AGREGAÇÃO DE ESTADOS DO PROTOCOLO RSVP PARA FLUXOS *MULTICAST*

Paulo Cesar Salgado Vidal

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 DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

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

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Asterio Kiyoshi Tanaka, Ph.D.

Prof. Jauvane Cavalcante de Oliveira, Ph.D.

Prof. Marcelo Gonçalves Rubinstein, D.Sc.

Prof. Paulo César Coelho Ferreira, Dr.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2003

VIDAL, PAULO CESAR SALGADO

Agregação de Estados do Protocolo RSVP [Rio de Janeiro] 2003

XIV, 107 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2003)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Qualidade de Serviço
2. Protocolos de Reserva de Recursos
3. Escalabilidade

I. COPPE/UFRJ II. Título (Série)

A Margareth, minha esposa e amiga de todas as horas.

A Bianca, Nathália e Marianna, minha filhas.

Agradecimentos

A Deus por me dar a oportunidade da vida e saúde para realizar este trabalho.

A Margareth, minha esposa por seu amor, paciência e apoio durante toda a tese.

Ao Prof. Otto Carlos Muniz Bandeira Duarte pela amizade, orientação e incentivo na realização deste trabalho.

Aos Professores Aloysio de Castro Pinto Pedroza e Asterio Kiyoshi Tanaka pela amizade, incentivo e pela participação no Exame de Qualificação e na Banca de Defesa de Tese. Ao Prof. Luiz Fernando Rust da Costa Carmo pela participação no Exame de Qualificação. Aos Professores Jauvane Cavalcante de Oliveira, Marcelo Gonçalves Rubinstein e Paulo César Coelho Ferreira pela amizade e participação na Banca de Defesa de Tese.

Aos amigos Neves, Silva Neto, Cecílio, Bergman, Ishikawa, Luiz Henrique, Gustavo e Cardoso, pelo estímulo e apoio ao meu trabalho.

Aos professores do Departamento de Engenharia de Sistemas do IME pela amizade, apoio e incentivo.

Ao PEE/COPPE, pelas instalações e equipamentos utilizados.

Aos Profs. Rezende, Richard e Leão; ao Marcelo Rubi, Marcos, Fred, Sílvia, Luiz Henrique, Renata, Flávio, Daniel, Marcelo Amorim, Belém, Alexandre, Marcial, Antônio, Emerson, Márcio, Jonny, Artur, Paulo André, Aline, Kleber, Granato, Fagundes, Eric, Pedro Velloso, Valentim, Gardel, David, Bagatelli, Bia e Solange pela amizade e apoio desde 1997, início da minha jornada.

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

AGREGAÇÃO DE ESTADOS DO PROTOCOLO RSVP PARA FLUXOS *MULTICAST*

Paulo Cesar Salgado Vidal

Setembro/2003

Orientador: Otto Carlos Muniz Bandeira Duarte

Programa: Engenharia Elétrica

A Arquitetura de Serviços Integrados da *Internet* (*IntServ*) tem um problema fundamental de escalabilidade devido ao armazenamento e processamento de estados de cada fluxo em todos os roteadores e sistemas finais que fornecem suporte ao fluxo. Reduzir o número de estados do protocolo RSVP armazenados nos roteadores é uma importante questão para tornar escalável a arquitetura *IntServ*. Este trabalho apresenta e avalia um mecanismo de agregação de estados do protocolo RSVP para fluxos *multicast* utilizando túneis IP-sobre-IP. Foram realizadas simulações para avaliar o mecanismo quanto ao nível de agregação e os resultados mostram uma significativa diminuição dos estados. Entretanto, o mecanismo apresenta uma longa latência no estabelecimento de reservas de recursos, por causa da perda de mensagens RSVP, prejudicando o desempenho de aplicações multimídias. Este trabalho também apresenta o uso de temporizadores de renovação em etapas para acelerar o procedimento de configuração de reservas. O mecanismo foi avaliado na presença de perdas de mensagens RSVP túnel e dois modelos de perdas foram utilizados nas simulações: taxa média de perda de pacotes e perdas periódicas em situações de congestionamento. As medidas obtidas através de simulações mostram uma significativa diminuição no tempo de estabelecimento de sessões *multicast* com o uso de temporizadores de renovação em etapas.

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

AGGREGATION OF THE STATES IN THE RSVP PROTOCOL FOR THE MULTICAST FLOWS

Paulo Cesar Salgado Vidal

September/2003

Advisor: Otto Carlos Muniz Bandeira Duarte

Department: Electrical Engineering

The Internet Integrated Services Architecture (IntServ) has a fundamental scaling problem due to the storage and processing of states for every flow at all routers and end-systems supporting a flow. Reducing the number of RSVP protocol states stored in routers is an important issue in order to scale the IntServ architecture. This work presents and evaluates a state aggregation mechanism for the RSVP protocol for multicast flows using IP in IP tunneling. The simulation results show a significant decrease of the number of states. Because of the loss of tunnel RSVP messages, the mechanism presents a very long latency at reservation establishment. This problem reduces the performance of multimedia applications. This work also presents the use of staged refresh timers in the mechanism to speed up the set-up procedure of the resource reservation. The mechanism was evaluated in the presence of tunnel RSVP message losses. Two loss models were used: average packet loss rate and periodic losses in congestion situations. The measurements obtained through of simulations show a significant decrease in the time of establishment of multicast sessions with the use of the staged refresh timers.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ATM : *Asynchronous Transfer Mode*;

BGP : *Border Gateway Protocol*;

CBQ : *Class Based Queue*;

CBR : *Constant Bit Rate*;

CBT : *Core Based Trees*;

DS : *Differentiated Services*;

GGSN : *Gateway GPRS Support Node*;

GPRS : *General Packet Radio Service*;

IANA : *Internet Assigned Numbers Authorithy*;

IETF : *Internet Engineering Task Force*;

IP : *Internet Protocol*;

LDP : *Label Distribution Protocol*;

LSP : *Label-Switched Path*;

LSR : *Label-Switching Router*;

MTU : *Maximum Transfer Unit*;

MPLS : *Multiprotocol Label Switching Architecture*;

ns : *network simulator*;

OSI : *Open Systems Interconnection*;

PIM : *Protocol Independent Multicast*;

PSB : *Path State Block*;

QoS : *Quality of Service*;

RTP : *Real Time Protocol*;

RTCP : *Real Time Control Protocol*;

RSB : *Reservation State Block*;

RSVP : *Resource ReSerVation Protocol*;

SGSN : *Serving GPRS Support Node*;

SMART : *Scalable Multipath Framework Architecture*;

ST-II : *Internet Stream Protocol Version 2*;

TCSB : *Traffic Control State Block*;

TCP : *Transmission Control Protocol*;

TOS : *Type Of Service*;

UDP : *User Datagram Protocol*;

UML : *Unified Modeling Language*;

UMTS : *Universal Mobile Telecommunications System*;

VINT : *Virtual InterNet Testbed*;

YESSIR : *YEt another Sender Session Internet Reservation.*

Capítulo 1

Introdução

Muitas aplicações multimídias de tempo real estão sendo desenvolvidas para *Internet*, onde o modelo de encaminhamento de melhor esforço (*best-effort*) do protocolo IP (*Internet Protocol*) é inadequado. O protocolo IP fornece um serviço na camada de rede sem conexão e sem confiabilidade, e não fornece nenhuma garantia contra atrasos ou perdas de pacotes. Portanto, existe a necessidade de fornecer às aplicações multimídias, classes de serviços com melhor qualidade (QoS - *Quality of Service*) em relação à largura de banda, atraso nas filas e perdas de pacotes. Buscando atingir estes objetivos, o IETF (*Internet Engineering Task Force*) definiu uma arquitetura de serviços integrados da *Internet* (*Integrated Services Architecture - IntServ*) [1], onde são definidas classes de serviços que, se suportadas pelos roteadores, podem fornecer um fluxo de dados com garantias de QoS. O nível de QoS fornecido por estas classes é programável por fluxo, de acordo com os requisitos das aplicações. Os requisitos da aplicação são parâmetros que compõem a especificação do fluxo (*Flow Specification - FlowSpec*) como a taxa de transmissão máxima, a taxa de transmissão média e a MTU (*Maximum Transmission Unit*) [2]. Estes requisitos são passados para os roteadores, usando o protocolo RSVP (*Resource reServation Protocol*) [3, 4].

Um fluxo é identificado pelo endereço *unicast* ou *multicast* e seu estado é armazenado nos roteadores entre a fonte e o destino do fluxo de dados. A arquitetura de serviços integrados (Figura 1.1) possui quatro tipos de estados:

- os estados do escalonador que consiste de filas de serviço de diferentes tipos de tráfego para encaminhar o pacote;
- os estados do classificador que é usado para atribuir, na chegada do pacote, uma fila no escalonador;
- os estados para os pedidos de reserva de recursos do protocolo RSVP;
- os estados do protocolo de roteamento.

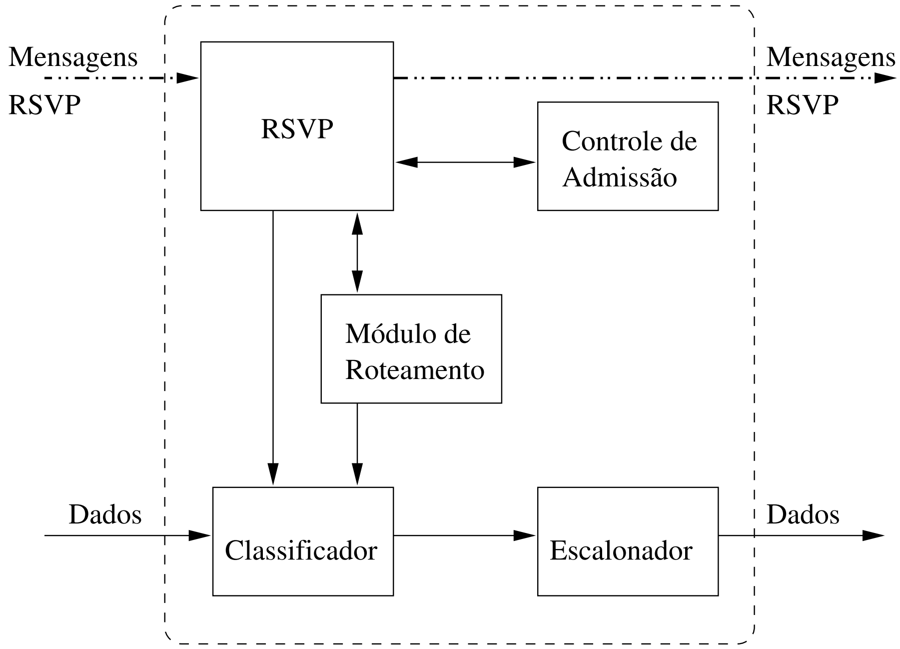

Figura 1.1: Módulos da arquitetura de serviços integrados no roteador.

Durante o estabelecimento de uma reserva, o módulo de controle de admissão é consultado para verificação da existência de recursos disponíveis para o fluxo.

O protocolo RSVP mantém atualizados os estados de reserva através do envio periódico de mensagens de renovação (*refresh*) entre a fonte e o destino, processo chamado de mecanismo *soft-state*. O intervalo de tempo padrão de envio das mensagens é de 30 segundos. Se após um certo período de tempo um roteador não receber, por exemplo, uma mensagem *Resv* (*Reservation*) de determinado destino, ele libera os recursos que haviam sido reservados para aquele fluxo e remove o registro da reserva deste fluxo da tabela de estados.

O processamento por fluxo na arquitetura de serviços integrados apresenta problemas de escalabilidade, pois o número de estados aumenta linearmente com o aumento do número de sessões RSVP ativas no roteador.

Primeiro, os roteadores do *backbone* de uma rede podem processar milhares de fluxos. Isto significa que o roteador pode ser sobrecarregado, porque precisa armazenar uma grande quantidade de estados.

Segundo, para manter os estados através do envio e recebimento de mensagens de renovação, o roteador necessita de grande capacidade de processamento, pois aumenta o volume de mensagens de renovação à medida que aumenta o número de sessões ativas. Além disso, o mecanismo *soft-state* não fornece confiabilidade na transmissão de mensagens. A perda de mensagens de controle em períodos de congestionamento acarreta o atraso na configuração de reservas e o atraso na liberação de recursos.

Pesquisas têm sido realizadas, apresentando soluções para os problemas de escalabilidade, podendo ser divididas em duas áreas:

- agregação de fluxos para reduzir a quantidade de estados armazenados nos roteadores [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];
- redução no volume de mensagens de renovação e no tempo de estabelecimento de reservas de recursos [16, 17, 18, 19].

1.1 Agregação de Fluxos

O trabalho de Terzis et al. [5, 6] utiliza o protocolo RSVP em uma rede que implementa túneis IP-sobre-IP, onde várias sessões RSVP ponto-a-ponto (*unicast*) podem ser mapeadas para uma sessão túnel entre os dois pontos da extremidade do túnel (roteador de entrada e roteador de saída). A redução de estados é alcançada porque os roteadores interiores da região não processam as mensagens RSVP fim-a-fim, somente processam as mensagens RSVP túneis que transportam informações agregadas dos fluxos fim-a-fim. Assim, os roteadores interiores somente armazenam estados das sessões RSVP túneis.

Schelén e Pink propõem [7] uma arquitetura para reservas de recursos baseada em agentes, onde as aplicações ponto-a-ponto fazem pedidos de reserva para um agente responsável pelo controle de admissão das reservas no seu domínio. Os pedidos de reserva contêm a largura de banda a ser reservada, o endereço da fonte do fluxo e o endereço do receptor. Os pedidos podem ser do tipo reserva imediata e reserva antecipada. O pedido de reserva antecipada inclui o tempo de início e a duração da reserva [20]. O agente reúne vários pedidos de diferentes endereços fontes em um grupo que possui o mesmo prefixo do endereço de destino (mesmo domínio). Assim, o agente armazena um estado contendo o endereço destino da rede e a soma da largura de banda de todos os receptores.

A arquitetura proposta por Berson e Vincent [8] mantém constante o número de estados do escalonador, em todos os roteadores da região de agregação, e do classificador nos roteadores interiores devido ao número fixo de classes de tráfego ponto-a-ponto configuradas na região de agregação. Não existem estados do protocolo RSVP armazenados nos roteadores interiores para fluxos ponto-a-ponto. Assim, o controle de admissão aceita, ou recusa, um fluxo baseado em estimativas de tráfego no roteador e na taxa de pico especificada na mensagem *Admission Request* que percorre o caminho entre o roteador de entrada e o roteador de saída da região, solicitando a cada roteador reserva de recursos. Este tipo de controle de admissão é chamado de controle de admissão baseado em medidas [21, 22]. Os estados do tráfego *multicast* são armazenados em todos os nós.

Diferente das propostas citadas quanto a finalidade do uso da agregação, Ramanathan e Dovrolis [9] propõem um mecanismo de tolerância a falhas para arquitetura de serviços integrados. A idéia básica é configurar um caminho secundário com reservas de recursos que serve como uma alternativa no caso de falha no caminho principal do fluxo. As reservas de recursos do caminho secundário são agregadas para reduzir a sobrecarga do mecanismo.

Fukuda et al. [10] apresentam um algoritmo de agregação de fluxos para transporte de vídeo *multicast*. O algoritmo combina requisitos similares de qualidade de serviço dos usuários em um único requisito de QoS. O algoritmo é executado a partir dos nós folhas (receptores) da árvore *multicast* até o nó raiz (servidor de vídeo). Cada roteador recebe como parâmetro de QoS, o intervalo entre a largura de banda mínima e a largura de banda máxima de cada receptor. O roteador combina os requisitos, fazendo uma interseção entre os intervalos para alcançar a agregação. Após executar o algoritmo, o roteador envia o requisito de qualidade de serviço resultante para o próximo roteador, até chegar no servidor de vídeo. Assim, o número necessário de fluxos que o servidor de vídeo necessita enviar e a largura de banda total consumida pela rede são reduzidos pelo compartilhamento de recursos por diversos usuários.

O trabalho de Schmitt et al. [11] realiza uma análise matemática sobre a agregação em fluxos homogêneos da classe de serviço garantido, isto é, fluxos que possuem características de tráfego e requisitos de atraso iguais e que percorrem a mesma rota até o destino. Fluxos de áudio para serviços de telefonia na *Internet* podem ser enquadrados nesta categoria. O serviço garantido fornece um limite superior no atraso fim-a-fim para pacotes de um fluxo pertencente a esta classe. A provisão de tal garantia determinística aumenta o custo da largura de banda para garantir um maior limite de atraso, e, algumas vezes, uma taxa de serviço mais alta que a taxa de pico do fluxo [23, 24]. A análise matemática da agregação de fluxos homogêneos mostrou que a quantidade de largura de banda necessária para reservar um grupo de fluxos pode ser menor que a soma das reservas individuais, fornecendo a mesma garantia de serviço. Este tipo de agregação foi também abordado por Rampal e Guérin [25] com um estudo de caso simples de telefonia IP, diferente da análise mais complexa de Schmitt et al. [11].

O trabalho de Pagani e Magalhães [12] apresenta uma arquitetura de comutação para integrar os mecanismos da arquitetura de serviços integrados com a tecnologia ATM (*Asyncronous Transfer Mode*) chamada Comutação IS (*IntServ*). A Comutação IS estabelece políticas de agregação de fluxos para redução do uso de canais virtuais devido à escassez de rótulos. As políticas de agregação são por domínios IP ou por prefixos de endereços agregados CIDR (*Classless Inter-Domain Routing*) [26] para associar um maior número de fluxos a um único canal virtual.

Baker et al. [13] descrevem um mecanismo que utiliza uma única reserva RSVP para agregar um conjunto de reservas individuais em uma região de agregação, de modo similar ao uso de caminhos virtuais em uma rede ATM. O mecanismo cria dinamicamente uma reserva agregada, classifica o tráfego para qual o agregado se aplica, e determina a largura de banda necessária para o agregado. Uma sessão RSVP agregada é identificada pelo endereço IP destino e pela informação do campo DS (*Differentiated Services*) [27]. Os pacotes de dados dos fluxos agregados são marcados pelo campo DS no cabeçalho IP. Múltiplos agregados entre um roteador de entrada e o roteador de saída são possíveis pelo uso de diferentes valores do campo DS. Os roteadores interiores somente armazenam estados das sessões RSVP agregadas.

Fisher et al. [14] apresentam o projeto e a implementação do protocolo de sinalização *Beagle* para fornecer suporte ao compartilhamento temporal de recursos. O protocolo fornece o compartilhamento de um mesmo conjunto de recursos para fluxos relacionados sobre o tempo lidando com ganhos significativos de recursos e diminuição no número de estados da reserva. Compartilhamento temporal foi primeiro introduzido no protocolo RSVP através de estilos de reservas, permitindo que diferentes emissores de um grupo *multicast* compartilhem um mesmo conjunto de recursos. No protocolo *Beagle*, o compartilhamento é mais amplo e se aplica para fluxos *unicast* e *multicast*.

Vutukury e Garcia-Luna-Aceves [15] propõem a arquitetura SMART (*Scalable Multipath Framework Architecture*) que agrega fluxos ao longo de multicaminhos (*multipaths*) na qual o número de estados de reservas de recursos é baseado no número de destinos e classes de fluxos. A idéia da arquitetura SMART é usar multicaminhos fixos para cada destino. Um multicaminho é um grafo direcionado acíclico tendo como destino o nó sumidouro. Os fluxos de um particular destino são sempre estabelecidos ao longo de grafos de roteamento correspondentes e são reunidos (*merged*) com os outros fluxos que compartilham o mesmo multicaminho.

1.2 Redução de Mensagens de Renovação

Uma forma de reduzir a sobrecarga no processamento do roteador é substituir as mensagens de renovação por uma única mensagem chamada mensagem *Digest* [16]. A mensagem *Digest* contém um conjunto de assinaturas MD5 [28] que representam uma versão comprimida dos estados RSVP compartilhados entre dois roteadores RSVP vizinhos. Quando um roteador RSVP recebe uma mensagem *Digest* de um roteador vizinho, ele compara o valor conduzido na mensagem com o valor resultante do processamento do algoritmo MD5 em seus estados. Se os valores forem iguais, o roteador renova os estados individuais, senão inicia um processo de sincronização de estados para reparar as inconsistências.

Uma segunda proposta, é implementar o conceito de *soft-state* por nó vizinho [17], isto é, os nós RSVP vizinhos enviam periodicamente mensagens chamadas *heartbeats*. Estas mensagens são enviadas após o estabelecimento das reservas de um conjunto de fluxos e substituem as mensagens RSVP no período de estabilidade das reservas. Após a ausência de um número consecutivo de mensagens *heartbeats*, o roteador interpreta como falha na rede e envia mensagens RSVP para atualizar os estados. Quando ocorre mudança ou remoção de reserva de um fluxo, são enviadas as mensagens normais do protocolo RSVP. Depois de um período de estabilização, as mensagens *heartbeats* são enviadas novamente. O benefício da utilização do modelo de estado *soft-state* por nó vizinho em relação ao modelo de estado *soft-state* por fluxo, é que no modelo de estado *soft-state* por nó vizinho as mensagens são geradas a uma taxa fixa, independente do número de reservas estabelecidas.

