nh_iguais(0) 0
    foreach viz [array names nh] { ;# unica
        set met_atual [$nh($viz) metricU? [concat $dst $CoS]]
      if { $met_atual > $met_min } {
  catch "unset nh($viz)"
      }
      if { $met_atual == $met_min } {
  set nh_iguais($viz) 1
  set mt2 [$nh($viz) metric2? [concat $dst $CoS]]
  set mt  [$nh($viz) metric?  [concat $dst $CoS]]
      }
        if { $met_atual < $met_min } {  ;# quanto menor, melhor
    set met_min $met_atual
            set mt2 [$nh($viz) metric2? [concat $dst $CoS]]
  set mt  [$nh($viz) metric? [concat $dst $CoS]]
          foreach lui [array names nh_iguais] {
      catch "unset nh($lui)"
  }
  set nh_iguais($viz) 2
        }
        set nbr_ant $viz
        set counter [expr $counter + 1] ;# incremento do numero de passos
    }
    set mtU $met_min
```

```
  $rtObject_ set contador_ $counter
          set metric_dummy $metric1_([concat $dst $CoS])

         catch "unset nh(0)"

          if { $pf == $MAXPREF && $mtdummy == $UNREACH_BW } continue
          if { $pf > $rtpref_([concat $dst $CoS]) || \
      ($metric_dummy > $MIN_BW && $mtdummy < \
      $metric_dummy ) }                     \
    continue
          if {$mtdummy <= $MIN_BW} {
            set mtdummy $UNREACH_BW
    set mt  $UNREACH_DELAY
    set mt1 $UNREACH_BW
    set mt2 $UNREACH_LP
    set mtU $UNREACH_UNIC
          }

        incr rtsChanged_
        if {$pf < $rtpref_([concat $dst $CoS]) || \
    $mtdummy > $metric1_([concat $dst $CoS]) } {
         set rtpref_([concat $dst $CoS]) $pf
          set metric_([concat $dst $CoS]) $mt
          set metric1_([concat $dst $CoS]) $mt1
          set metric2_([concat $dst $CoS]) $mt2
          set metricU_([concat $dst $CoS]) $mtU
  set nextHop_([concat $dst $CoS]) ""
  set nextHopPeer_([concat $dst $CoS]) ""
  foreach n [array names nh] {
    lappend nextHop_([concat $dst $CoS]) $n
    lappend nextHopPeer_([concat $dst $CoS]) $nh($n)
    if !$multiPath_ break;
  }
continue
        }

        set rtpref_([concat $dst $CoS]) $pf
        set metric_([concat $dst $CoS]) $mt
        set metric1_([concat $dst $CoS]) $mt1
        set metric2_([concat $dst $CoS]) $mt2
        set metricU_([concat $dst $CoS]) $mtU
        set newNextHop ""
        set newNextHopPeer ""
        foreach rt $nextHop_([concat $dst $CoS]) {
  if [info exists nh($rt)] {
    lappend newNextHop $rt
    lappend newNextHopPeer $nh($rt)
    unset nh($rt)
  }
```

```
        }

        set nextHop_([concat $dst $CoS]) $newNextHop
        set nextHopPeer_([concat $dst $CoS]) $newNextHopPeer
        if { $multiPath_ || $nextHop_([concat $dst $CoS]) == "" } {
  foreach rt [array names nh] {
    lappend nextHop_([concat $dst $CoS]) $rt
    lappend nextHopPeer_([concat $dst $CoS]) $nh($rt)
    if !$multiPath_ break
          }
        }
      }      ;# fim de CoS 1

#
#   Classe de Servico 2
#
      set CoS 2
      if {$CoS==2} {
catch "unset nh"
        set p [lindex $nextHopPeer_([concat $dst $CoS]) 0]
        if {$p != ""} {
  set metric_([concat $dst $CoS])  [$p metric? [concat $dst $CoS]]
  set metric1_([concat $dst $CoS]) [$p metric1? [concat $dst $CoS]]
  set metric2_([concat $dst $CoS]) [$p metric2? [concat $dst $CoS]]
  set metricU_([concat $dst $CoS]) [$p metricU? [concat $dst $CoS]]
  set rtpref_([concat $dst $CoS])  [$p preference? [concat $dst $CoS]]
        }