Para fornecer um encaminhamento de mensagens de controle RSVP com maior rapidez e maior confiabilidade, Pan e Schulzrinne [18] apresentaram o mecanismo de temporizadores de renovação em etapas (*staged refresh timers*) no protocolo RSVP. Temporizadores de renovação em etapas controlam o intervalo de retransmissão das mensagens RSVP. Para cada estado de reserva ou estado de caminho está associado um temporizador em etapas. O temporizador inicia com um pequeno intervalo de tempo de retransmissão (3s) no qual vai aumentando exponencialmente até alcançar um limite. A diminuição no tempo de retransmissão de mensagens RSVP ocasiona tempos menores na configuração de reservas de recursos. A confiabilidade é fornecida através de mensagens de reconhecimento *PathAck* e *ResvAck* entre os nós (reconhecimento salto-a-salto) de uma rota entre a fonte e o receptor.

Mathy et al. [19] apresentam um mecanismo de estabelecimento rápido (*FEM - Fast Establisment Mechanism*) de reservas de recursos do protocolo RSVP, semelhante ao emprego de temporizadores renovação em etapas. O trabalho apresenta um modelo matemático (estocástico) da fase de estabelecimento de fluxos do protocolo RSVP e realiza simulações para fluxos ponto-a-ponto demonstrando a redução no tempo de estabelecimento das reservas. O mecanismo FEM fornece confiabilidade através de reconhecimento de mensagens RSVP entre os usuários finais e não usa reconhecimento salto-a-salto como citado em [18].

1.3 Provisão de QoS na *Internet*

Além da arquitetura de serviços integrados, diferentes modelos de serviços e mecanismos para fornecer garantias de qualidade de serviço na *Internet* têm sido propostos e são apresentados nesta Seção.

Arquitetura de Serviços Diferenciados

O modelo de Serviços Diferenciados (*Differentiated Services - DiffServ*) [27] tem o objetivo de prover serviços distintos de forma escalável sem a necessidade de manutenção de um estado por fluxo através de sinalização em cada roteador ao longo da rota do fluxo. O tratamento fornecido por um nó *DiffServ* é efetuado em uma agregação de fluxos, não mais em fluxos individuais.

Os fluxos individuais são classificados em agregações de fluxos pré-estabelecidas que receberão diferentes serviços entre elas. A marcação e classificação de pacotes ocorre nos roteadores de fronteira de um domínio *DiffServ* através da utilização do campo DS dentro do cabeçalho do pacote IP definido pelo IETF. O campo DS é a renomeação do campo ToS (*Type of Service*) do cabeçalho IP. Decisões de escalonamento nos roteadores são baseadas no campo DS. Os valores contidos no campo DS definem diferentes comportamentos por nó (*Per Hop Behavior - PHB*) que implementam diferentes classes de serviços além do serviço de melhor-esforço IP.

A arquitetura de serviços diferenciados é um modelo que fornece prioridades de serviços, não fornece garantias severas de QoS, pois não existe reserva explícita de recursos. Por exemplo, para um cliente receber serviços diferenciados de seu provedor de serviços *Internet*, seu contrato de serviço (*SLA - Service Level Agreement*) tem que incluir as classes de serviços e a quantidade de tráfego para cada classe. Estes parâmetros podem ser configurados de forma estática ou dinâmica. No caso de SLAs dinâmicos, existe a necessidade de mecanismos de sinalização como o protocolo RSVP [29, 30].

Arquitetura MPLS

A arquitetura MPLS (*Multiprotocol Label Switching Architecture*)[31] é uma tecnologia de comutação de rótulos (*labels*) que tem o objetivo de integrar a funcionalidade e a flexibilidade da camada de rede (utilizando informações de roteamento) com a agilidade e a capacidade de tráfego da camada de enlace (empregando comutadores de alta velocidade). A arquitetura MPLS está conceitualmente localizada entre a camada de enlace e a camada de rede segundo o modelo OSI (*Open Systems Interconnection*). A idéia básica do MPLS é adicionar um rótulo nos pacotes que entram em redes MPLS. Os roteadores de comutação de rótulos (*Label-Switching Routers - LSRs*) fazem a classificação e o encaminhamento. Além de prover os serviços para os pacotes baseado no valor do rótulo.

O roteador de ingresso LSR de um domínio MPLS classifica e encaminha os pacotes de chegada baseado na informação do cabeçalho do pacote IP e na informação armazenada na tabela de roteamento. Um rótulo é adicionado no pacote. Dentro do domínio MPLS, este rótulo é usado como índice para a tabela de encaminhamento do roteador LSR. O rótulo de entrada é substituído por um rótulo de saída e o pacote é enviado para o próximo roteador LSR. O rótulo é removido antes do pacote sair do domínio MPLS. O caminho seguido pelos pacotes entre os roteadores de ingresso e egresso são chamados de caminhos comutados por rótulos (*Label-Switched*

Paths - LSPs). Os caminhos LSPs são configurados usando um protocolo de sinalização, como o protocolo RSVP [32] ou pelo protocolo LDP (*Label Distribution Protocol*) [31].

Roteamento baseado em restrições

Roteamento baseado em restrições [33] é o processo de fazer decisões de roteamento levando em conta um conjunto de restrições. Estas restrições podem ser requisitos de QoS como atraso e largura de banda, e neste caso é chamado de roteamento baseado em QoS, mas em geral outras restrições como políticas configuradas pelos administradores de rede são possíveis. O objetivo do roteamento baseado em restrições é construir rotas que alcancem certos requisitos e aumentem a eficiência na utilização dos recursos da rede.

O roteamento clássico geralmente considera uma única métrica na computação da rota, como atraso ou número de saltos, ou um custo configurado na rede. Roteamento baseado em QoS é mais complexo porque decisões de roteamento podem ser baseadas em n métricas [34] e porque algumas métricas, tais como largura de banda disponível, podem ser muito mais dinâmicas do que a métrica número de saltos.

Serviços Integrados sobre Serviços Diferenciados

A arquitetura *IntServ* fornece garantias de QoS para aplicações fim-a-fim (fluxos individuais) e apresenta problemas de escalabilidade quando o número de sessões ativas aumenta nos roteadores. A arquitetura é adequada para a fronteira da rede onde o número de conexões é limitado. O modelo *DiffServ* foi planejado para fornecer QoS para fluxos agregados, e não apresenta questões de escalabilidade, e assim é adequado para o núcleo da rede. Portanto, acredita-se que uma parte significante da *Internet* da próxima geração consistirá da integração destes dois modelos [35]. Como resultado, as arquiteturas com o modelo *IntServ* na fronteira da rede e o modelo *DiffServ* no núcleo da rede têm sido propostas [36, 37].

A arquitetura *IntServ-DiffServ* necessita de um mapeamento de fluxos individuais para as classes *DiffServ*. O trabalho de Ivancic et. al [35] descreve uma função de mapeamento executada no roteador de ingresso do domínio *DiffServ*. A função de mapeamento é usada para atribuir um código DSCP (*DiffServ Code Point*) na chegada de pacotes pertencentes a um determinado fluxo. O fluxo é especificado pelos parâmetros do objeto FLOWSPEC (*Flow Specification*) do protocolo RSVP no domínio *IntServ* [4].

1.4 Posicionamento do Trabalho

O objetivo deste trabalho é apresentar um mecanismo de agregação de estados do protocolo RSVP que identifica os fluxos *multicast* que estão em trânsito (fluxos que somente atravessam a região, sem iniciar ou terminar no seu interior) em uma determinada região de agregação, agregando seus estados entre os roteadores de fronteira. A região de agregação pode ser, por exemplo, um domínio, um sistema autônomo, um provedor de serviços *Internet*, ou uma Rede Virtual Privada. O mecanismo agrega os fluxos utilizando a técnica de tunelamento IP-sobre-IP e está baseado no trabalho de Terzis et al. [5], adicionando funcionalidades para agregar fluxos *multicast*. Foram realizadas simulações para grupos esparsos e os resultados mostram uma substancial redução dos estados nos roteadores interiores.

O estabelecimento de uma reserva de recursos apresenta uma longa latência quando ocorrem perdas de mensagens RSVP. Perdas de pacotes na *Internet* podem ser freqüentes em alguns enlaces e a configuração multidestinatária piora ainda mais esta característica. Medidas realizadas no MBone (*Internet Multicast Backbone*) [38] apresentaram uma taxa média de perda de pacotes de aproximadamente 1 a 2% e, ocasionalmente podem alcançar 20% em enlaces com congestionamento. Este problema não é aceitável para aplicações multimídias em tempo real, tais como conferência de áudio/vídeo ou Voz sobre IP, pois estas aplicações experimentam degradação na qualidade com o aumento de perda de pacotes e longos atrasos na rede. Assim, este trabalho também aborda o emprego de temporizadores de renovação em etapas (*staged refresh timers*) [18] no mecanismo de agregação, para configuração de reserva de recursos de fluxos *multicast*. Temporizadores de renovação em etapas controlam o intervalo de retransmissão das mensagens RSVP túneis e reduzem o tempo de configuração de reservas em relação aos temporizadores de renovação de tempo fixo do protocolo RSVP original.

Foram realizadas simulações para avaliar o mecanismo na presença de perdas de mensagens de controle. Dois modelos de perdas foram utilizados: a taxa média de perdas de pacotes e perdas periódicas. A taxa média avalia o impacto da perda de pacotes na qualidade de serviço em períodos de longa duração na região de agregação. O modelo de perdas periódicas avalia o impacto da perda de pacotes na qualidade de serviço das aplicações multimídias em períodos de congestionamento. Os resultados obtidos mostram uma significativa diminuição no tempo de estabelecimento de sessões *multicast* com o uso de temporizadores de renovação em etapas.

O trabalho está organizado da seguinte forma. O Capítulo 2 apresenta uma visão geral e implementações do protocolo RSVP. Também relata outros protocolos de sinalização de QoS e propostas para uma nova versão do protocolo RSVP. O Capítulo 3 apresenta tipos de perdas de pacotes e sua modelagem que servem de base para as simulações do mecanismo de agregação. O Capítulo 4 descreve o modelo de túneis RSVP para fluxos ponto-a-ponto e o mecanismo estendido para agregar fluxos *multicast*. O Capítulo 5 mostra a implementação do mecanismo de agregação baseado em túneis IP-sobre-IP no simulador de redes *ns* e os resultados obtidos através de simulações para avaliar o desempenho do mecanismo de agregação na presença de perdas de mensagens de controle. O Capítulo 6 apresenta as conclusões do trabalho desenvolvido e sugestões para sua continuação. O Apêndice A descreve as principais características do simulador *ns* e a programação na linguagem OTcl para executar as simulações do mecanismo de agregação *multicast*. O Apêndice B descreve as classes utilizadas na implementação do mecanismo de agregação.

Capítulo 2

O Protocolo RSVP

2.1 Estrutura e Funcionamento

O protocolo RSVP [39] pode ser utilizado por uma estação para solicitar reserva de recursos, através da classe de serviço garantido, serviço de carga controlada e serviço garantido modificado com *pool*, para fluxos de dados de uma aplicação ponto-a-ponto ou multiponto.

O protocolo RSVP é orientado a receptor, isto é, o receptor do fluxo de dados inicia e mantém as reservas de recursos usadas para o fluxo. O protocolo RSVP identifica uma sessão de um determinado fluxo pela combinação do endereço destino, tipo de protocolo de transporte e número da porta de destino. O protocolo RSVP não é um protocolo de roteamento. Ele usa as rotas escolhidas pelos protocolos de roteamento para encaminhar suas mensagens e para reservar recursos para os fluxos de dados.

2.1.1 Classes de Serviços

O serviço de carga controlada [40] reserva largura de banda, mas não garante limites de atraso. O pedido de reserva sendo aceito, o roteador oferece um serviço equivalente ao melhor esforço em uma rede de carga leve. O fluxo que recebe o serviço de carga controlada não se deteriora de forma visível, como ocorre com o fluxo que recebe o serviço de melhor-esforço quando a carga da rede aumenta. O serviço de carga controlada é adequado para aplicações em tempo-real adaptativas que podem tolerar uma variação no atraso e na perda de pacotes. Os roteadores que implementam o serviço de carga controlada devem verificar a conformidade dos fluxos com os recursos reservados. O roteador divide os fluxos em grupos que estão de acordo os recursos reservados e grupos de fluxos não conformes. Os fluxos que não estão em conformidade são encaminhados pelo serviço de melhor-esforço.

O serviço garantido [41] fornece uma garantia de largura de banda e um limite de atraso fim-a-fim. Ele é planejado para aplicações com requisitos severos de encaminhamento em tempo-real, tais como, aplicações de áudio e vídeo que utilizam *buffers* de exibição (*playback*), e não toleram a chegada de algum pacote após seu tempo de exibição. Cada roteador caracteriza o serviço garantido para um fluxo alocando uma largura de banda R, e um espaço de *buffer* B que o fluxo consome. Isto é realizado pela aproximação do modelo de fluido do serviço [42].

Em um modelo de fluido perfeito, a especificação do fluxo é baseada no modelo *token bucket*, no qual o fluxo terá uma taxa de geração de *tokens* r e o tamanho do *token* b. O fluxo terá um atraso limitado por b/R fornecido $R \geq r$. O atraso b/R deve ser acrescentado por dois termos de erro, C e D [23]. O termo C representa o atraso que o fluxo deve experimentar dependente da taxa R. O termo D representa o tempo de trânsito (não depende da taxa R) no pior caso dentro do roteador. Para o roteador que usa o controle de tráfego *Stop-and-Go* [43], o termo D representa o número máximo de quadros (*frames*) que um datagrama deve esperar antes de ser transmitido. Assim, o atraso total fim-a-fim de um pacote nas filas dos roteadores entre a fonte e o destino está limitado por $b/R + C_{tot}/R + D_{tot}$, onde C_{tot} e D_{tot} representam o somatório dos termos C e D para cada roteador no caminho do fluxo.

O serviço garantido fornece um serviço determinístico que tende a desperdiçar recursos, pois mais largura de banda é reservada do que o necessário [23]. Assim, o trabalho de Song e Lutfiyya [44] apresenta um novo serviço chamado MSGP (*Modified Guaranteed Service with Pool*). O MGSP é semelhante ao serviço garantido com a diferença que a parte do recurso reservado e não usado pelo fluxo será retornada para um *pool* de recursos disponíveis que pode ser utilizado por outros fluxos. Se recursos extras são requeridos, então o fluxo pode usar recursos disponíveis do *pool*. Isto permite a alocação dinâmica de recursos durante a sessão e é possível fazer pedidos adicionais de recursos do *pool* sem iniciar uma nova sessão.

Os fluxos que solicitam um serviço MGSP são configurados de maneira similar aos fluxos que pedem um Serviço Garantido (SG). No serviço MGSP cada roteador tem um agente que monitora o uso de recursos atuais de um fluxo após completar a reserva. O valor monitorado é um atributo do estado de reserva do fluxo. Quando uma mensagem de renovação *Resv* é recebida, o valor monitorado mais recente e o valor reservado são comparados. Se o valor monitorado for maior ou igual que o valor reservado, o valor reservado é mantido e a necessidade de recursos que excedem o valor reservado podem ser retirados do *pool* se for possível. Por outro lado, o valor reservado para o fluxo é configurado com o valor monitorado. A quantidade de recursos do *pool* pode ser atualizada através do valor monitorado e do valor estimado do *pool*. A equação abaixo exemplifica o uso do *pool* de recursos:

$$(RR_{SG_i} - RM_i)_{max} + RM_{pool,medio} \tag{2.1}$$

Para um fluxo i do serviço MGSP, o termo RR_{SG_i} representa os recursos iniciais alocados para o fluxo i. O termo RM_i representa o valor corrente monitorado para o fluxo i. O termo $RM_{pool,medio}$ representa o valor médio monitorado do *pool* e a parcela $(RR_{SG_i} - RM_i)_{max}$ representa a maior diferença para todos os fluxos entre a quantidade inicialmente reservada e a quantidade monitorada. Como um fluxo MGSP é configurado como um fluxo SG, a alocação inicial de recursos é baseada no pior caso e a diferença $(RR_{SG_i} - RM_i)_{max}$ é adicionada ao *pool*. Quando ocorre o atraso no pior caso e o fluxo necessita de recursos, o fluxo pode usar os recursos disponíveis no *pool*.

2.1.2 Mensagens

As principais mensagens usadas pelo protocolo RSVP são as mensagens *Path*, transmitidas pelos emissores, e as mensagens *Resv*, transmitidas pelos receptores. A Figura 2.1 ilustra um exemplo de uma sessão *multicast* composta de uma fonte e quatro receptores. A mensagem *Path* instala o estado de roteamento reverso em cada roteador ao longo do caminho, isto é, o estado do caminho percorrido pelo fluxo é armazenado em todos os roteadores entre a fonte e o destino. Também fornece aos receptores, informações sobre as características de tráfego do emissor. A mensagem *Resv* (*Reservation*) conduz os pedidos de reservas para os roteadores ao longo do caminho entre os emissores e receptores. Quando uma mensagem *Resv* é recebida por um roteador, o controle de admissão é consultado para verificar se é possível realizar a reserva. A sessão é estabelecida quando a mensagem *Resv* chega na fonte e o pedido de reserva foi aceito em todos os roteadores ao longo do caminho.

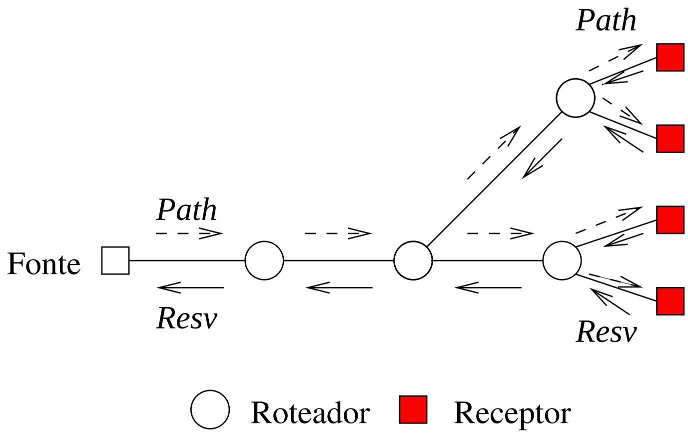

Figura 2.1: Principais mensagens do protocolo RSVP.

Este modelo básico de reserva é chamado de *One Pass*, isto é, o receptor envia o pedido de reserva na direção da fonte e cada nó no caminho aceita ou rejeita o pedido. Este esquema não fornece ao receptor um modo de controlar a qualidade de serviço fim-a-fim. Entretanto, o protocolo RSVP também fornece o modelo básico de reserva com mecanismo de controle de QoS fim-a-fim conhecido como *One Pass With Advertising* (OPWA) [45]. Neste esquema, as mensagens *Path* conduzem o objeto ADSPEC (*Advertising Specification*) [46] que contém parâmetros como: o somatório da latência em cada enlace na rota, a menor largura de banda dos enlaces percorridos, a menor MTU (*Maximum Transmission Unit*) dos enlaces no caminho, o número de roteadores que possuem o protocolo RSVP, etc. Estas informações devem ser usadas pelos receptores para construir ou ajustar dinamicamente um pedido de reserva apropriado aos requisitos da aplicação multimídia.

Para exclusão de estados do protocolo RSVP são utilizados dois tipos de mensagens: a mensagem *PathTear* e a mensagem *ResvTear*. Embora não seja necessário remover explicitamente uma reserva, a RFC 2205 [39] recomenda que as estações enviem uma mensagem de remoção logo que a aplicação termine suas atividades de transmissão ou recepção de dados. A mensagem *PathTear* é transmitida da fonte ou de um roteador na direção do(s) receptor(es) removendo o estado do caminho e o estado da reserva. A mensagem *ResvTear* remove o estado da reserva e é encaminhada na direção da(s) fonte(s). Uma mensagem de remoção deve ser enviada por uma aplicação (fonte ou receptor) ou por um roteador como resultado de um *timeout* do estado ou por iniciativa da aplicação. Uma vez iniciado o processo de remoção, a mensagem de remoção deve ser encaminhada salto-a-salto sem atraso. Uma mensagem de remoção exclui o estado armazenado no nó onde é recebida. Esta mudança de estado será propagada imediatamente para o próximo nó.

Para tratamento de erros, são utilizadas duas mensagens: a mensagem *PathErr* e a mensagem *ResvErr*. A mensagem *PathErr* é transmitida para o emissor que causou o erro e não muda o estado do caminho nos roteadores ao longo do caminho. As mensagens *ResvErr* são encaminhadas para os receptores.

Uma mensagem do protocolo RSVP consiste de um cabeçalho comum, seguido de um conjunto de objetos da mensagem.

Os campos do cabeçalho são (Figura 2.2):

- versão (4 bits)- número da versão do protocolo. Esta é a versão 1;
- flags (4 bits) - não está definido;
- tipo msg (8 bits) - especifica o tipo de mensagem RSVP;

- checksum RSVP (16 bits) - controle de erro para toda a mensagem;
- TTL (8 bits) - tempo de vida da mensagem (*TTL - Time to Live*) do pacote IP na qual a mensagem está contida;
- reservado - campo reservado para uso futuro;
- tamanho RSVP (16 bits) - tamanho da mensagem RSVP.

0	4	8	16	31
Versão	Flags	Tipo Msg	Checksum RSVP	
TTL		Reservado	Tamanho RSVP	

Figura 2.2: Formato do cabeçalho das mensagens do protocolo RSVP.

O objetos do protocolo RSVP são encapsulados nas mensagens para especificar as informações necessárias para a configuração da reserva, o controle de QoS fim-a-fim e a manutenção das tabelas de estados. A Tabela 2.1 descreve os principais objetos contidos na especificação do protocolo [39].

0	16	24	31
Tamanho	Classe	Tipo	
Conteúdo do Objeto			

Figura 2.3: Formato do objeto nas mensagens do protocolo RSVP.