        set pf $MAXPREF
        set mt  $UNREACH_DELAY
        set mt1 $UNREACH_BW
        set mt2 $UNREACH_LP
        set mtU $UNREACH_UNIC
        set nh(0) 0

        foreach nbr [array names peers_] {
  set pmt  [$peers_($nbr) metric? [concat $dst $CoS]]
  set pmt1 [$peers_($nbr) metric1? [concat $dst $CoS]]
  set pmt2 [$peers_($nbr) metric2? [concat $dst $CoS]]
  set pmtU [$peers_($nbr) metricU? [concat $dst $CoS]]
  set ppf [$peers_($nbr) preference? [concat $dst $CoS]]

  # if peer metric not valid continue
  # if peer pref higher continue
  # if peer pref lower set to latest values
  # else peer pref equal
  # if peer metric higher continue
  # if peer metric lower set to latest values
  #  else peer metrics equal append latest values
```

```
        set pmtdummy $pmt2
        set mtdummy  $mt2

     if { $pmtU < 0 || $pmtU >= $UNREACH_UNIC || $pmt1 < $MIN_BW || \
  $ppf > $pf || $pmtU > $mtU || $pmt2 > $MAX_LP } \
    continue
  if { $ppf < $pf || $pmtU < $mtU } {
    set pf $ppf
    set mtdummy $pmtdummy
    set mt $pmt
    set mt1 $pmt1
            set mt2 $pmt2
            set mtU $pmtU
            unset nh    ;# because we must compute *new* next hops
  }
  set nh($ifs_($nbr)) $peers_($nbr)
  set counter [expr $counter + 1]     ;# incremento do numero de passos
      }    ;#foreach nbr

$rtObject_ set contador_ $counter
set metric_dummy $metric2_([concat $dst $CoS])

        catch "unset nh(0)"       ;# adiantamos este unset

         if { $pf == $MAXPREF && $mtdummy == $UNREACH_LP } continue
         if { $pf > $rtpref_([concat $dst $CoS]) || \
    ($metric_dummy < 1 && $mtdummy >         \
    $metric_dummy ) }                      \
    continue
         if {$mtdummy >= $MAX_LP} {
           set mtdummy $UNREACH_LP
    set mt $UNREACH_DELAY
    set mt1 $UNREACH_BW
    set mt2 $UNREACH_LP
    set mtU $UNREACH_UNIC
         }

       incr rtsChanged_
       if {$pf < $rtpref_([concat $dst $CoS]) ||               \
    $mtdummy < $metric2_([concat $dst $CoS]) } {
  set rtpref_([concat $dst $CoS]) $pf
         set metric_([concat $dst $CoS]) $mt
         set metric1_([concat $dst $CoS]) $mt1
         set metric2_([concat $dst $CoS]) $mt2
         set metricU_([concat $dst $CoS]) $mtU
  set nextHop_([concat $dst $CoS]) ""
  set nextHopPeer_([concat $dst $CoS]) ""
  foreach n [array names nh] {
    lappend nextHop_([concat $dst $CoS]) $n
    lappend nextHopPeer_([concat $dst $CoS]) $nh($n)
```

```
    if !$multiPath_ break;
  }
continue
        }

        set rtpref_([concat $dst $CoS]) $pf
        set metric_([concat $dst $CoS]) $mt
        set metric1_([concat $dst $CoS]) $mt1
        set metric2_([concat $dst $CoS]) $mt2
        set metricU_([concat $dst $CoS]) $mtU
        set newNextHop ""
        set newNextHopPeer ""
        foreach rt $nextHop_([concat $dst $CoS]) {
  if [info exists nh($rt)] {
    lappend newNextHop $rt
    lappend newNextHopPeer $nh($rt)
    unset nh($rt)
  }
        }

        set nextHop_([concat $dst $CoS]) $newNextHop
        set nextHopPeer_([concat $dst $CoS]) $newNextHopPeer
        if { $multiPath_ || $nextHop_([concat $dst $CoS]) == "" } {
  foreach rt [array names nh] {
    lappend nextHop_([concat $dst $CoS]) $rt
    lappend nextHopPeer_([concat $dst $CoS]) $nh($rt)
    if !$multiPath_ break
          }
        }
      }      ;# fim de CoS 2

    };# foreach dst
    set rtsChanged_
}
```