A Figura 2.3 ilustra o formato do objeto na mensagem RSVP. Cada objeto consiste de palavras de 32 bits com um cabeçalho composto dos seguintes campos:

- tamanho - tamanho total do objeto em bytes;
- classe - identifica a classe do objeto;
- tipo - identifica o tipo do objeto.

Os valores da classe e do tipo do objeto estão definidos na especificação do protocolo RSVP [39]. A Tabela 2.1 mostra os objetos do protocolo RSVP.

Tabela 2.1: Objetos do protocolo RSVP.

Objeto	Descrição
SESSION	Endereço, protocolo de transporte e porta de destino.
RSVP_HOP	Endereço IP e identificador da interface de saída do nó RSVP que enviou a mensagem.
TIME_VALUES	Valor do período de renovação usado pelo criador da mensagem.
STYLE	Estilo de reserva.
FLOWSPEC	QoS requisitada pelo receptor e características de tráfego da fonte.
FILTER_SPEC	Endereço(s) e porta(s) da(s) fonte(s), usado na mensagem *Resv*.
SENDER_TEMPLATE	Endereço(s) da(s) fonte(s), usado na mensagem *Path*.
ADSPEC	Parâmetros para monitorar à qualidade de serviço fim-a-fim.
ERROR_SPEC	Especifica o tipo de erro nas mensagens *PathErr* e *ResvErr*.
POLICY_DATA	Informações da aplicação para controle de políticas.
RESV_CONFIRM	Endereço IP de um receptor que pediu uma confirmação de reserva.

2.1.3 O Mecanismo *soft-state*

Os *estados* nos nós de uma rede são as informações armazenadas pelos protocolos sobre as condições da rede e podem ser modificados para refletir as mudanças na rede. Os estados mantidos pelos nós na rede podem ser divididos em dois tipos: o modelo *hard-state* e o modelo *soft-state*. No modelo *hard-state*, os estados são instalados nos nós na recepção de uma mensagem de configuração e são removidos somente na recepção de uma mensagem explícita de remoção. O protocolo de reserva ST-II [47] e o protocolo de roteamento CBT (*Core Based Tree*) [48] são exemplos de protocolos que usam o paradigma *hard-state*. As arquiteturas que usam o modelo *hard-state* usam explicitamente mensagens de reconhecimento, aumentando a confiabilidade, entretanto, aumentam a complexidade do protocolo.

O paradigma *soft-state* usa mensagens de renovação para manter ativo o estado, caso o nó da rede não receba mensagens periódicas em um determinado intervalo de tempo, o estado é removido. O protocolo RSVP, o protocolo de roteamento PIM (*Protocol Independent Multicast*) [49, 50] e o protocolo de transporte RTP (*Real Time Protocol*) [51] são exemplos de protocolos que utilizam o paradigma *soft-state*.

Durante a estabilidade da rede, os protocolos *hard-state* requerem menor controle de tráfego do que os protocolos soft-state porque não enviam mensagens de renovação. Os protocolos *soft-state* fornecem uma adaptação mais rápida as mudanças na rede, mas gastam mais mensagens de renovação. Entretanto, quando a rede é dinâmica, mensagens de controle do protocolo *hard-state* são geradas para se adaptarem as mudanças. Em tal cenário a largura de banda usada pelas mensagens de controle seriam comparáveis para ambos paradigmas [52].

No protocolo RSVP, o estado de uma determinada reserva é mantido pelos roteadores através do mecanismo *soft-state*. Existe um temporizador associado a cada estado da reserva, que controla o tempo de duração do estado. O protocolo RSVP mantém atualizado o estado através do envio periódico de mensagens *Path* e mensagens *Resv* pela fonte e destino, respectivamente. O intervalo de tempo padrão de envio das mensagens é de 30 s. Se após um certo período de tempo um roteador não receber, por exemplo, uma mensagem *Resv* de determinado destino, ele libera os recursos que haviam sido alocados para aquele destino. A abordagem *soft-state* adiciona grande capacidade de adaptação ao protocolo RSVP para os casos de reconfiguração de reservas, falhas da rede, mudanças de rotas e a dinâmica de grupos *multicast*. Entretanto, aumenta a sobrecarga de processamento no roteador, pois o número de mensagens de renovação enviadas e recebidas periodicamente aumenta linearmente com o número de sessões RSVP ativas.

2.1.4 Estilos de Reservas

O protocolo RSVP define estilos de reserva (Tabela 2.2) para caracterização de uma sessão com objetivo de melhor adaptação das reservas às características de transmissão. Os estilos de reservas oferecem opções aos fluxos na seleção das fontes de transmissão e do modo de compartilhamento do canal entre as fontes.

Tabela 2.2: Estilos de reserva do protocolo RSVP.

	Reservas	
Seleção da Fonte	Exclusiva	Compartilhada
Explícita	Estilo *Fixed-Filter* (FF)	Estilo *Shared-Explicit* (SE)
Implícita	Não definido	Estilo *Wildcard-Filter* (WF)

O modo de compartilhamento determina se as reservas serão realizadas de forma exclusiva (FF) para cada fonte escolhida na sessão ou se a reserva do canal será compartilhada (SE ou WF) por todas as fontes. A escolha das fontes que transmitem para uma sessão pode ser realizada de maneira explícita (FF ou SE), selecionando as fontes nominalmente, ou de maneira implícita (WF), na qual todas as fontes são selecionadas. A escolha de fontes permite que um receptor escolha alguns fluxos dentre todos os oferecidos em uma sessão.

Os estilos de reserva SE e WF são adequadas para aplicações tipo conferência, que podem ter várias fontes, mas que somente uma fonte esteja ativa por vez. O estilo FF, cria reservas distintas para fluxos de diferentes emissores, é adequada para comunicação ponto-a-ponto.

Quando o roteador recebe mensagens *Resv* de uma sessão, ele executa o processo de junção (*merge*) das reservas, isto é, encaminha a mensagem *Resv* com uma reserva resultante dos pedidos recebidos. A reserva resultante de uma determinada sessão na interface de saída do roteador é a maior de todos os pedidos recebidos pelo roteador. A Figura 2.4 mostra o processo de junção para uma sessão *multicast* composta de uma fonte e quatro receptores. Cada receptor envia uma mensagem *Resv* com a largura de banda expressa em *kbps*, colocada entre parênteses. Cada roteador encaminha para a fonte, uma mensagem *Resv* com o valor resultante do processo de junção. O processo de junção deve ocorrer entre mensagens com o mesmo estilo de reserva.

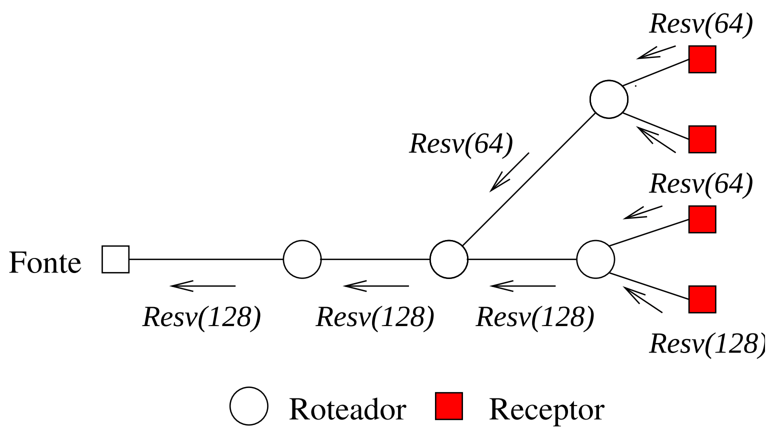

Figura 2.4: Encaminhamento da mensagem *Resv* após o processo de junção.

2.2 Implementações do Protocolo RSVP

O protocolo RSVP foi originalmente proposto como o protocolo de reservas de recursos da arquitetura de serviços integrados para ser utilizado nos roteadores da *Internet*. Entretanto, devido ao problema de escalabilidade no armazenamento de estados e no processamento da mensagens, diversas implementações têm sido realizadas em redes de menor porte.

O relatório técnico [53] apresenta uma lista de implementações do protocolo RSVP (no roteador ou na estação cliente) realizadas por empresas públicas e privadas, constituindo esforços iniciais para comercialização do protocolo RSVP.

Podemos encontrar alguns trabalhos na literatura onde o protocolo RSVP é testado ou avaliado. Em [54] são realizados diversos testes com o protocolo para avaliar: o tempo de configuração de uma reserva, a sobrecarga devido às mensagens do protocolo e a sobrecarga devido ao escalonamento de pacotes. Os resultados de testes com o objetivo de verificar a correção da implementação do protocolo realizada pelas empresas Intel, MCI e Cisco podem ser encontrados em [55].

O trabalho de Karsten [56] apresenta um projeto e implementação do protocolo utilizando técnicas de modelagem de banco de dados relacional como o modelo entidade-relacionamento, para especificar as tabelas de estados. Um dos objetivos é tornar mais compreensível as operações do protocolo do que as regras de processamento de mensagens [57].

Em [58] são realizados testes para avaliar a qualidade de serviço de aplicações de vídeo onde o protocolo RSVP atua em roteadores que implementam o escalonador CBQ (*Class Based Queue*) [59].

2.3 Controle de Admissão baseado em Políticas

A admissão de um fluxo na arquitetura de serviços integrados é baseada somente no pedido de reserva de recursos do receptor e na capacidade disponível de recursos do roteador para decidir se aceita ou rejeita o pedido. Entretanto, este tipo de controle de admissão não é suficiente para atender as necessidades atuais na *Internet*, que é composta, na sua maioria, de provedores de serviços e redes corporativas. Os administradores de redes precisam de uma infra-estrutura de controle de políticas (*policy*) que regule quais usuários, aplicações, ou estações devem acessar quais recursos/serviços sob determinados critérios, como requisitos de tráfego, segurança, hora do dia, dia da semana, etc.

Política é a combinação de regras e serviços onde as regras definem critérios para acesso e uso de recursos pelos usuários de uma rede. Neste contexto, estruturas de gerência de políticas foram propostas [60, 61], onde o controle de admissão baseado em políticas é usado em conjunto com o protocolo RSVP.

Para dar suporte às atividades de gerência de políticas, Yavatkar et al. [60] descrevem uma estrutura de controle de admissão baseado em políticas que utiliza as informações das aplicações/usuários contidas no objeto POLICY_DATA [62] transportado pelas mensagens RSVP. Estas informações servem de base para decidir a admissão de um fluxo.

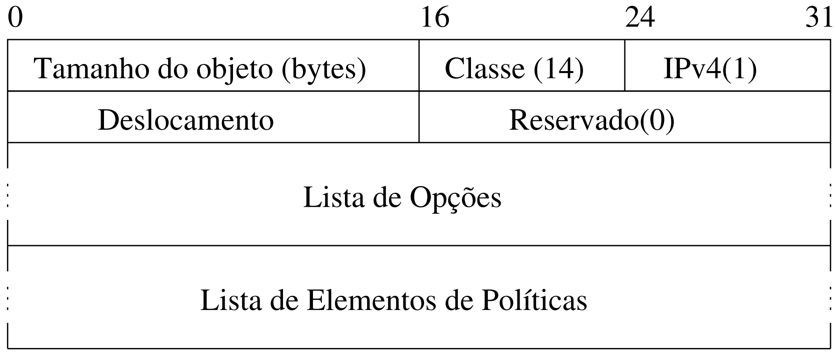

Figura 2.5: Formato do objeto POLICY_DATA.

A Figura 2.5 mostra o formato básico do objeto POLICY_DATA, composto dos seguintes campos:

- tamanho do objeto em bytes;
- número da classe do objeto RSVP - 14 para o objeto POLICY_DATA;
- tipo de protocolo IP - valor 1 para IPv4;
- deslocamento (16 bits) - deslocamento em bytes da porção de dados (do primeiro byte do cabeçalho do objeto);
- reservado (16 bits) - sempre 0;
- Lista de Opções (tamanho variável) - pode incluir os objetos FILTER_SPEC e RSVP_HOP;
- Lista de Elementos de Políticas (tamanho variável) - descrevem informações sobre a aplicação e/ou o usuário.

Uma estrutura do elemento de política da aplicação é descrita na RFC2872 [63]. A estrutura conduz informações detalhadas sobre uma determinada aplicação como: identificador do local da aplicação (por exemplo, o endereço do *site*), nome da aplicação e versão da aplicação.

A Figura 2.6 ilustra um roteador que executa o controle de políticas através de dois módulos : ponto de aplicação de políticas (PEP - *Policy Enforcement Point*) e

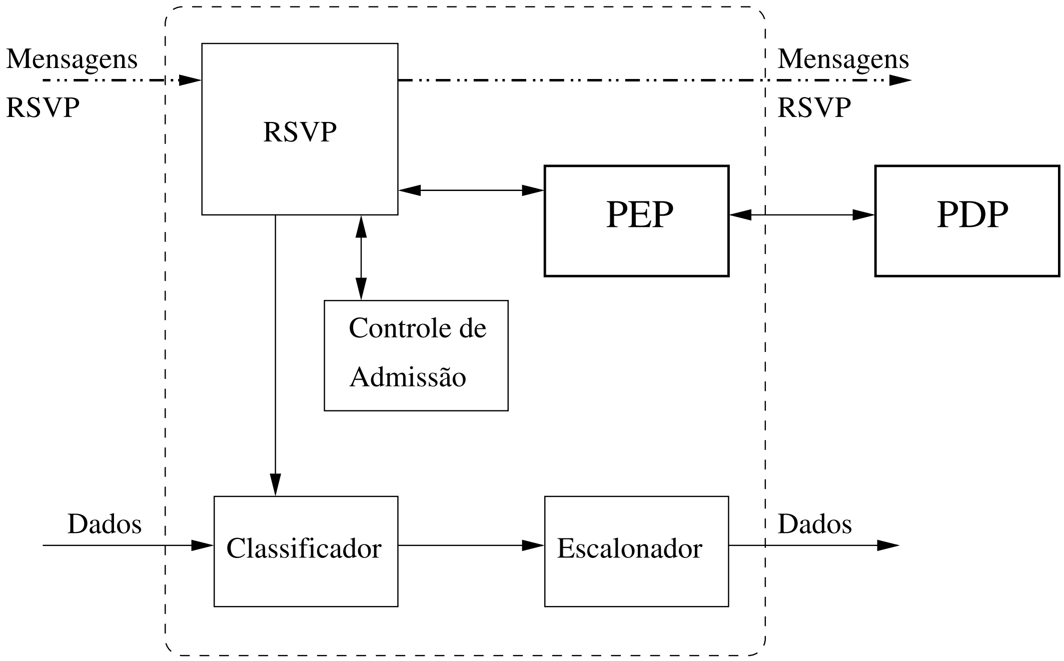

Figura 2.6: Roteador com uma estrutura de controle de políticas.

ponto de decisão de políticas (PDP - *Policy Decision Point*).

Na chegada de uma mensagem *Resv* ou mensagem *Path* no roteador, o protocolo RSVP envia o pedido para o módulo PEP. O módulo PEP extrai a informação do objeto POLICY_DATA e compara com a informação armazenada pelo módulo PDP. A decisão é baseada no resultado da comparação e é enviada ao protocolo RSVP.

O módulo PDP deve determinar a(s) classe(s) de serviço(s) que podem estar associadas com o pedido do fluxo. Isto é baseado em políticas e no estado corrente da rede. Se o módulo PEP for remoto, isto é, um servidor de políticas (*policy server*), a comunicação entre o PEP e o PDP pode ser realizada pelo protocolo COPS (*Common Open Policy Service*) [64, 65].

Como exemplo de aplicação de políticas, podemos citar o trabalho de Ponnappan et al. [66] que relata o projeto, a implementação e a avaliação de desempenho de um sistema de gerenciamento de QoS baseado em políticas para as arquiteturas *IntServ* e *DiffServ*. O sistema usa o protocolo COPS para servir de interface com o dispositivo de rede. O projeto é baseado em componentes distribuídos e a arquitetura CORBA (*Commom Object Request Broker Architecture*) é usada como um *middleware* para interação entre os componentes. As políticas *DiffServ* instaladas no roteador Linux são baseadas em regras atribuídas para interface de rede. Medidas preliminares de desempenho mostram que o tempo de resposta do servidor de políticas é aproximadamente 90ms para 25 pedidos COPS-RSVP por segundo.

2.4 Extensões do Protocolo RSVP para Mobilidade

Com o contínuo crescimento de redes sem fio e do número de usuários móveis tem surgido a necessidade de desenvolver mecanismos de qualidade de serviço para as aplicações multimídias em redes sem fio e redes móveis. Alguns trabalhos têm sido propostos para adicionar funcionalidades ao protocolo RSVP para operar em redes móveis e redes sem fio [67, 68, 69, 70].

O trabalho de Talukdar et al. [67] apresentou uma arquitetura de serviços integrados para fornecer suporte às estações móveis. Foi proposto estender os serviços com a dimensão de mobilidade. Assume-se que a mobilidade é previsível e pode ser caracterizada por uma especificação que contenha um conjunto de células que o nó móvel espera visitar durante o tempo de vida do fluxo. Um nó pode pedir um serviço que é *independente de mobilidade* pelo qual receberá o mesmo serviço em alguma célula de acordo com seu perfil de mobilidade ou um *serviço dependente de mobilidade* que é somente garantido na localização corrente.

Talukdar et al. [69] propuseram o protocolo de sinalização MRSVP como parte da arquitetura de serviços integrados em redes móveis [67]. O protocolo MRSVP faz a reserva de recursos em todas as localizações onde o nó móvel espera visitar durante o tempo de conexão. O nó móvel fará uma reserva ativa na sua atual localização e fará reservas passivas em cada local a ser visitado. A diferença entre reserva ativa e reserva passiva é que nas reservas ativas, os fluxos de dados atravessam o caminho onde recursos estão reservados, enquanto em reservas passivas os recursos são reservados, mas os fluxos não atravessam o caminho. Quando o nó móvel se move de uma célula para outra, a reserva ativa da fonte para a sua localização anterior será configurada como passiva e a reserva passiva para a nova célula será configurada como reserva ativa.

O trabalho de Zhang et al. [68] apresenta um protocolo de sinalização para usuários móveis na arquitetura *IntServ*. O protocolo trabalha através da combinação de Túneis RSVP [5] com o protocolo IP Móvel [71]. O protocolo será referenciado como protocolo RSVP Túnel - IP Móvel (RT-IM). O protocolo RT-IM considera que um serviço independente da mobilidade fornecido pelo protocolo MRSVP pode apresentar dois problemas. Primeiro é se um nó móvel está habilitado a predizer seu itinerário com precisão. Se o nó não prever precisamente seu perfil de mobilidade então o serviço percebido pelo usuário será o serviço dependente de mobilidade. O segundo problema é que os recursos reservados ao longo do caminho podem ser usados ou não (o caso de reservas passivas). Isto ocasiona o desperdício de recursos em redes sem fio. Normalmente os recursos em redes sem fio são menores que os recursos em redes com fio. Assim, o protocolo RT-IM parte da premissa que redes sem fio requerem uma mudança fundamental nas expectativas dos usuários quanto ao serviço fornecido pela rede e ao nível de confiabilidade da rede. As aplicações devem pedir algum serviço da rede mas no caso de falha, as aplicações devem se adaptar ou renegociar o pedido do serviço.

O protocolo RT-IM executa o encapsulamento de mensagens RSVP fim-a-fim sobre um túnel conectando o agente de origem (*home agent*) com o agente de destino (*foreign agent*) e aplicando a reserva de recursos entre estes agentes.

Quando a estação muda de célula, ele envia uma mensagem *Resv* fim-a-fim para o agente de destino, este agente configura uma reserva no túnel RSVP com o agente de origem. Após receber a confirmação da reserva no túnel, o agente de destino encapsula a mensagem *Resv* fim-a-fim e envia para o agente de origem. O túnel entre o agente de origem e o agente de destino pode ser pré-configurado, isto é, um nível de recursos é pré-alocado entre os dois agentes para atender os pedidos dos nós móveis que estão atravessando as duas células. O esquema de pré-alocação diminui a latência no estabelecimento das reservas sobre o túnel.

Um recente trabalho de Y. Suh et al. apresenta um protocolo de reserva de recursos para aplicações multimídias em sistemas de telefonia móvel de terceira geração (sistemas 3G), chamado de protocolo 3G-RSVP [70]. O protocolo proposto usa a abordagem de túneis RSVP [5] no núcleo de uma rede UMTS (*Universal Mobile Telecommnications System*). O protocolo reserva recursos usando túneis IP-sobre-IP entre os nós SGSN (*Serving GPRS Support Node*) e os nós GGSN (*Gateway GPRS Support Node*). O nó SGSN é um nó que realiza as funções de autenticação e autorização de assinantes, controle de admissão, roteamento de pacotes, gerência de mobilidade e tarifação. O nó GGSN é uma interface entre as redes de pacotes e as redes móveis que estão conectadas com núcleo do sistema UMTS. Uma estação móvel envia uma mensagem *Resv* fim-a-fim para o nó SGSN, e este nó estabelece um túnel com reserva de recursos com o nó GGSN para atender o pedido do usuário.

2.5 Protocolos de Sinalização

Esta seção apresenta propostas de protocolos de sinalização de requisitos de qualidade de serviço na *Internet* e compara as propostas com o protocolo RSVP .

Protocolo ST-II

O protocolo ST-II (*Internet Stream Protocol Version 2*) é um protocolo experimental orientado a conexão que opera na camada de rede [47]. Ele foi desenvolvido para dar suporte ao encaminhamento de fluxos de dados para aplicações ponto-a-ponto e ponto-multiponto que requerem garantias de qualidade de serviço. O protocolo ST-II possui mensagens de encaminhamento de dados e mensagens de controle, diferente do protocolo RSVP que contém somente mensagens de controle de QoS.

O ST-II é orientado ao emissor, isto é, o emissor transmite uma mensagem de pedido de reserva de recursos da fonte do fluxo para o destino, e ao longo do caminho os roteadores reservam os recursos. Ambos protocolos permitem receptores heterogêneos, isto é, receptores de um grupo *multicast* podem ter requisitos de QoS diferentes em uma mesma sessão. O protocolo RSVP possui uma maior sobrecarga no processamento de mensagens do que o ST-II porque utiliza a abordagem *soft-*

state. Uma análise mais aprofundada sobre as diferenças entre os protocolos RSVP e ST-II são apresentadas em dois artigos [72, 73].

Protocolo YESSIR

O protocolo YESSIR (*YEt another Sender Session Internet Reservations*) [74] é um protocolo orientado ao emissor e utiliza o protocolo RTP como protocolo de transporte dos fluxos com reservas de recursos e o protocolo RTCP (*Real Time Control Protocol*) para estabelecer as reservas de recursos fim-a-fim. A sinalização é mais eficiente para aplicações ponto-a-ponto e a sobrecarga de processamento é menor do que o protocolo RSVP [75]. O protocolo YESSIR usa a abordagem *soft-state* para manutenção dos estados das reservas.

O protocolo YESSIR é menos abrangente que o protocolo RSVP pois não faz pedidos heterogêneos de múltiplos receptores de um grupo *multicast* e não realiza reserva de recursos para aplicações multiponto-multiponto.

Protocolo Boomerang

O protocolo Boomerang [76] possui somente uma mensagem para configurar a reserva de recursos bidirecional para fluxos ponto-a-ponto. Esta mensagem é encaminhada salto-a-salto na rede realizando a reserva na direção do receptor. Ao chegar no receptor, a mensagem é enviada de volta a fonte fazendo a reserva na direção oposta. As mensagens de reserva são encapsuladas em mensagens ICMP (*Internet Control Message Protocol*).

Uma análise numérica e resultados de simulação mostraram que a sobrecarga de sinalização de mensagens e a probabilidade de bloqueio de pedidos de reservas é menor que o protocolo RSVP [76, 77].

As propostas apresentadas contêm importantes contribuições para sinalização de QoS, e o protocolo Boomerang e o protocolo YESSIR são mais escaláveis que o protocolo RSVP. Entretanto, os dois protocolos fornecem um suporte limitado à aplicações ponto-a-ponto e o protocolo YESSIR fornece suporte à aplicações ponto-multiponto. O protocolo RSVP fornece suporte à aplicações ponto-a-ponto, ponto-multiponto e multiponto-multiponto.

Embora o protocolo RSVP tenha sido projetado originalmente para reserva de recursos, alguns trabalhos usam o protocolo RSVP para conduzir outros tipos de informação na rede, como informações de segurança [78] e informações de rótulos [79, 32] em redes MPLS. Assim, o protocolo RSVP fornece uma flexibilidade e abrangência maior no suporte às aplicações multimídias e pode ser considerado um protocolo de sinalização de propósito geral [80].

2.6 Nova Versão do Protocolo RSVP

Embora o protocolo RSVP seja o protocolo de sinalização mais utilizado, principalmente para atender as necessidade de aplicações *multicast*, ele torna-se complexo para ser usado em aplicações ponto-a-ponto. Assim, duas recentes propostas para uma nova versão do protocolo RSVP foram apresentadas na comunidade científica para reduzir a complexidade e aumentar a escalabilidade do protocolo RSVP [81, 82].

O trabalho de Brunner et al. [81] apresenta as primeiros etapas para a definição de uma nova versão do protocolo RSVP, chamada de RSVPv2. Este trabalho também faz parte do grupo de trabalho NSIS (*Next Steps in Signaling*) do IETF [83]. A meta do protocolo RSVPv2 é fornecer uma abordagem com menor complexidade para que possa melhorar a manutenção das reservas na rede. O novo protocolo deve ser mais eficiente no tempo de configuração de reservas. Também são introduzidos mecanismos que estendem o protocolo para aplicações baseadas no protocolo IP Móvel. Esta versão não substitui o RSVPv1, cada versão deve atender às necessidades específicas de seu projeto.

Os princípios do projeto do protocolo RSVPv2 são:

- o núcleo do protocolo deve ser simples e eficiente, de modo que um conjunto mínimo de funções deve ser usado no núcleo;
- adicionar funções de forma modular, sem sobrecarregar o núcleo do protocolo;
- o protocolo RSVPv2 deve atender basicamente aplicações ponto-a-ponto;

- o protocolo RSVPv2 deve atender às aplicações *multicast* através de um módulo específico, fornecendo suporte a grupos com poucos membros e com receptores homogêneos;

- o protocolo deve ser orientado pela fonte, pois esta abordagem reduz a complexidade do protocolo;

- o mecanismo *soft-state* é realizado apenas nos sistemas-finais e nos roteadores de fronteira, retirando este processamento dos roteadores interiores.

Para verificar e quantificar o projeto do protocolo, foi realizada uma implementação na linguagem C++ e também foram executados testes para avaliação do protótipo. A implementação foi desenvolvida modificando a implementação KOM RSVP da Universidade de Darmstad [56]. A avaliação teve o objetivo de comparar o desempenho entre o protocolo RSVPv1 e o protocolo RSVPv2. Os resultados dos testes apresentaram uma redução no tempo de estabelecimento de sessões e uma diminuição de uso de memória pelo protocolo RSVPv2 em relação ao protocolo RSVPv1.

A segunda proposta é o protocolo RSVP Lite [82] que como o protocolo RSVPv2 é também direcionado para aplicações ponto-a-ponto. O protocolo RSVP Lite possui as seguintes características:

- fornece basicamente suporte para aplicações ponto-a-ponto;

- o pedido de reserva pode ser feito pela fonte ou pelo receptor;

- somente o estado do caminho (*path*) é mantido nos roteadores, os estados do caminho e os estados da reserva são mantidos nas aplicações finais;

- existem mensagens de reconhecimento para as mensagens *Path* e mensagens *Resv*.

Capítulo 3

Perda de Pacotes

A perda de pacotes é um fator determinante na qualidade de aplicações multimídias sensíveis ao atraso, tais como conferência de áudio/vídeo e telefonia na *Internet*. Estas aplicações experimentam degradação na qualidade com o aumento da perda de pacotes e atrasos na rede.

O conhecimento e a modelagem da perda de pacotes na *Internet* são importantes para o projeto e análise de protocolos de comunicações que fornecem um suporte na qualidade de serviço para aplicações multimídias.

3.1 Fontes de Perda de Pacotes

Um sistema de comunicações de áudio/vídeo ponto-a-ponto pode ser representado pelo diagrama em blocos na Figura 3.1. Perdas de informação podem aparecer nos diversos elementos que integram o sistema. Por exemplo, no codificador, partes da informação original são perdidas se técnicas de compressão com perdas são usadas. As placas de rede, no transmissor e no receptor, podem introduzir perdas, se a taxa de chegada de pacotes for maior que a taxa que a placa pode receber e se o *buffer* estiver cheio. A rede pode introduzir uma significante quantidade de perdas.

Por outro lado, os esquemas de codificação para áudio e vídeo estão evoluindo e podem fornecer altas taxas de compressão com uma alta qualidade. As placas

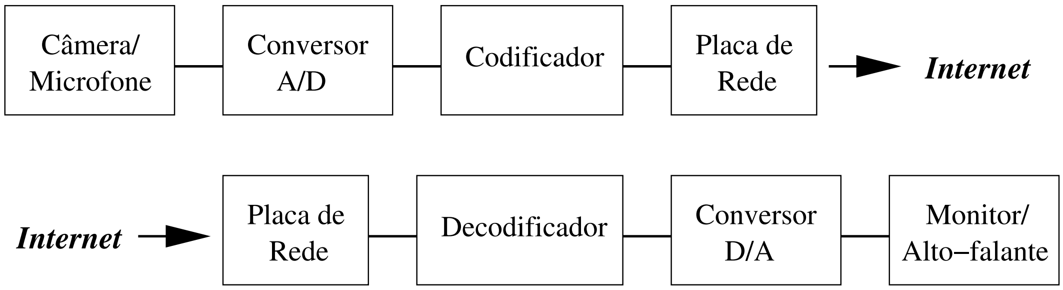

Figura 3.1: Elementos de um sistema de comunicações para transferência multimídia.

de rede usualmente tem taxas altas de transmissão e raramente são um gargalo na transmissão dos dados. Entretanto, perdas de pacotes na *Internet* podem ter um grande impacto na qualidade do áudio/vídeo percebida pelos usuários finais.

As fontes de perda de pacotes na *Internet* estão associadas com os principais elementos da infra-estrutura de rede: os roteadores e os enlaces que conectam os roteadores.

Se a taxa de chegada de pacotes no roteador for maior do que a taxa de encaminhamento, os pacotes são temporariamente armazenados nos *buffers* dos roteadores, esperando serem encaminhados para uma interface de saída. O tamanho dos *buffers* é limitado, e se estiverem cheios, algum novo pacote chegando será descartado. Assim, pacotes são perdidos por causa da sobrecarga nos *buffers*. Esta sobrecarga é a principal causa da perda de pacotes em redes com fio.

Patologias de roteamento como *loops* de pacotes IP na rota [84] causam a expiração do campo *time-to-live* (TTL) no cabeçalho dos pacotes IP e consequentemente o pacote é descartado e considerado perdido.

A falha em enlace é outra fonte de perda de pacotes. Por exemplo, um enlace em redes sem fio muitas vezes tem altas taxas de erro alta devido ao desvanecimento, ruído ou interferência.

3.2 Taxa Média de Perda de Pacotes

Numerosos pesquisadores têm estudado perda de pacotes na *Internet*, e a taxa média de perda de pacotes é diferente para cada experimento. Esta variação é decorrente de fatores como topologia, tipo de tráfego, período do dia, duração do experimento, etc.

Medidas realizadas no MBone (*Internet Multicast Backbone*) apresentaram uma taxa média de perda de pacotes de aproximadamente 1 a 2% [85, 38].

O trabalho de Yajnik et al. [38] apresentou uma análise de perda de pacotes em estações geograficamente distribuídas na Europa e nos Estados Unidos conectadas através do MBone. O objetivo foi examinar a correlação temporal e a correlação espacial na perda de pacotes entre os participantes de uma sessão *multicast*. Correlação espacial significa a perda de um mesmo pacote próximo da fonte percebido por várias estações e correlação temporal, significa a perda de pacotes consecutivos em um receptor. Um estudo posterior de Yajnik et al. [86] apresentou uma taxa média entre 1,38% e 11% de perda de pacotes para tráfego de áudio.

Loguinov e Radha [87] realizaram um experimento com tráfego de vídeo MPEG-4 com baixa taxa de transmissão. Os fluxos ponto-a-ponto foram enviados de um servidor de vídeo para milhares de clientes *dial-up* de três provedores de serviços *Internet* de grande porte nos Estados Unidos. A taxa média de perdas variou entre 0,3 e 1,4%.

Existem sítios que realizam medidas regionais e globais para taxa média de perda de pacotes na *Internet* [88, 89]. As medidas obtidas de perda são para períodos de uma hora, um dia, ou meses. Por exemplo, o *Internet Traffic Report* [88] relata para o dia 18 de fevereiro de 2003, um índice global de perda de pacotes variando entre 2,8 e 4,9%.

A taxa média de perdas dá uma noção geral da qualidade de serviço a longo prazo, e seus efeitos são diferentes para cada tipo de aplicação [90]. Zhang et al. [91] classificaram de uma maneira geral os efeitos da perda de pacotes nas aplicações em categorias apresentadas na Tabela 3.1.

Tabela 3.1: Efeitos da perda de pacotes nas aplicações.

Taxa de Perda	Noção Qualitativa
0-0,5%	Sem perdas
0,5-2%	Perdas com pouco importância
2-5%	Perdas toleráveis
5-10%	Perdas sérias
10-20%	Perdas muito sérias
maior que 20%	Perdas inaceitáveis

3.3 Perdas Periódicas

Perdas periódicas compreendem um importante tipo de perda. As perdas ocorrem em ciclos (ou períodos) de tempo em que os roteadores recebem uma grande quantidade de pacotes. Os *buffers* ficam sobrecarregados e o roteador descarta muitos pacotes. O principal motivo de tal fenômeno é o uso freqüente de temporizadores em protocolos de roteamento, e porque tais temporizadores podem algumas vezes se sincronizarem [92]. Os temporizadores controlam o envio periódico de mensagens de atualização das tabelas de roteamento.

Os períodos de perdas podem ocorrer em ciclos de 30, 60, ou 90 segundos [92, 91]. Durante o período de perda, uma série de pacotes consecutivos são perdidos. A perda de pacotes consecutivos é também chamada de episódio de perda ou perda em rajadas. Dois parâmetros caracterizam a perda de pacotes em rajadas: o tamanho da rajada e a duração da rajada.

O tamanho e a duração da rajada

O tamanho da rajada é definido como o número de pacotes perdidos consecutivamente. Existe uma predominância de rajadas de tamanho pequeno na distribuição do tamanho de rajadas nos experimentos [38, 86, 87]. Por exemplo, o trabalho de

Loguinov e Radha [87] mostra que rajadas de tamanho de até 20 pacotes cobrem 99% das rajadas e rajadas de até dois pacotes cobrem 90% das rajadas.

O tamanho médio da rajada depende do espaçamento entre pacotes durante a transmissão. O experimento de Bolot [93] demonstrou que pacotes UDP espaçados de 8ms sofreram perdas de rajadas maiores (média de 2,5 pacotes) do que pacotes espaçados de 500ms (média de 1,1 pacotes). Yajnik et al. [86] relataram uma correlação similar entre o tamanho da rajada e a distância entre pacotes.

A duração da rajada de perdas pode variar de experimento para experimento. O trabalho de Loguinov e Radha [87] mostrou que mais 98% das durações de perda em rajadas foram menores que 1 (um) segundo, embora ocorressem rajadas com tempos maiores que 36s. O estudo de Yajnik et al. [38] apresentou perdas de 0,6s em intervalos de 30s. O experimento de Zhang et al. [91] mostrou que 95% dos tempos de duração de perdas em rajadas foram menores que 220ms.

3.4 Caracterização da Perda de Pacotes

Probabilidade de perda incondicional e condicional

O trabalho de Bolot [38] mediu a perda de pacotes entre o sítio do INRIA (*Institut National de Recherche en Informatique et en Automatique*) e universidades dos Estados Unidos. Bolot usou pacotes UDP contendo um número de seqüência e sendo enviados em intervalos regulares δ. O foco da análise de perdas foi somente nas probabilidades de perdas de pacotes, tais como a probabilidade de perda incondicional $ppi = P(\text{pacote n é perdido})$ e a probabilidade de perda condicional $ppc = P(\text{pacote n+1 é perdido}|\text{pacote n é perdido})$. Bolot mostrou empiricamente que $ppc \geq ppi$. Para pacotes espaçados muito próximos (valores pequenos de δ), $ppc > ppi$. Em outras palavras, se o pacote anterior for perdido, então a probabilidade que o próximo pacote será perdido é mais alta. Esta correlação entre duas perdas consecutivas indica que a perda aparece em rajadas.

A probabilidade de perda incondicional ppi é um importante parâmetro que afeta o desempenho de aplicações interativas fim-a-fim. Entretanto, as características não podem ser capturadas especificando somente a probabilidade de perda, e elas requerem um estudo detalhado de modelos de perdas. A probabilidade de perda condicional *ppc* refere-se à perda de dois pacotes consecutivos, mas não especifica o comportamento de episódios de perda com tamanhos maiores que dois.

Modelo de perdas de pacotes em rajadas

Um modelo de cadeia de Markov de 2-estados também conhecido como modelo de Gilbert pode ser usado na representação de perda de pacotes em rajadas. A rede com perda pode ser modelada com dois estados ilustrados na Figura 3.2. O estado $X = 0$ representa o estado da chegada do pacote com sucesso e o estado $X = 1$ representa perda do pacote. As probabilidades p_{01} e p_{10} denotam as probabilidades de transição de estados:

$$p_{01} = P(X = 1|X = 0) = P(\text{pacote n é perdido}|\text{pacote n-1 é recebido}) \quad (3.1)$$

$$p_{10} = P(X = 0|X = 1) = P(\text{pacote n é recebido}|\text{pacote n-1 é perdido}) \quad (3.2)$$

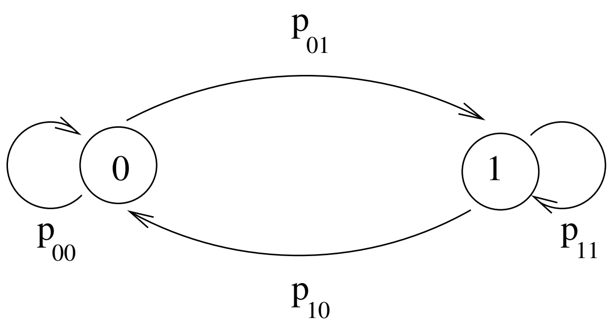

Figura 3.2: Modelo de Gilbert.

Para o estado estacionário, as probabilidades de estado e as probabilidades de transição podem ser descritas na seguinte matriz:

$$\begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}$$

A probabilidade de perda incondicional $P(X = 1) = \dfrac{p_{01}}{p_{01} + p_{10}}$.

O modelo de Gilbert tem uma memória de somente um evento passado. A probabilidade de que o próximo evento será o pacote recebido ou o pacote perdido depende somente do estado anterior.

Este modelo de perdas será usado para avaliar o efeito das perdas na construção de túneis multidestinatários com reserva de recursos.

Capítulo 4

Agregação em Túneis IP-sobre-IP

4.1 Agregação em Túneis para Fluxos Ponto-a-Ponto

A técnica de túneis IP-sobre-IP é um mecanismo bastante utilizado na *Internet* para transportar pacotes através de regiões da rede que não fornecem diretamente o serviço desejado. Entre os exemplos mais comuns do uso da técnica de túneis estão o roteamento IPv6 em roteadores que executam o IPv4, o roteamento *multicast*, o IP móvel [71] e a Rede Virtual Privada [94, 95].

Atualmente existem diversos protocolos para tunelamento IP-sobre-IP. Em [96] é descrito um método de tunelamento de pacotes do protocolo IPv4. O trabalho de C. Perkins [97] apresenta um modo de reduzir o tamanho do cabeçalho interno usado em [96] quando o datagrama original não é fragmentado. O método de tunelamento genérico em [98] pode ser usado para encapsular pacotes IPv4 ou IPv6 dentro do pacote IPv6.

A técnica de tunelamento IP-sobre-IP pode ser usada para criar uma região de agregação de fluxos. Desta forma, os fluxos de múltiplas fontes que chegam ao roteador de entrada da região de agregação são encaminhados, de uma forma agregada, para o roteador de saída da região de agregação em um túnel IP-sobre-IP. Assim, todos os roteadores internos à região de agregação processam um fluxo agregado ao invés de processar os fluxos individualmente. Baseado neste princípio, o trabalho de Terzis et al. [5] implementa um mecanismo de agregação de estados do protocolo RSVP sobre túneis para fluxos ponto-a-ponto.

O mecanismo de túneis RSVP está representado na Figura 4.1, onde a fonte e o receptor de uma sessão RSVP individual (fim-a-fim) estão conectados pelo túnel. O roteador $R_{entrada}$ é um roteador de entrada do túnel que encapsula e encaminha os datagramas para o roteador R_{saida} através dos roteadores interiores (ou intermediários). O roteador R_{saida} é o ponto final do túnel que recebe, desencapsula e encaminha os datagramas para o destino baseado no cabeçalho original de cada pacote. No mecanismo existem duas sessões diferentes, uma sessão RSVP túnel e uma sessão RSVP fim-a-fim. A sessão RSVP fim-a-fim estabelece uma ligação com reserva de recursos entre a fonte e o receptor. A sessão RSVP túnel localizada entre o ponto $R_{entrada}$ e o ponto R_{saida} fornece reserva de recursos para um conjunto de sessões RSVP individuais configuradas no túnel.

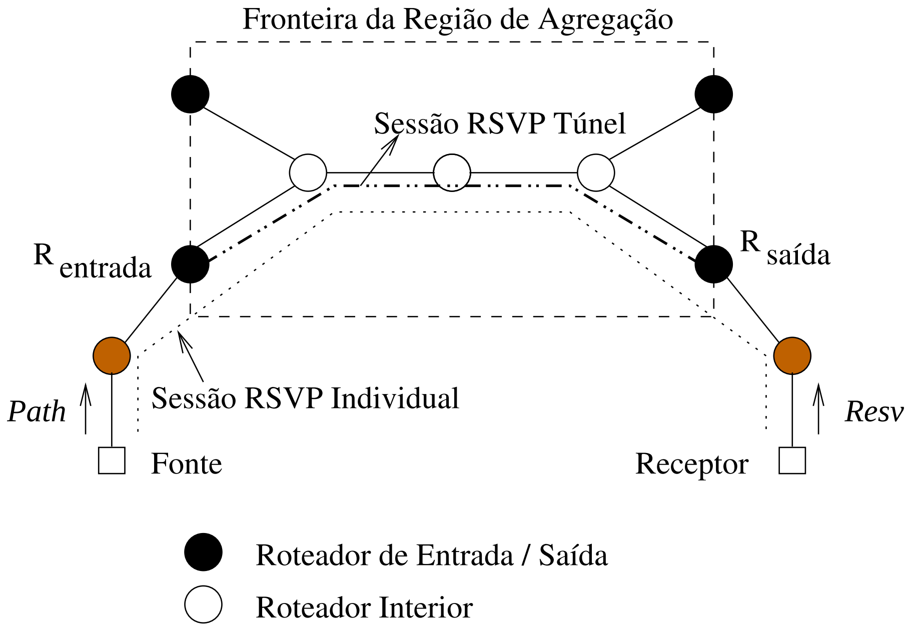

Figura 4.1: Modelo de túneis RSVP ponto-a-ponto.

As mensagens *Path* e *Resv* de uma sessão fim-a-fim, que chegam nos pontos finais do túnel, são mapeadas para uma sessão túnel correspondente a uma classe de tráfego agregada. Elas são encapsuladas e desencapsuladas na mesma maneira que pacotes IP, sendo acrescidas do cabeçalho externo especificando o ponto de entrada do túnel como fonte e o ponto de saída como destino. A Figura 4.1 mostra o mapeamento de apenas uma sessão individual, mas podem existir outras sessões fim-a-fim associadas à sessão RSVP túnel.

Os roteadores interiores não conhecem as mensagens RSVP fim-a-fim, e manipulam somente as mensagens RSVP geradas pelos pontos finais do túnel. Portanto, existe uma redução de estados nos roteadores interiores, que gerenciam somente sessões agregadas por classe de tráfego. Se existir somente uma sessão RSVP túnel configurada sobre um túnel, então todas as sessões RSVP fim-a-fim estarão limitadas para esta sessão túnel configurada. Entretanto, quando mais do que uma sessão RSVP túnel está em uso sobre um túnel IP, haverá a necessidade de um mecanismo de associação entre a(s) sessão(ões) fim-a-fim e as sessões RSVP túneis configuradas sobre o túnel IP. Esta associação entre as sessões individuais e a sessão RSVP túnel é feita através do objeto SESSION_ASSOC, inserido na mensagem *Path* individual no roteador $R_{entrada}$ ligando uma sessão individual a uma sessão túnel. O objeto SESSION_ASSOC contém informações sobre a sessão individual (endereço destino, protocolo de transporte e número da porta de destino) e informações sobre a sessão túnel (endereço e porta do roteador de entrada do túnel).

O roteador de entrada e o roteador de saída devem concordar com estas associações para que as mudanças no estado de reserva original possam ser corretamente mapeadas em mudanças no estado de reserva do túnel, e que erros relatados pelos roteadores interiores para os pontos finais do túnel possam ser corretamente transformados em erros relatados pelos pontos finais para a sessão RSVP fim-a-fim.

A decisão de quais sessões individuais serão mapeadas para sessões túneis é uma questão de implantação de políticas pelos gerentes de rede. Entretanto, observa-se que, se limitarmos ao simples mapeamento de todas as sessões fim-a-fim de uma mesma classe de tráfego para uma única sessão túnel, não existe a necessidade do objeto SESSION_ASSOC.

Para padronização no texto, as mensagens RSVP individuais serão representadas em itálico (mensagens *Path* e *Resv*). As mensagens RSVP geradas pelos roteadores de entrada e saída do túnel que conduzem informações dos fluxos agregados serão escritas em letras maiúsculas seguida da palavra túnel (por exemplo, mensagem PATH túnel).

O procedimento de configuração de reservas de recursos no túnel está especificado pelo diagrama de seqüência de mensagens na linguagem UML (*Unified Modeling Language*) [99] (Figura 4.2) e é executado através dos seguintes passos:

1. o roteador $R_{entrada}$ recebe uma mensagem *Path* individual e cria o estado do caminho (*Path*) da sessão RSVP fim-a-fim;

2. o roteador $R_{entrada}$ envia a mensagem *Path* fim-a-fim encapsulada sobre o túnel para o roteador R_{saida};

3. o roteador R_{saida} recebe a mensagem, desencapsula, cria o estado do caminho da sessão RSVP fim-a-fim e encaminha a mensagem original ao destino;

4. quando uma mensagem *Resv* fim-a-fim correspondente à sessão RSVP fim-a-fim chega no roteador R_{saida}, o roteador cria o estado de reserva, encapsula e envia a mensagem para o roteador de entrada;

5. o roteador $R_{entrada}$ recebe a mensagem *Resv*, desencapsula, cria o estado de reserva e mapeia a sessão RSVP fim-a-fim para uma sessão RSVP túnel;

6. após o mapeamento, o roteador $R_{entrada}$ envia uma mensagem PATH túnel correspondente à sessão RSVP túnel. A mensagem é processada em cada roteador interior, criando o estado do caminho da sessão túnel. O roteador $R_{entrada}$ também transmite uma mensagem *Path* fim-a-fim encapsulada contendo o objeto SESSION_ASSOC com as informações da associação entre a sessão RSVP fim-a-fim e a sessão RSVP túnel;

7. o roteador R_{saida} recebe a mensagem PATH túnel e envia uma mensagem RESV túnel com o objeto RESV_CONFIRM para o interior da região de agregação, solicitando reserva de recursos;

8. o roteador R_{saida} recebe a mensagem *Path*, registra a associação e remove o objeto antes de encaminhar a mensagem para o destino;

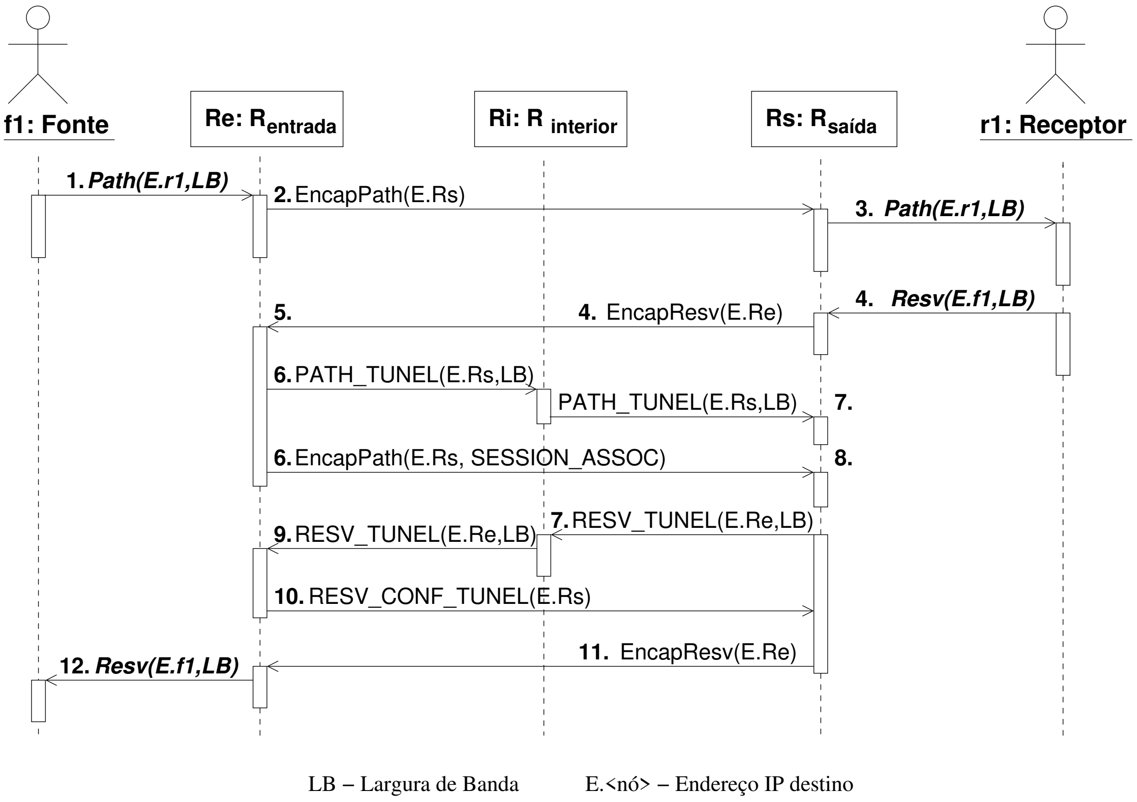

Figura 4.2: Diagrama de seqüência de mensagens RSVP túneis.

9. quando a mensagem RESV túnel chega no roteador $R_{entrada}$, significa que o pedido foi aceito pelos roteadores interiores. Se algum roteador interior não aceitar a reserva, então ele envia uma mensagem RESV_ERROR túnel para o roteador de saída;

10. depois de receber a mensagem RESV túnel, o roteador $R_{entrada}$ envia uma mensagem RESV_CONFIRM túnel para o roteador R_{saida} como confirmação da reserva no túnel;

11. o roteador R_{saida} recebe a mensagem RESV_CONFIRM túnel e envia a mensagem *Resv* encapsulada para o roteador de entrada;

12. o roteador $R_{entrada}$ recebe a mensagem *Resv* encapsulada, desencapsula, e encaminha a mensagem *Resv* original para a fonte.

Após a configuração da reserva, o roteador de saída recebe periodicamente as mensagens *Resv*. Entretanto, pode ocorrer uma mudança nos parâmetros do estado da reserva individual e da reserva da sessão túnel na chegada de uma mensagem

Resv. Se a mudança aumenta a reserva, então o roteador R_{saida} envia uma mensagem RESV túnel, incluindo o objeto RESV_CONFIRM, e o processamento é igual ao do recebimento de uma mensagem *Resv* de uma nova sessão fim-a-fim. Se a mudança diminui a reserva de recursos, então uma mensagem RESV túnel é enviada sem o objeto RESV_CONFIRM, com a finalidade de atualizar os estados de reservas nos roteadores interiores e nos roteadores de entrada.

Quando o roteador de entrada recebe os pacotes de dados, ele verifica se existe reserva de recursos para o fluxo de dados correspondente à sessão RSVP fim-a-fim contida no pacote. Se não existir reserva para o pacote, então o pacote recebe somente o encapsulamento IP-sobre-IP e enviado para o roteador de saída. Caso tenha reserva para o fluxo, o pacote recebe o encapsulamento IP e o encapsulamento UDP (*User Datagram Protocol*), pois as sessões individuais são diferenciadas pela porta UDP da fonte. Outros componentes da sessão túnel, como endereço IP da fonte e destino, porta UDP destino são iguais para todas as sessões individuais. A porta UDP do destino é atribuída pelo IANA (*Internet Assigned Numbers Authorithy*) de número 363 [6].

A porta UDP fonte é escolhida pelo roteador de entrada quando é estabelecido o estado do caminho inicial para uma nova sessão túnel. A porta UDP fonte associada com a nova sessão é então transportada para o roteador de saída através do objeto SESSION_ASSOC.

4.2 Mecanismo de Agregação *Multicast*

O mecanismo de agregação de estado dos fluxos *multicast* apresentado neste trabalho é aplicado ao serviço de carga controlada e agrega o tráfego em trânsito na região de agregação usando a técnica de tunelamento [100, 101].

O mecanismo é uma extensão do modelo de túneis apresentado em [5] e as mudanças introduzidas são:

- o mapeamento de uma sessão RSVP individual para várias sessões RSVP túneis;

- a difusão de uma mensagem *Path* para vários roteadores de saída;
- a marcação dos pacotes reservados através do campo TOS (*Type of Service*) ou o campo DS (*Differentiated Services*) do cabeçalho externo do pacote IP;
- o uso do endereço *multicast* no cabeçalho externo dos datagramas IP para reduzir a largura de banda na região de agregação;
- a utilização de temporizadores de renovação em etapas na retransmissão de mensagens túneis.

4.2.1 Associação de Sessões

A associação de uma sessão RSVP *multicast* para várias sessões RSVP túneis está ilustrada na Figura 4.3. A figura mostra uma árvore *multicast* composta de uma fonte e quatro receptores (C1,C2 no domínio C e D1, D2 no domínio D), todos pertencentes à sessão *multicast*. No exemplo foram criadas duas sessões RSVP túneis localizadas entre o roteador $R_{entrada}$ e os roteadores R_{saida1} e R_{saida2} para fornecer a reserva de recursos em cada túnel para a sessão *multicast* RSVP fim-a-fim.

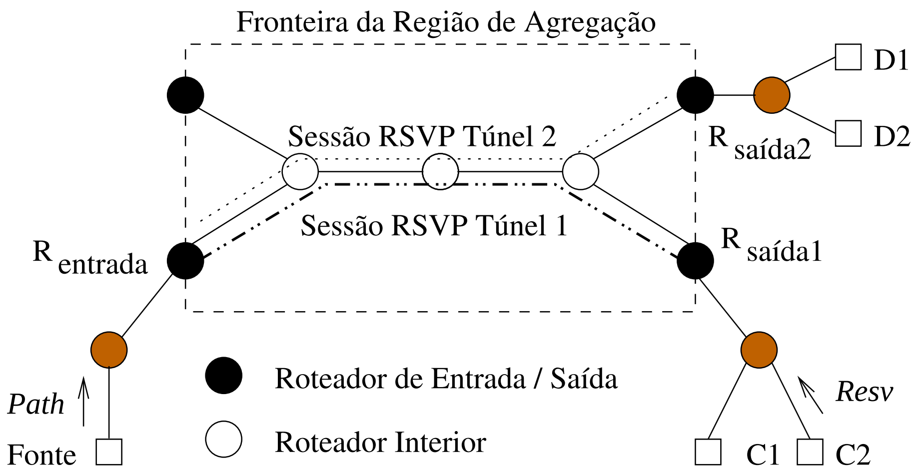

Figura 4.3: Mecanismo de túneis RSVP *multicast*.

O roteador $R_{entrada}$ recebe uma mensagem *Path*, verifica através de um protocolo interdomínio como o protocolo BGP (*Border Gateway Protocol*) [102] para qual roteador ou quais roteadores de saída a mensagem *Path* encapsulada será encaminhada. Tendo o conhecimento do(s) roteador(es) de saída, o roteador de entrada inicia o processo de configuração da reserva em cada túnel composto pelas seguintes etapas:

1. o roteador $R_{entrada}$ recebe uma mensagem *Path* individual e cria o estado do caminho (*Path*) da sessão RSVP fim-a-fim;

2. o roteador $R_{entrada}$ envia uma mensagem *Path* fim-a-fim encapsulada para o roteador R_{saida1} e para o roteador R_{saida2};

3. para cada sessão RSVP túnel são executados passos 3 a 12 descritos na Seção 4.1 para configuração da reserva de recursos entre pontos finais do túnel.

O mapeamento entre sessões RSVP túneis e sessões RSVP individuais é a parte mais complexa do mecanismo, pois tem que processar a inclusão, remoção e atualização dos estados e requer estruturas de dados dinâmicas que serão explicadas na Seção 5.2 do Capítulo 5.

4.2.2 Marcação de Pacotes

Os pacotes de dados com reservas de recursos são encapsulados pelo protocolo IP e pelo protocolo UDP no roteador de entrada [5]. O uso do encapsulamento UDP aumenta o tempo de processamento nos roteadores. A marcação de pacotes processada diretamente na camada de rede é mais rápida do que na camada de transporte.

Assim, os pacotes de dados referentes à sessão *multicast* que chegam no roteador $R_{entrada}$ e que possuem reservas de recursos associadas às duas sessões túneis podem ser marcados com a classe de tráfego (número identificador do túnel) no campo TOS (*Type of Service*) do cabeçalho externo do pacote IP, ou pelo campo DS (*Differentiated Services*). O campo DS é obtido pela renomeação do campo TOS, no caso do IPv4, ou do campo *Traffic Class*, no caso do IPv6 [103]. A utilização do campo DS é para manter a compatibilidade com pedido de reservas que usam Serviços Diferenciados. Após a marcação, são geradas cópias (uma para cada túnel), sendo que cada pacote é encapsulado e enviado para o endereço do roteador de saída. Quando os pacotes chegam nos roteadores de saída, eles são desencapsulados e encaminhados para o destino.

O número de cópias do pacote inseridas na região aumenta linearmente com o número de roteadores de saída, aumentando assim a largura de banda consumida pela rede. Uma solução para o problema é usar um endereço *multicast* reservado no campo endereço de destino do cabeçalho externo do pacote IP, em vez de usar o endereço do roteador de saída. O endereço *multicast* representaria um conjunto de roteadores de saída. Cada roteador na região deve ter uma tabela com dois campos: endereço *multicast* reservado e a lista de roteadores de saída. O estado de cada sessão túnel seria mantido. A Figura 4.4 ilustra o caminho percorrido do pacote, só havendo duplicação no roteador que tem duas interfaces.

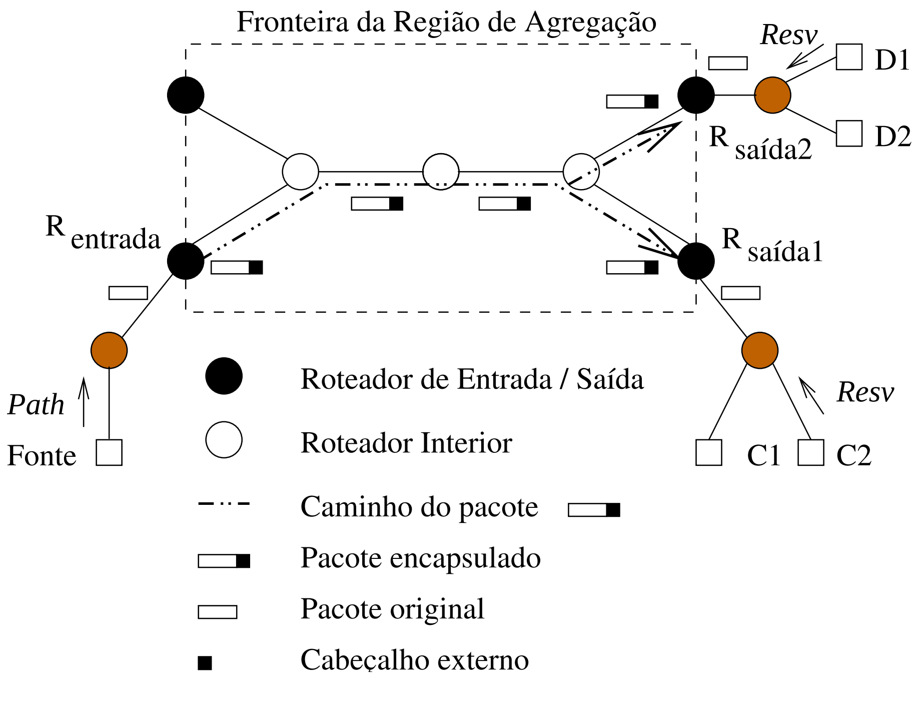

Figura 4.4: Marcação do pacote com endereço *multicast*.

4.2.3 Temporizadores de Renovação em Etapas

O mecanismo *soft-state* envia as mensagens periódicas de renovação para manutenção dos estados de reserva do protocolo RSVP em intervalos que variam de 15s até 45s. O valor normalmente utilizado é de 30s [39]. Este valor é estabelecido no início da configuração da reserva e normalmente permanece fixo durante a duração da sessão RSVP. O envio das mensagens para cada estado de reserva é controlado por um temporizador que é referenciado nesta tese como temporizador de renovação de taxa fixa (temporizador TF).

O mecanismo *soft-state* não fornece confiabilidade na entrega de mensagens e pode causar problemas na ocorrência de perda de mensagens. Se a primeira mensagem *Path* ou *Resv* de um usuário for perdida na rede, uma cópia da mensagem não será retransmitida até o término do intervalo de renovação, ocasionando um atraso de 30s ou mais no estabelecimento de uma reserva.

Uma alternativa para reduzir o tempo de estabelecimento de reserva seria diminuir o período de renovação das mensagens. Entretanto, esta abordagem aumenta o controle de tráfego associado com cada fluxo, com isso aumentando o processamento de mensagens no roteador. Uma melhor solução para reduzir o tempo e tornar confiável o processo de estabelecimento de reservas na presença de perdas de mensagens RSVP, é a utilização de temporizadores de renovação em etapas (*staged refresh timers*) proposto por Pan e Schulzrinne [18].

Temporizadores de renovação em etapas (temporizadores TE) controlam o intervalo de retransmissão de mensagens e iniciam o processo com um valor inicial pequeno no tempo de retransmissão e vão aumentando gradativamente este valor até alcançar um limite. O diagrama de transição de estados apresentado na Figura 4.5 descreve o processo de retransmissão em etapas das mensagens de controle do temporizador. Após enviar uma mensagem *Path* inicial, um nó RSVP entra no estado *Espera_Ack1*, inicializa o temporizador e espera um período inicial de 3s. Se após este tempo uma mensagem de reconhecimento *PathAck* não foi recebida, o nó retransmite a mensagem *Path*. A transição entre os estados é composta de duas partes: evento de ativação e as ações executadas pelo estado. No diagrama, o evento está separado das ações através de uma barra ("/"). A separação entre as ações está representada pelo ponto e vírgula.

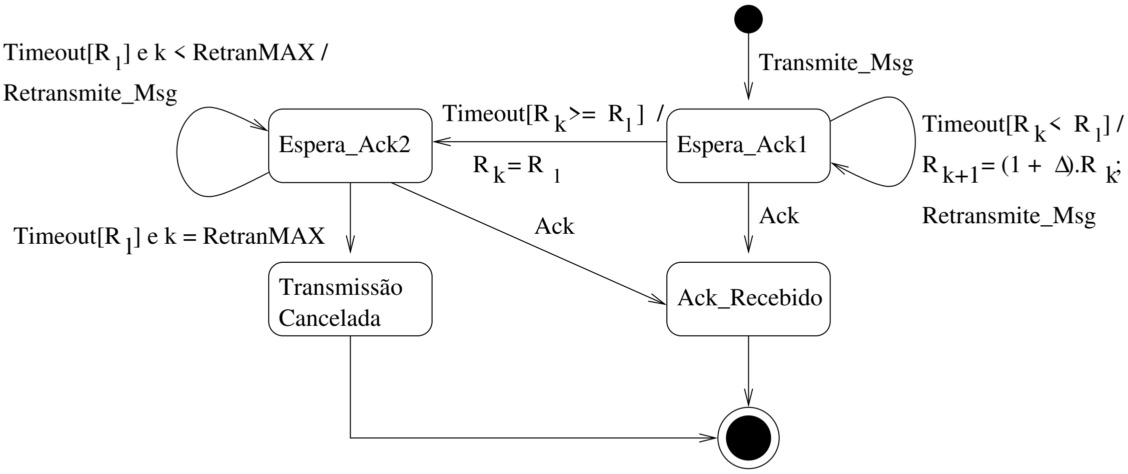

Figura 4.5: Diagrama de transição de estados do temporizador na retransmissão em etapas.

Para ser adaptativo em diversas situações de congestionamento, o valor do intervalo de retransmissão R_k deve ser multiplicado pelo fator $(1 + \Delta)$ a cada retransmissão de uma mensagem *Path*, sendo k o número de retransmissões. O termo Δ é um valor incremental usado para gradualmente aumentar o intervalo de retransmissão. A retransmissão em etapas continuará até que uma mensagem *PathAck* seja recebida ou o intervalo de retransmissão alcance um valor limite R_l. Quando o intervalo alcança o valor R_l, as retransmissões ocorrerão dentro deste valor e o temporizador passa para o estado *Espera_Ack2*. Os valores normalmente usados para Δ e R_l são respectivamente 0,3 e 30s [18, 19].

As retransmissões serão interrompidas quando da chegada de uma mensagem de reconhecimento ou quando as retransmissões alcançarem um valor máximo *Retran-MAX* especificado pela aplicação. O mesmo processo se aplica na transmissão de mensagens *Resv*.

Os temporizadores de renovação em etapas fornecem confiabilidade através de mensagens de reconhecimento *PathAck* e *ResvAck*. Entretanto, no mecanismo RSVP

Túnel uma mensagem RESV túnel representa um reconhecimento de uma mensagem PATH túnel e uma mensagem RESV_CONFIRM túnel representa um reconhecimento de uma mensagem RESV túnel. Portanto não necessita incluir novas mensagens no mecanismo para ter confiabilidade.

O mecanismo de temporizadores em etapas é implementado nos roteadores de fronteira para retransmissão de mensagens PATH túnel e mensagens RESV túnel. Quando o roteador $R_{entrada}$ envia uma mensagem PATH túnel, ele inclui um objeto PATH_CONFIRM túnel na mensagem e na lista de espera de mensagens RESV túnel. O objeto contém o endereço do roteador de entrada, o número do objeto e a hora de transmissão da mensagem (utilizada para computar o tempo de estabelecimento da sessão). Existe um temporizador de renovação em etapas associado a cada objeto. Caso não receba a mensagem RESV túnel com o objeto correspondente, uma mensagem PATH túnel é retransmitida após um período de 3s e o temporizador é configurado com um novo valor do intervalo de retransmissão.

Quando o roteador R_{saida} envia uma mensagem RESV túnel, ele inclui um objeto RESV_CONFIRM túnel na mensagem e na lista de espera de mensagens RESV_CONFIRM túnel. O objeto contém o endereço do roteador de entrada e o número do objeto extraídos do objeto PATH_CONFIRM túnel. Existe um temporizador de renovação em etapas associado a cada objeto. Quando o roteador de saída receber a mensagem RESV_CONFIRM túnel, ele pesquisa na lista de espera a existência do número do objeto. Se o objeto for encontrado, o processo de retransmissão é interrompido para este objeto e o objeto é removido da lista. Caso não receba a mensagem RESV_CONFIRM túnel, uma mensagem RESV túnel é retransmitida após um período de 3s e o temporizador é configurado com novo valor de retransmissão. O processo continua até o recebimento da mensagem RESV_CONFIRM.

Capítulo 5

Simulação do Mecanismo *Multicast*

Este capítulo apresenta a modelagem, a programação e os resultados obtidos por simulações do mecanismo de agregação *multicast*. Os objetivos das simulações são:

- avaliar o nível de agregação alcançado pelos roteadores de fronteira na região de agregação;
- avaliar o desempenho entre o uso de temporizadores de renovação de taxa fixa e temporizadores de renovação em etapas (*staged refresh timers*) na presença de perda de mensagens RSVP túneis.

Para obtenção das medidas foi utilizado o simulador de redes de computadores *ns* (*Network Simulator*) [104]. Uma extensão ao pacote básico do *ns* foi implementada para fornecer o suporte ao mecanismo de agregação sendo constituída de dois módulos: RSVP clássico (fluxos individuais) e RSVP túnel (fluxos agregados). O módulo RSVP clássico foi obtido do trabalho de M. Greis [105].

O módulo RSVP túnel desenvolvido executa as seguintes funções:

- processamento das mensagens PATH_Tunel e RESV_Tunel;
- processamento da mensagem RESV_CONFIRM_Tunel;
- mapeamento das sessões individuais para sessões túneis;

- encapsulamento e desencapsulamento dos pacotes;
- manutenção das tabelas de estados agregados;
- retransmissão de mensagens usando *staged refresh timers*.

5.1 Modelagem do RSVP Túnel

Para a especificação do módulo RSVP Túnel foi utilizada a linguagem UML. A linguagem UML é uma linguagem gráfica para visualização, especificação, construção e documentação de componentes de um sistema orientado a objeto. Foram utilizados na modelagem 3 (três) tipos de diagramas: o diagrama de classes, o diagrama de colaboração e o diagrama de seqüência de mensagens.

A Figura 5.1 apresenta o diagrama de classes do módulo RSVP Túnel. O diagrama mostra a modelagem da visão estática do RSVP Túnel através de um conjunto de classes e seus relacionamentos. O diagrama apresenta algumas classes do simulador *ns* que servem de base para desenvolver as classes do módulo RSVP Túnel. As classes *Connector*, *Agent*, *Node* e *SALink* pertencem a biblioteca de classes do simulador *ns*.

As topologias são criadas no *ns* através de nós (Classe *Node*) e da conexão destes nós por enlaces (Classe *SALink*). Os enlaces no *ns* são compostos por objetos chamados conectores pertencentes a Classe *Connector*. Os conectores apenas geram dados para um receptor. Um enlace básico é criado com características específicas de banda passante e atraso.

Os agentes do *ns* (Classe *Agent*) representam pontos onde os pacotes são gerados ou consumidos e são usados para a implementação de protocolos de várias camadas. Geralmente, um usuário querendo criar uma nova fonte ou receptor de pacotes da camada de rede deve criar uma classe derivada da classe *Agent*. A principal classe no diagrama é a classe *RSVPTunel* (sub-classe da classe *Agent*). Os objetos desta classe executam o mecanismo de agregação.

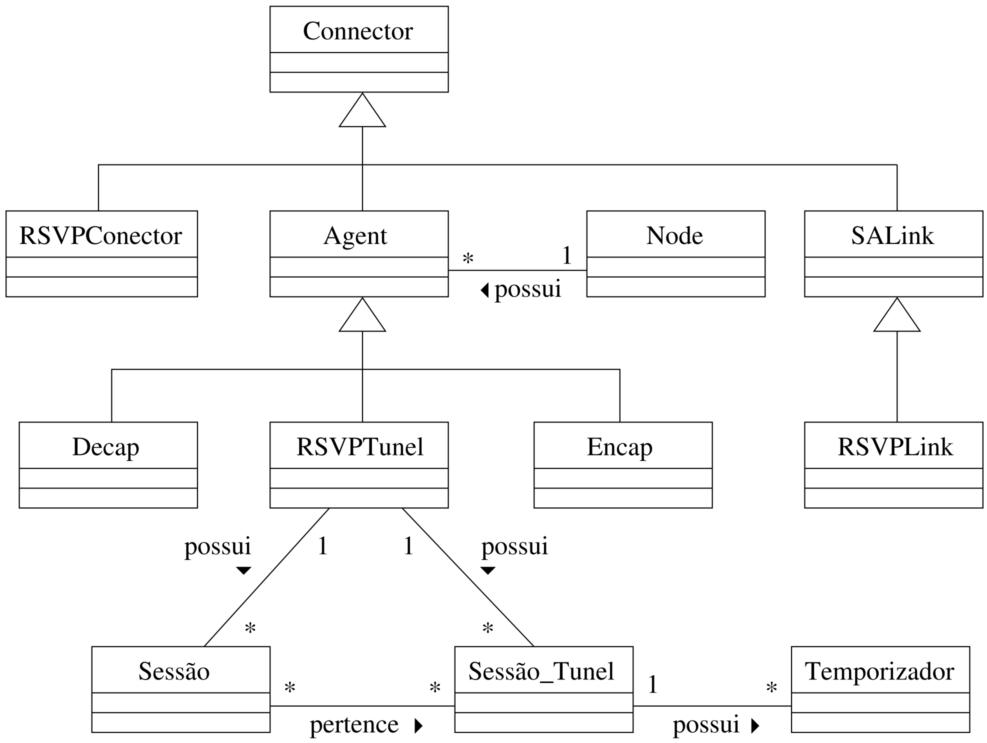

Figura 5.1: Diagrama de classes do módulo RSVP Túnel.

A classe *RSVPTunel* possui um relacionamento de um para muitos (1 para *) com a classe *Sessão* (uma instância representa uma sessão RSVP individual) e a classe *Sessão_ Tunel* (uma instância representa uma sessão RSVP túnel). Várias instâncias da classe *Temporizador* estão associadas à classe *Sessão_ Tunel* para manutenção dos estados de reserva e retransmissão de mensagens.

A classe *RSVPConector* seleciona os pacotes de dados e de controle chegando no nó e encaminha-os para outras classes. As classes *Encap* e *Decap* são sub-classes da classe *Agent*. A classe *Encap* realiza o encapsulamento IP dos pacotes na entrada do túnel e a classe *Decap* executa o desencapsulamento dos pacotes na saída do túnel.

Um enlace entre dois nós (roteadores) com reservas de recursos é especificado pela classe *RSVPLink*. A classe *RSVPLink* é descrita no trabalho de M. Greis [105]. Um objeto da classe *RSVPLink* configura o enlace através dos seguintes parâmetros: a largura de banda do enlace, o atraso do enlace, a porção da largura de banda do enlace que pode ser reservada pelo RSVP, a largura de banda reservada para mensagens RSVP, o tamanho da fila para classe *best-effort* e o tipo de controle de

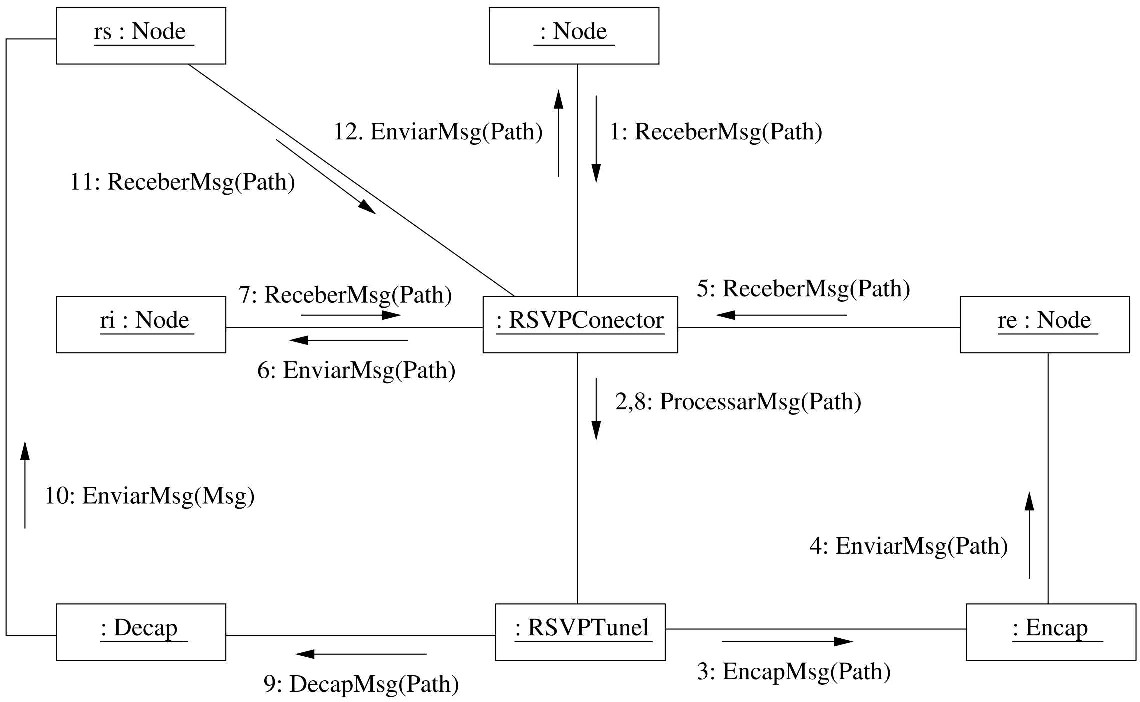

Figura 5.2: Diagrama de colaboração do módulo RSVP Túnel para o processamento da mensagem *Path*.

admissão. A classe *RSVPLink* herda os métodos da classe *SALink*. A classe *SALink* também especifica as operações do controle de admissão.

As especificações das principais classes são apresentadas no Apêndice B.

O módulo RSVP Túnel executa o procedimento de configuração de reservas de recursos no túnel e a modelagem da dinâmica deste processo é representado pelo diagrama de seqüência de mensagens e pelo diagrama de colaboração. O diagrama de seqüência está ilustrado na Seção 4.1(Figura 4.2). Este diagrama apresenta a seqüência completa de transmissão de mensagens RSVP individuais e mensagens RSVP túneis na configuração de uma reserva considerando apenas o uso do objeto *Roteador* no processamento das mensagens RSVP na região de agregação. Entretanto, existem outros objetos envolvidos neste processo que estão representados no diagrama de colaboração.

A Figura 5.2 apresenta os relacionamentos entre os objetos *ns* do módulo RSVP Túnel através de um diagrama de colaboração para o processamento de mensagens

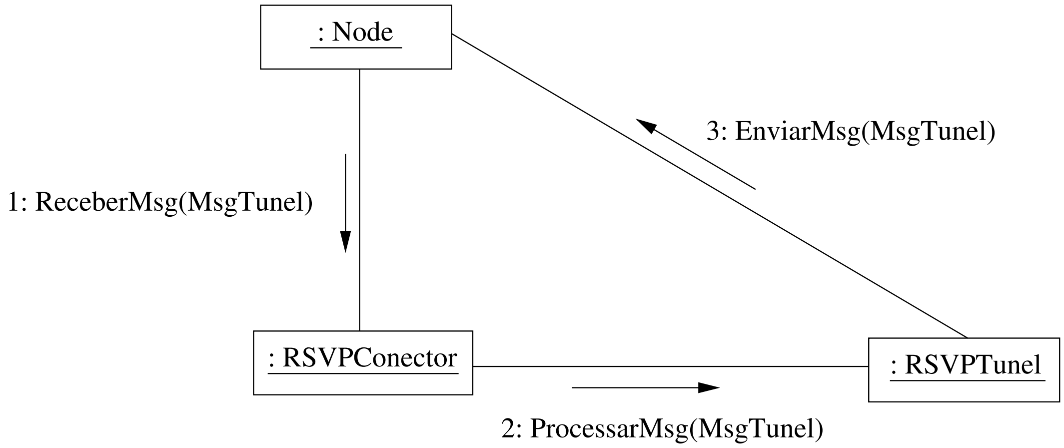

Figura 5.3: Diagrama de colaboração do módulo RSVP Túnel para o processamento de mensagens RSVP túneis.

Path no roteador de entrada e no roteador de saída da região de agregação. O diagrama de colaboração dá ênfase à organização dos objetos que participam desta interação. Conforme mostra a Figura 5.2, o diagrama é formado por objetos que participam da interação como vértices de um grafo. Neste diagrama estão representados os vínculos que conectam esses objetos como arcos de um grafo e também as mensagens que os objetos enviam e recebem. Este diagrama fornece uma indicação visual do fluxo de controle no contexto da organização estrutural dos objetos.

Para indicar a ordem temporal de execução das mensagens, existe um número de seqüência associado a cada mensagem. No diagrama o objeto *RSVPConector* tem a função de interceptar mensagens RSVP individuais (por exemplo, a primeira mensagem é "1: ReceberMsg(Path)" da fonte) na entrada da região para serem processadas pelo objeto *RSVPTunel*. Após o processamento, a mensagem é encaminhada para o encapsulador (objeto *Encap*) e posteriormente encaminhada para o objeto *re* (roteador de entrada) da classe *Node*. No interior da região o objeto *RSVPConector* intercepta as mensagens *Path* e envia para o objeto *ri* (roteador interior). O objeto *RSVPTunel* não processa mensagens individuais no interior da região.

O diagrama de colaboração (Figura 5.3) mostra os relacionamentos entre os objetos *ns* para o processamento de mensagens túneis. O objeto *RSVPConector* intercepta mensagens RSVP túneis oriundas de um objeto da classe *Node* (algum nó que pertence à região de agregação ou é externo à região) e encaminha para o objeto *RSVPTunel*.

5.2 Implementação do RSVP Túnel

Na implementação do módulo RSVP Túnel foram desenvolvidas os seguintes agentes:

- o agente *RSVPTunel*, que executa o mecanismo de agregação;
- o agente *RSVPConector*, que seleciona os pacotes de dados e de controle chegando no nó e encaminha-os para outras classes;
- o agente *Encap*, que realiza o encapsulamento IP-sobre-IP dos pacotes na entrada do túnel;
- o agente *Decap* que executa o desencapsulamento dos pacotes na saída do túnel.

O agente *RSVPTunel* é composto de rotinas obtidas de [105] para processamento de sessões individuais e de rotinas para o processamento do mecanismo de agregação. A maioria das rotinas do RSVP clássico foi mantida com o código original, exceto as rotinas que processam as mensagens *Path* e *Resv* no momento da chegada na região de agregação e as rotinas de envio destas mensagens para o interior da região. A implementação do processamento das sessões RSVP individuais está baseada na especificação funcional e nas regras de processamento de mensagens do protocolo RSVP [39, 57].

O agente *RSVPTunel* mantém os estados das sessões RSVP individuais e estados das sessões RSVP túneis nos roteadores de entrada e saída da região, e mantém os estados das sessões RSVP túnel nos roteadores interiores. Processa as mensagens RSVP individuais chegando na entrada e saída da região, bem como as mensagens

PATH_Tunel, RESV_Tunel e RESV_CONFIRM_Tunel no interior e na saída da região.

Para manutenção dos estados individuais e agregados do protocolo RSVP são usadas listas encadeadas. Uma lista é composta de registros. Segundo a terminologia contida na especificação do protocolo RSVP, um registro é chamado de bloco de estado (*State Block*) quando ele armazena um conjunto de informações sobre o estado do caminho, estado da reserva ou controle de tráfego de uma sessão RSVP. As listas para os estados individuais são as seguintes (Figura 5.4):

- SESSÃO - contém basicamente um identificador da sessão, endereço destino, ponteiro para a próxima sessão da lista, e ponteiros para as listas de PSBs, RSBs e TCSBs;
- PSB (*Path State Block*) - armazena informações da mensagem *Path*, como endereço da fonte, especificação de tráfego (taxa do *token bucket* e tamanho do *bucket*) e endereço do roteador anterior (*previous hop*) que enviou a mensagem *Path*;
- RSB (*Reservation State Block*) - armazena informações da mensagem *Resv*, como endereço do receptor e especificação da reserva (taxa do *token bucket* e tamanho do *bucket*) e estilo de reserva;
- TCSB (*Traffic Control State Block*) - armazena a resultante da junção (*merge*) das reservas de vários RSBs aplicadas a ume determinada interface de saída.

Para controle das sessões que devem ser renovadas (*refreshed*), existe uma lista de temporizadores (lista_timers). Esta lista é classificada em ordem crescente pelo próximo tempo de renovação da sessão.

As sessões RSVP túneis são armazenadas em uma lista encadeada, e cada sessão túnel possui os seguintes registros de estados:

- PSB túnel - armazena a soma das especificações de tráfego das sessões RSVP individuais;

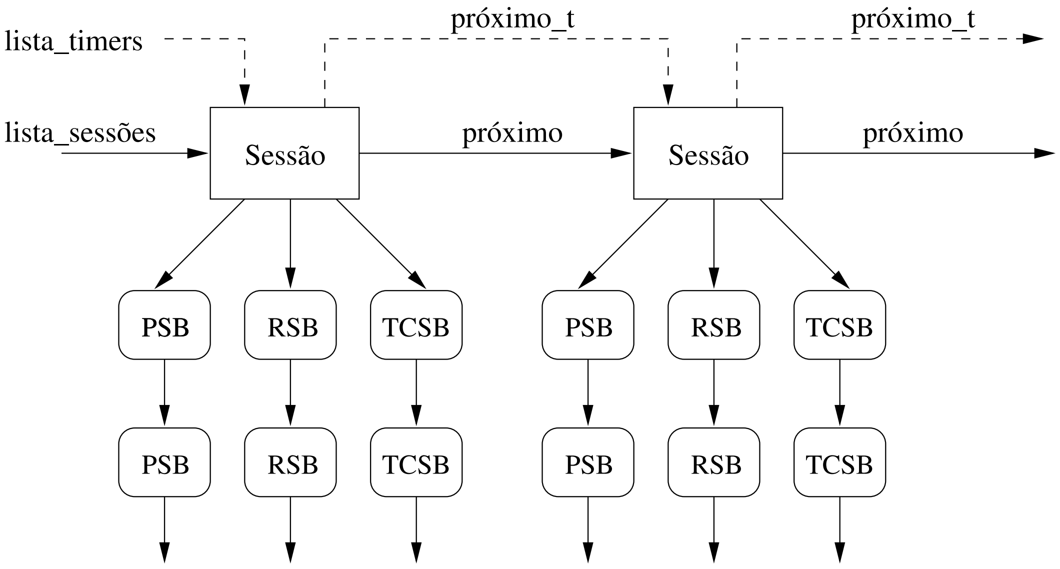

Figura 5.4: Estrutura de dados das sessões individuais.

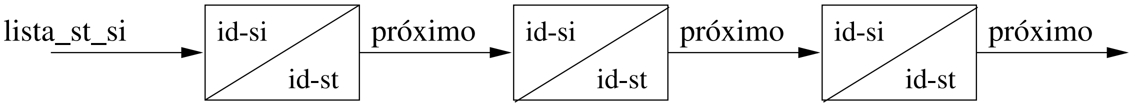

Figura 5.5: Lista com associações entre uma sessão túnel e uma sessão individual.

- RSB túnel - armazena a soma das especificações da reserva das sessões RSVP individuais.

A associação entre sessões RSVP individuais e sessões RSVP túneis está representada pelo relacionamento m:n, isto é, uma sessão RSVP túnel possui uma sessão RSVP individual ou m sessões RSVP individuais e uma sessão RSVP individual pode pertencer a uma ou n sessões RSVP túneis. Uma lista encadeada implementa a associação entre as sessões RSVP túnel e as sessões RSVP individuais. A Figura 5.5 mostra que cada elemento da lista possui um identificador da sessão RSVP túnel (id-st) e um identificador da sessão RSVP individual (id-si).

Os objetos criados na implementação são:

- objeto SESSION_T - contém o endereço do roteador de entrada e o endereço do roteador de saída do túnel;

- objeto SESSION_ASSOC - contém o identificador do fluxo, endereço do grupo *multicast* e endereço do roteador de entrada do túnel;
- objeto PATH_CONFIRM_T - contém o endereço do roteador de entrada do túnel, um número inteiro identificador do objeto e a hora de transmissão da mensagem PATH túnel;
- objeto RESV_CONFIRM_T - contém o endereço do roteador de entrada do túnel e um número inteiro identificador do objeto.

5.3 Nível de Agregação

As simulações realizadas avaliam o nível de agregação, isto é, a comparação entre estados individuais e estados agregados no roteador de fronteira [100].

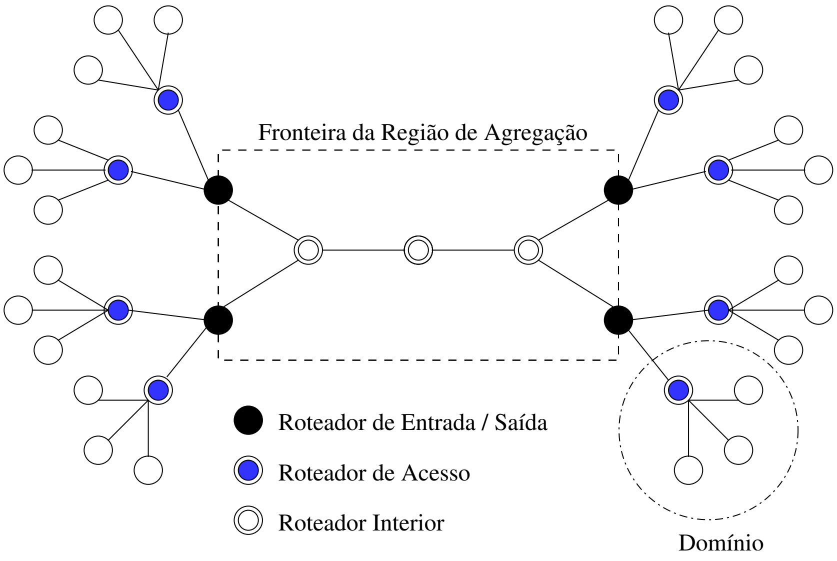

Figura 5.6: Topologia utilizada.

A topologia da simulação está representada na Figura 5.6. Os nós de cor preta representam os roteadores de entrada ou de saída da região de agregação. O conjunto de um nó de acesso ligado aos nós finais (cor branca) representam um domínio que utiliza o protocolo RSVP clássico.

Em cada domínio, os nós finais são receptores e fontes de tráfego para grupos *multicast*. Na simulação, um grupo é composto de uma fonte e até 8 (oito) receptores. Uma sessão *multicast* fica caracterizada pelo par (Fonte, Grupo). Os nós na região de agregação são interligados por enlaces de 155 Mbps, o enlace entre o nó de acesso e o nó de entrada/saída da região tem 50 Mbps. Os nós finais estão conectados por enlaces de 10 Mbps.

Cada nó final envia fluxos CBR (*Constant Bit Rate*) com taxa de 64 kbps. Antes de um fluxo RSVP ser transmitido, as mensagens *Path* são enviadas em intervalos aleatórios. Quando os receptores recebem as mensagens, eles disparam automaticamente um pedido de reserva através da mensagem *Resv*. Cada nó fonte envia uma mensagem *Path* para cada grupo a que pertence, e o número de grupos por fonte pode variar de dois a oito. A fonte, o receptor e o número de receptores por grupo são variáveis aleatórias uniformemente distribuídas. Os receptores de um grupo não podem pertencer ao mesmo domínio da fonte e nem pertencer ao domínio vizinho do domínio da fonte, pois observando a Figura 5.6, os domínios vizinhos estão ligados ao mesmo roteador de entrada/saída e entre os domínios vizinhos não se formam túneis. Foram executadas sessões de simulações com topologias de 31, 39 e 55 nós.

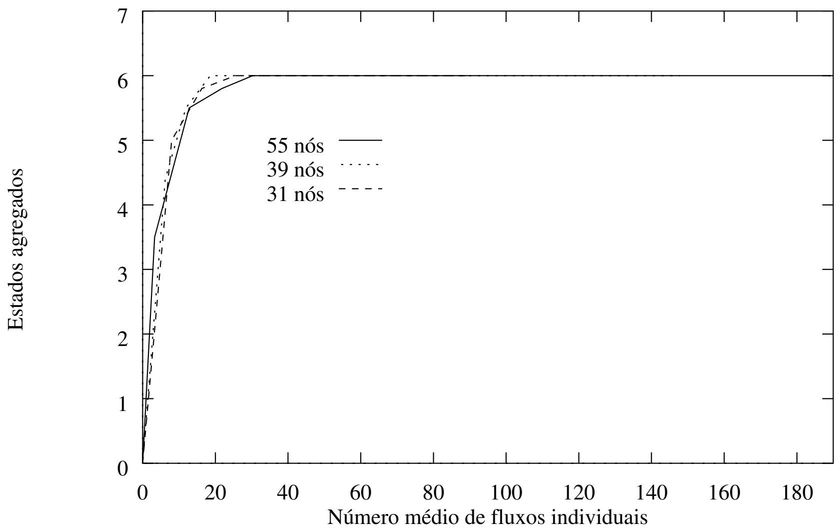

Figura 5.7: Estados agregados X Estados individuais.

A Figura 5.7 apresenta um gráfico que relaciona o número médio de estados agregados (equivalente ao número médio de túneis) e o número médio de fluxos individuais sendo reservados por roteador de entrada/saída. Um fluxo representa uma sessão *multicast*. Observa-se que o número de estados agregados permanece constante a partir de um determinado instante, enquanto fluxos individuais reservados aumentam. Nas topologias simuladas o número máximo de túneis no roteador de entrada/saída é 6 (seis).

Na comparação entre estados individuais e estados agregados é utilizado o Fator Médio de Agregação. O Fator Médio de Agregação é a relação entre o número médio de estados individuais e número máximo de estados agregados (número de túneis) por roteador de entrada/saída. A Tabela 5.1 mostra o valor médio do Fator de Agregação para 3(três) topologias. O número de sessões é o número máximo de sessões configuradas com reservas de recursos na região de agregação.

Tabela 5.1: Fator Médio de Agregação.

Nr total de nós	Nr de nós por domínio	Nr de sessões	Nr médio de estados individuais	Fator médio de agregação
31	3	120	90	15:1
39	4	200	148	24:1
55	6	240	189	31:1

5.4 Estabelecimento de Sessões *Multicast*

Esta seção apresenta uma avaliação do desempenho do mecanismo de agregação no estabelecimento de reserva de recursos dentro da região de agregação através de simulações.

As medidas obtidas foram o tempo médio e o número médio de mensagens RSVP túneis no estabelecimento de uma sessão *multicast* utilizando temporizadores de renovação de taxa fixa e temporizadores de renovação em etapas.

As simulações consistiram de repetidos estabelecimentos de reservas de recursos em grupos *multicast* (um grupo é constituído de uma fonte e de n receptores), sobre vários tamanhos de árvores *multicast* e várias condições de perda de pacotes.

Foram utilizados dois tipos de topologias nas simulações (Figura 5.8): uma árvore binária cheia e uma árvore binária com enlace de gargalo (*bottleneck link*). A árvore binária cheia pode representar um subconjunto de nós da topologia da estrutura principal de roteadores utilizada no MBone, pois a maioria dos roteadores possuem pelo menos duas rotas alternativas para o encaminhamento dos pacotes, fornecendo robustez suficiente em caso de falhas nos enlaces [38].

A árvore binária com enlace de gargalo representa uma situação crítica, onde existem enlaces que concentram a maioria do tráfego em uma rede. Estes enlaces estão localizados próximos da fonte de um grupo *multicast*. No caso da região de agregação, o roteador de entrada pode ser considerado como a fonte geradora de mensagens RSVP, pois é o ponto de entrada do túnel. Além disso, o enlace de gargalo é também considerado um ponto crítico com relação à perdas de mensagens, pois a perda de uma única mensagem prejudica a configuração de reservas dos todos receptores de um grupo *multicast*.

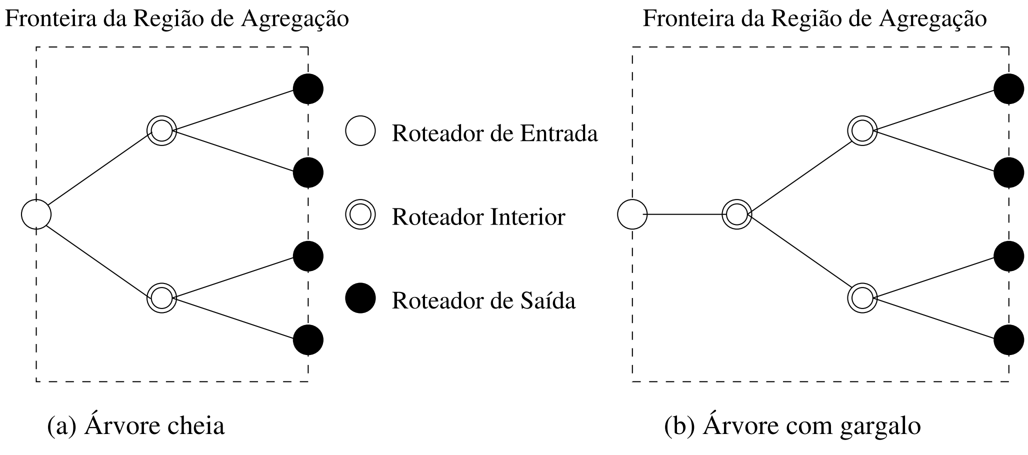

Figura 5.8: Topologias utilizadas nas simulações.

As configurações da árvore binária cheia são árvores com 3, 7 e 15 nós e as configurações da árvore binária com gargalo são árvores com 4, 8 e 16 nós. As Figuras 5.8(a) e 5.8(b) apresentam uma árvore binária cheia com 7 nós e uma árvore binária com enlace de gargalo com 8 nós. O nó raiz é o roteador de entrada e os nós folha são os roteadores de saída. Na topologia estão consideradas implícitas as ligações entre os roteadores de fronteira e roteadores de acesso de outros domínios para recepção de mensagens RSVP individuais.

Para cada configuração, foram estabelecidos 2000 e 3000 fluxos (grupos *multicast*) com reservas de recursos. Não foi introduzido atraso nos enlaces para isolar o tempo de processamento da operação do mecanismo. O tempo médio de duração de cada sessão *multicast* é de 180s com distribuição exponencial. O tempo médio de transmissão entre as mensagens PATH túnel é de 100ms com distribuição exponencial. Estes valores foram escolhidos com o objetivo de injetar um tráfego grande de mensagens de controle em um período de 40 min de simulação de uso da rede. Foram realizadas simulações no máximo para 3000 fluxos em um período de 40 min de uso da rede devido à limitações de memória RAM e capacidade do processador da estação de trabalho em que o simulador era executado. Valores maiores foram testados, entretanto o tempo de execução das simulações eram muitos extensos, ocasionando à interrupção da execução por diversas ocasiões.

No estabelecimento de uma sessão RSVP túnel, uma mensagem RESV túnel é enviada imediatamente após a chegada da mensagem PATH túnel. O valor da taxa fixa de renovação de mensagens R_l é de 30s. Os valores usados para o período inicial de estabelecimento da sessão e o fator Δ são respectivamente 3s e 0,3.

Os dois modelos de perdas de pacotes utilizados nas simulações são: a taxa de perdas por pacotes e perdas periódicas.

O Apêndice A apresenta um exemplo de *script* de simulação com a descrição das etapas para obtenção dos resultados ilustrados nesta seção.

As execuções das simulações foram realizadas em um computador Pentium III 500MHz com 256 Mbytes de memória RAM, um disco rígido de 10 Gigabytes. O sistema operacional usado foi o Linux RedHat 6.2.

5.4.1 Taxa de Perdas por Pacotes

É o percentual de pacotes perdidos em cada direção do enlace com distribuição uniforme. Este modelo mostra o impacto de perdas de pacotes na qualidade de serviço em períodos de longa duração fornecido pelo RSVP Túnel nas árvores *multicast* (Figura 5.8).

Existem dois tipos de enlace quanto à taxa de erro: enlace com taxa baixa de erro (0,1 a 1% de perdas) e enlace com taxa média de erro de pacotes (2 a 5% de perdas). Os valores da taxa média podem representar uma variação de perdas durante uma hora em redes de alta velocidade [91] ou durante um dia no MBone [85].

Para a taxa de perdas foi utilizada a classe *ErrorModel* do simulador *ns*. Esta classe possui três variáveis usadas nas simulações: a variável *unit* que representao tipo de erro (por bits ou por pacotes), a variável aleatória geradora de erros e a variável *rate* que indica o percentual de pacotes perdidos no enlace.

Para cada tipo de topologia foram estabelecidos 2000 e 3000 fluxos (grupos *multicast*) com reservas de recursos.

Atraso Médio para 2000 Fluxos

As Figuras 5.9 e 5.10 mostram o atraso médio de estabelecimento de sessões com diversas taxas de erro por enlace e o ganho em desempenho usando os temporizadores de renovação em etapas (TE) e os temporizadores de renovação de taxa fixa (TF) tanto na árvore binária cheia quanto na árvore binária com enlace de gargalo [106]. O eixo horizontal do gráfico representa o percentual de pacotes perdidos (taxa de erro) por enlace, em ambas direções. Cada atraso médio é resultante de um conjunto de simulações realizadas em uma topologia, onde a mesma taxa de erro é inserida em todos os enlaces da topologia.

Nota-se um atraso médio muito pequeno no estabelecimento de sessões com temporizadores em etapas em enlaces com taxa de 2-5% de perdas. Na árvore binária cheia de 15 nós, o atraso médio máximo é de 1,9s (Figura 5.9). Na árvore com enlace de gargalo de 16 nós, o atraso médio máximo é de 2,4s (Figura 5.10).

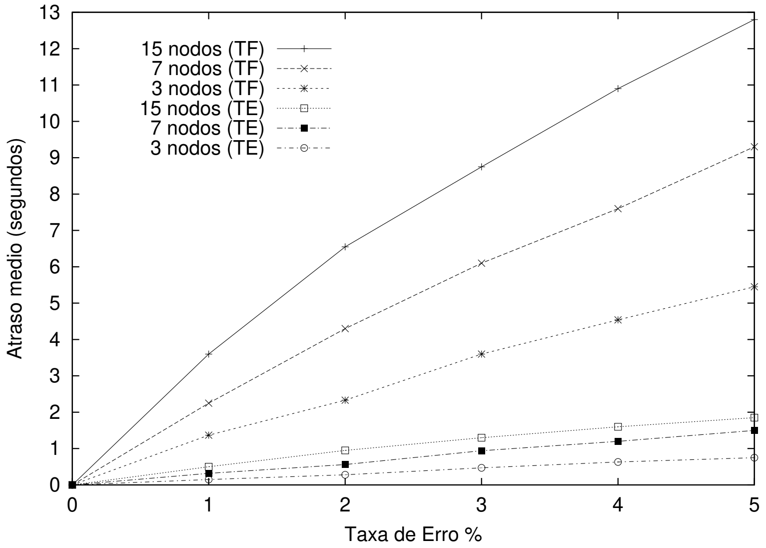

Figura 5.9: Atraso médio na ArvBin cheia com 1 a 5% de perda de pacotes para 2000 fluxos.

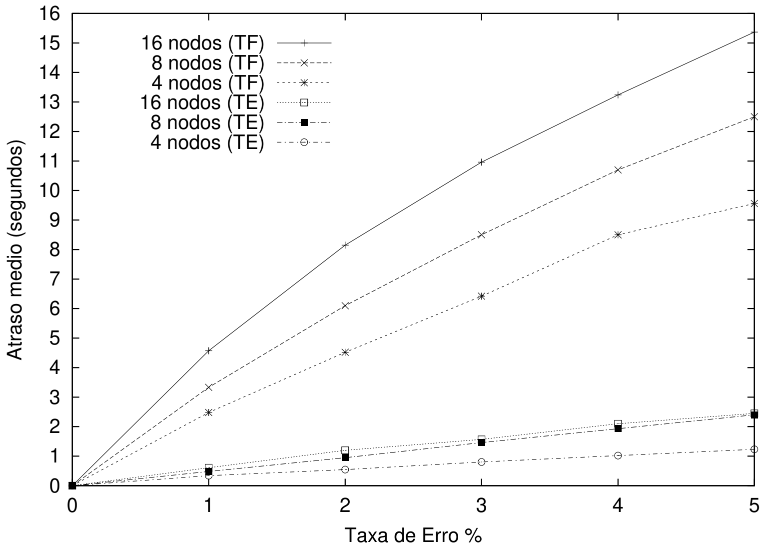

Figura 5.10: Atraso médio na ArvBin com enlace de gargalo com 1 a 5% de perda de pacotes para 2000 fluxos.

Estes resultados podem ser considerados satisfatórios, pois uma reconfiguração de reserva de recursos dentro destes valores não prejudicam as aplicações adaptativas de áudio e vídeo. Estes resultados também estão coerentes com os efeitos da perda de pacotes nas aplicações apresentados na Tabela 3.1, que considera o intervalo de perdas entre 2 e 5% como perdas toleráveis.

As árvores com enlace de gargalo com 8 e 16 nós usando os temporizadores de renovação em etapas apresentaram um tempo de estabelecimento de reservas muito próximos entre 3 e 5%.

Atraso Médio para 3000 Fluxos

As Figuras 5.11 e 5.12 mostram o atraso médio de estabelecimento de sessões com diversas taxas de erro por enlace e o ganho em desempenho usando os temporizadores TF e TE para árvore binária cheia.

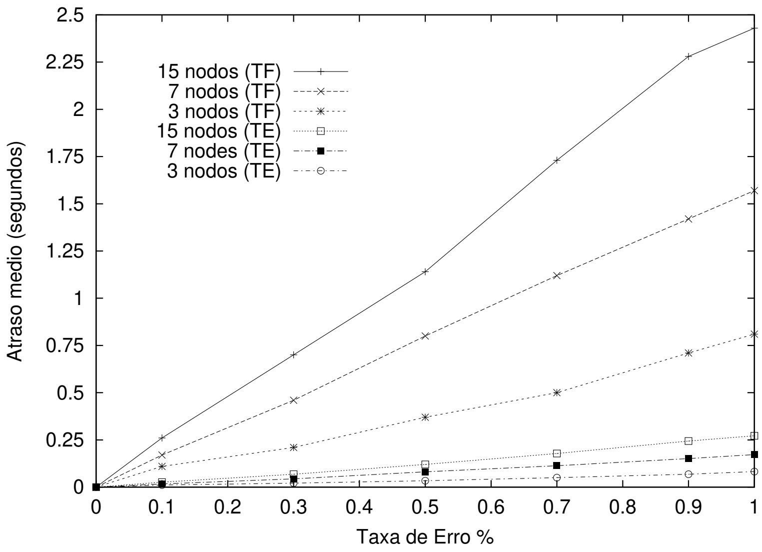

Figura 5.11: Atraso médio na ArvBin cheia com 0,1 a 1% de perda de pacotes para 3000 fluxos.

A Figura 5.11 apresenta resultados para pequenas taxas de erros, representando uma rede bem dimensionada. Os valores obtidos são muitos pequenos para temporizadores TF e para temporizadores TE, e não afetam o desempenho de aplicações multimídias. Por exemplo, para uma árvore de 15 nós com uma taxa de perda de 0,5%, os atrasos obtidos para temporizadores TF e TE foram respectivamente 1,1s

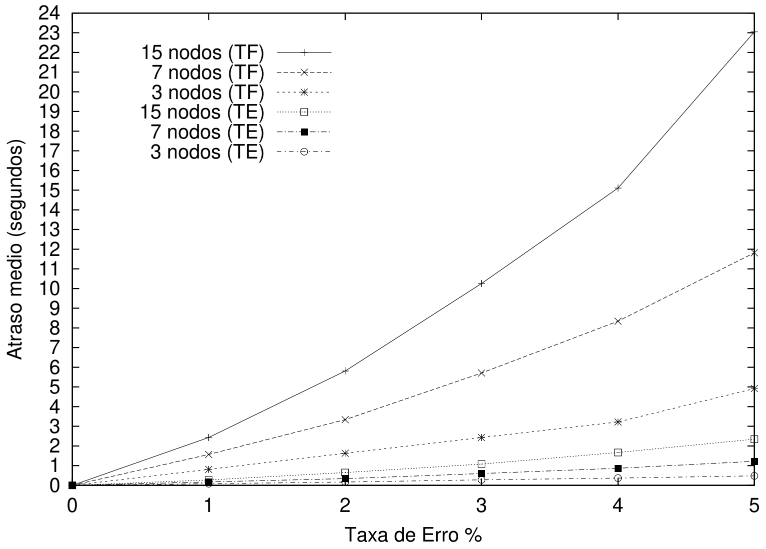

Figura 5.12: Atraso médio na ArvBin cheia com 1 a 5% de perda de pacotes para 3000 fluxos.

e 0,12s. Para o pior caso, a árvore de 15 nós com uma taxa de perda de 1%, o atraso obtido para temporizadores TF e TE foram respectivamente 2,4s e 0,27s. Assim, o ganho no estabelecimento de reservas usando temporizadores TE é bem maior do que utilizando temporizadores TF.

A Figura 5.12 apresenta resultados para taxas no intervalo de 1% até 5% de erros. As três curvas superiores representam as topologias que usam temporizadores de taxa fixa. A árvore de 4(quatro) nós mostra um atraso de 3,2s para 4% de perdas e um valor de 4,9s para 5% de perdas. Estes valores podem prejudicar aplicações multimídias que não toleram atrasos de pacotes. A árvore de 15 nós mostra um atraso de 15,1s para 4% de perdas e um valor de 23,1s para 5% de perdas. Estes valores degradam bastante a qualidade de aplicações de áudio e vídeo.

As três curvas inferiores representam as topologias que usam temporizadores TE. A curva para árvore de 15 nós apresenta um atraso de 1,6s para uma taxa de 4% e um atraso de 2,3s para 5% de perdas de pacotes. Assim, o atraso no estabelecimento de reservas usando temporizadores TE é bem menor do que utilizando temporizadores TF, mostrando um ganho bastante significativo, mesmo para uma perda de 5%.

Número de Mensagens para 3000 Fluxos

Uma outra medida de desempenho obtida foi o número médio de mensagens RSVP túneis necessário para configurar uma reserva recursos para uma sessão *multicast*. Na configuração de uma sessão sem erro são transmitidas três mensagens túneis (PATH túnel, RESV túnel e RESV_CONFIRM túnel). A Figura 5.13 mostra uma melhor eficácia dos temporizadores em etapas quando a taxa de erro aumenta. Por exemplo, para uma taxa de 5% de perdas de pacotes na árvore de 15 nós foram necessárias 4,8 mensagens para configurar uma sessão usando temporizadores TF enquanto com temporizadores TE foram consumidas 4,07 mensagens túneis.

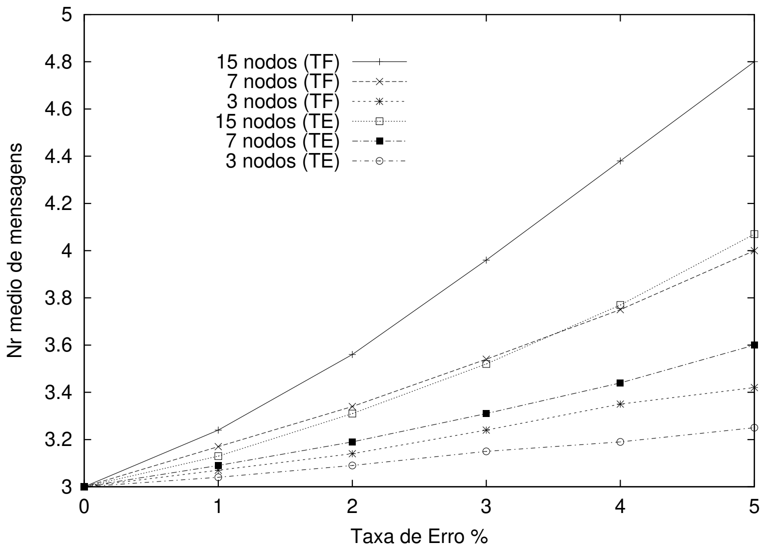

Figura 5.13: Número médio de mensagens de controle na ArvBin cheia com 1 a 5% de perda de pacotes para 3000 fluxos.

5.4.2 Perdas Periódicas

No segundo modelo de perda, nós simulamos o mecanismo na árvore binária *multicast* com perdas periódicas ocorrendo no enlace de gargalo (Figura 5.8.(b)) [107]. Perdas periódicas acontecem em períodos de congestionamento nos roteadores a cada 30s ou 60s, quando uma série de pacotes consecutivos são perdidos. Os períodos de congestionamento ocorrem por causa de atualizações periódicas de roteamento como relatado em [91, 92], isto é, um roteador na atualização da tabela de rotas pode produzir rajadas de mensagens de controle na qual temporariamente sobrecarregam um enlace.

O processo de perda em cada direção do enlace de gargalo é representado pelo modelo de 2-estados. Um dos estados representa períodos de congestionamento (com perdas), enquanto o outro representa períodos sem perdas de pacotes. O processo de perda consome um tempo médio de 250ms com distribuição exponencial no estado de perda. O tempo médio do processo de perda escolhido está na faixa de valores obtidos nas medições dos trabalhos citados na Seção 3.3 sobre duração das rajadas de perdas de pacotes. As perdas ocorrem a cada 30 ou 60 segundos. O processo de perda nos outros enlaces são representados pela taxa de perda de pacotes variando entre 0,1 e 1% com distribuição uniforme.

Para simular perdas periódicas com 2-estados foi utilizado a classe *ErrorModel/MultiState* do simulador *ns*. Para criar um modelo de 2-estados, foram fornecidos para o simulador os seguintes parâmetros: o número de estados, a duração de cada estado, o tipo de distribuição de probabilidade de cada estado e a matriz de transição de estados.

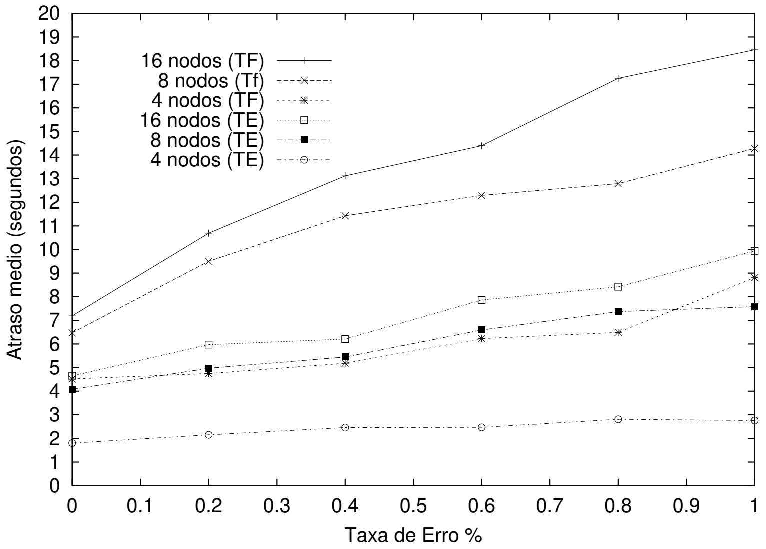

Figura 5.14: Atraso médio na ArvBin com enlace de gargalo com perda de pacotes a cada 30s para 3000 fluxos.

As Figuras 5.14 e 5.15 apresentam o atraso médio de estabelecimento para perdas

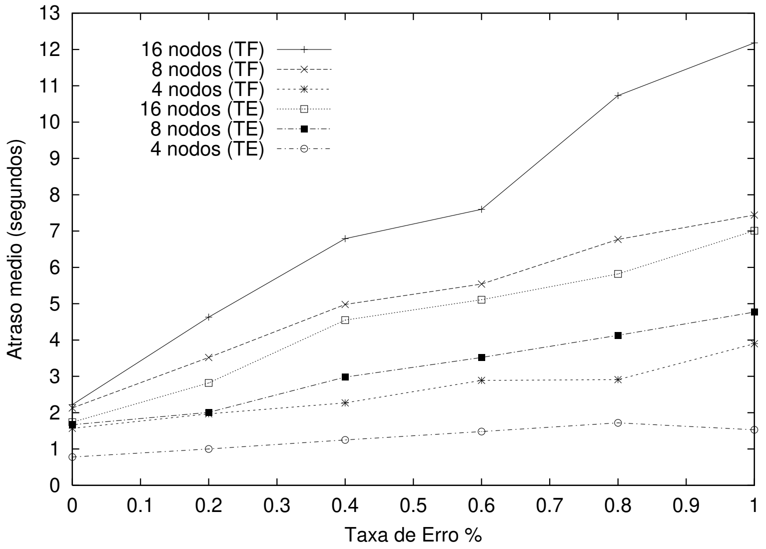

Figura 5.15: Atraso médio na ArvBin com enlace de gargalo com perda de pacotes a cada 60s para 3000 fluxos.

periódicas. A taxa de erro representa a perda de pacotes nos enlaces da árvore (exceto o enlace de gargalo) com taxa baixa de erro (0,1-1%) para cada configuração da árvore *multicast*. O valor zero no eixo x representa perdas de pacotes somente no enlace de gargalo. Mais uma vez, os resultados mostram o ganho no desempenho com temporizadores de renovação em etapas, mesmo na presença de um único ponto de congestionamento.

O ganho em perdas a cada 30s é maior do que em ciclos de 60s. Isto mostra que o ganho aumenta à medida que o congestionamento aumenta. O uso de temporizadores TE na árvore de 4 (quatro) nós apresentou uma pequena variação no atraso em razão da pouca influência dos 2 (dois) enlaces conectados aos nós folha no atraso médio total da topologia.

Finalmente, por alguns resultados apresentados nesta seção, poderia se argumentar que se a taxa de erro fosse muito baixa (menor que 1%) ou com valores entre 1% e 2% de perdas ou com erros ocorrendo apenas no enlace de gargalo, o desempenho dos temporizadores de tempo fixo seria satisfatório, e assim a utilização de temporizadores de renovação em etapas não seria necessária, especialmente para árvores de 3 ou 4 nós. Entretanto observa-se que os resultados consistem de valores médios, medidos sobre um grande número de fluxos e que em alguma ocasião particular, a perda de alguma mensagem de controle no estabelecimento do fluxo é penalizada com um atraso mínimo de 30s com temporizadores TF, mas somente por um atraso mínimo de 3s com temporizadores TE. Este fato provavelmente justifica o uso de temporizadores TE, mesmo quando a taxa de perda de pacotes for muito baixa.

Capítulo 6

Conclusões

Os mecanismos de agregação de fluxos permitem uma redução dos estados necessários para a armazenar as informações sobre as reservas de recursos do protocolo RSVP, e desta forma melhoram sua escalabilidade na *Internet*.

Este trabalho apresentou uma descrição de um mecanismo de agregação de fluxos *multicast* sobre túneis IP-sobre-IP, uma implementação no simulador *ns* e uma avaliação de desempenho quanto à agregação de fluxos e quanto à perda de mensagens de controle na sinalização de qualidade de serviço.

O mecanismo de agregação de fluxos *multicast* sobre túneis IP-sobre-IP reduz o número de estados armazenados e o número de mensagens processadas na região de agregação porque os roteadores interiores manipulam somente as mensagens RSVP agregadas, que são geradas pelos roteadores de fronteira de um domínio de agregação. O fator médio de agregação mostrou uma significativa redução de estados do protocolo RSVP nos roteadores interiores das topologias simuladas.

A utilização de um endereço *multicast* reservado como forma de marcação dos pacotes reservados proporciona um menor consumo de largura de banda na região. Este endereço representa os roteadores de saída para onde o pacote será enviado.

O mecanismo de agregação *multicast* utilizando a abordagem *soft-state* clássica do protocolo RSVP para manutenção de estados de reservas pode apresentar longos atrasos no tempo de estabelecimento de reservas de recursos quando ocorrem perdas de mensagens de controle. Na abordagem *soft-state* o intervalo de tempo de transmissão entre as mensagens de controle é de 30s. A perda de pacotes pode afetar a qualidade de aplicações multimídias sensíveis ao atraso, tais como aplicações de áudio/vídeo. Estas aplicações experimentam degradação na qualidade com o aumento da perda de pacotes e atrasos na rede.

A identificação do nível de perda de pacotes no tráfego com reserva de recursos pelo mecanismo de agregação é importante sob o ponto de vista de gerência de rede, pois o administrador da rede pode avaliar o impacto das perdas no tráfego na região de agregação e utilizar mecanismos para minimizar este impacto nas aplicações multimídias.

Para diminuir o tempo de configuração das reservas e aumentar a confiabilidade no encaminhamento de mensagens de controle, o mecanismo de agregação *multicast* utiliza temporizadores de renovação em etapas.

Para avaliar o impacto de perdas de pacotes foram realizadas simulações para comparar o desempenho do mecanismo de agregação *multicast* usando temporizadores de renovação de tempo fixo e temporizadores de renovação em etapas. As medidas obtidas mostraram uma redução substancial no tempo de estabelecimento de reservas de sessões *multicast* com temporizadores de renovação em etapas na ocorrência de perdas periódicas e perdas com valores médios.

Em decorrência do trabalho de pesquisa desenvolvido e dos resultados obtidos, esta tese apresenta as seguintes contribuições:

- a proposta de um mecanismo de agregação de estados do protocolo RSVP para fluxos *multicast* em túneis IP-sobre-IP. Este mecanismo reduz o número de estados armazenados e o número de mensagens do processadas nos roteadores em relação ao protocolo RSVP, com isso diminuindo os problemas de escalabilidade da arquitetura de serviços integrados;

- a utilização de temporizadores de renovação em etapas na retransmissão de mensagens de controle e na manutenção de estados do mecanismo de agregação. Os temporizadores de renovação em etapas reduzem o tempo de estabelecimento de reservas de recursos em relação ao uso de temporizadores de tempo fixo;

- a modelagem e a implementação do mecanismo de agregação no simulador de redes *ns*. Os resultados das simulações para validação da tese comprovaram os benefícios na utilização do mecanismo de agregação e no uso dos temporizadores de renovação em etapas na presença de perdas de mensagens.

As contribuições e os resultados parciais obtidos ao longo da pesquisa foram apresentados em [100, 101, 106, 107].

Diversas direções podem ser tomadas a partir dos resultados apresentados neste trabalho:

- uma avaliação do tempo de processamento associado com o encapsulamento e desencapsulamento de pacotes nos pontos finais do túnel e o impacto deste processamento no tempo de transmissão fim-a-fim das aplicações, pois o aumento de tráfego em uma rede privada virtual do tipo IP-sobre-IP pode sobrecarregar os roteadores de fronteira devido ao encapsulamento/desencapsulamento de pacotes;

- utilização de um controle de admissão baseado em políticas no mecanismo de agregação *multicast*. O controle de admissão na arquitetura *IntServ* não é suficiente para atender as necessidades atuais das redes. A admissão de um fluxo é baseada somente no pedido de reserva de recursos do receptor e na capacidade do roteador. Os administradores de redes precisam de uma infraestrutura de controle de políticas que regule quais usuários e aplicações devem acessar quais recursos/serviços sob determinados critérios, como requisitos de tráfego, segurança, hora do dia e dia da semana;

- implementação de um protótipo do mecanismo de agregação e a realização de testes para avaliar o atraso fim-a-fim das aplicações multimídias por causa da perda de mensagens de controle e do tempo de encapsulamento e desencapsulamento de pacotes. Fazer uma comparação dos resultados obtidos da implementação do protótipo com os resultados obtidos por simulação.

Referências Bibliográficas

[1] R. Braden, D. Clark e S. Shenker, "Integrated Services in the Internet Architecture: An Overview", *RFC 1633*, julho de 1994.

[2] C. Partridge, "A Proposed Flow Specification", *RFC 1363*, julho de 1992.

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker e D. Zappala, "RSVP: a new Resource ReSerVation Protocol", *IEEE Network Magazine*, vol. 7, pp. 8–18, setembro de 1993.

[4] J. Wroclawski, "The Use of RSVP with Integrated Services", *RFC 2210*, setembro de 1997.

[5] A. Terzis, L. Zhang e E. L. Hahne, "Reservations for Aggregate Traffic: Experiences from an RSVP Tunnels Implementation", *In IEEE/IFIP 6th International Workshop on Quality of Service (IWQoS)*, maio de 1998.

[6] A. Terzis, J. Krawczyk, J. Wroclawski e L. Zhang, "RSVP Operation Over IP Tunnels", *RFC 2746*, janeiro de 2000.

[7] O. Schelén e S. Pink, "Resource Reservation Agents in the Internet", *In 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pp. 153–156, julho de 1998.

[8] S. Berson e S. Vincent, "Aggregation of Internet Integrated Services State", *In IEEE/IFIP 6th International Workshop on Quality of Service (IWQoS)*, maio de 1998.

[9] C. Dovrolis e P. Ramanathan, "Resource Aggregation for Fault Tolerance in Integrated Services Networks", *ACM Computer Communication Review*, vol. 28, no. 2, pp. 39–53, 1998.

[10] K. Fukuda, N. Wakamiya, M. Murata e H. Miyahara, "On Flow Aggregation for Multicast Video Transport", *In 6th IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, maio de 1998.

[11] J. Schmitt, M. Karsten, L. Wolf e R. Steinmetz, "Aggregation of Guaranteed Service Flows", *In 7th IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, junho de 1999.

[12] C. E. Pagani e M. F. Magalhães, "Agregação de Tráfego em um Comutador Adaptativo com Serviços Integrados IP sobre ATM", *XVIII Simpósio Brasileiro de Redes de Computadores (SBRC)*, pp. 537–552, maio de 2000.

[13] F. Baker, C. Iturralde, F. L. Faucheur e B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", *RFC 3175*, setembro de 2001.

[14] P. R. Chandra, P.Steenkiste e A. Fisher, "Extensible Signaling for Temporal Resource Sharing", *IEEE Journal Selected Area in Communications*, vol. 19, no. 3, pp. 438–451, março de 2001.

[15] S. Vutukury e J. J. Garcia-Luna-Aceves, "SMART: A Scalable Multipath Architecture for Intra-domain QoS Provisioning", *In International Workshop on QoS in Multiservice IP Networks*, janeiro de 2001.

[16] A. Terzis, "A Two-Tier Resource Allocation Framework for the Internet", *Ph.D. Thesis*, Computer Science Department, University of California, 2000.

[17] L. Mathy, D. Hutchison, S. Schmid e S. Simpson, "REDO RSVP: Efficient Signalling for Multimedia in the Internet", *In International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS)*, pp. 18–30, outubro de 1999.

[18] P. Pan e H. Schulzrinne, "Staged Refresh Timers for RSVP", *In 2nd Global Internet Conference*, novembro de 1997.

[19] L. Mathy, D. Hutchison e S. Simpson, "Modelling and Improving Flow Establishment in RSVP", *In 6th International Workshop on Protocol for High-Speed Networks (PfHSN)*, pp. 133–150, agosto de 1999.

[20] O. Schelén e S. Pink, "Resource Sharing in Advance Reservation Agents", *Journal of High Speed Networks: Special Issue on Multimedia Networking*, vol. 7, no. 3-4, 1998.

[21] S. Jamin, S. Shenker e P. Danzig, "Comparison of Measurement-based Admission Control Algorithms for Controlled-Load Service", *In IEEE Conference on Computer Communications (INFOCOM)*, abril de 1997.

[22] L. Breslau, S. Shenker e S. Jamin, "Comments on the Performance of Measurement-Based Admission Control Algorithms", *In IEEE Conference on Computer Communications (INFOCOM)*, março de 2000.

[23] L. Georgiadis, R. Guérin, V. Peris e R. Rajan, "Efficient Support of Delay and Rate Guarantees in an Internet", *In ACM/SIGCOMM Symposium on Communications Architectures and Protocols*, agosto de 1996.

[24] J. F. Rezende e S. Romano, "Arquitetura de Serviços Integrados para Comutação IP", *XVI Simpósio Brasileiro de Redes de Computadores*, pp. 670–681, maio de 1998.

[25] S. Rampal e R. Guérin, "Flow Grouping for Reducing Reservation Requirements for Guaranteed Delay Service", *Internet - Draft*, julho de 1997.

[26] V. Fuller, T. Li, J. Yu e K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assigment and Aggregation Strategy", *RFC 1519*, setembro de 1993.

[27] S. Blake, D. L. Black, M. Carlson, E. Davies, Z. Wang e W. Weiss, "An architecture for differentiated services", *RFC 2475*, dezembro de 1998.

[28] R. Rivest, "The MD5 Message-Digest Algorithm", *RFC 1321*, abril de 1992.

[29] M. Karsten, "QoS Signalling and Charging in a Multi-service Internet Using RSVP", *Dr.-Ing. Thesis*, Computer Science Department, Darmstadt University of Technology, 2000.

[30] L. H. M. K. Costa, "Routing in the Internet: Quality of Service and Group Communication", *Thèse de Doctorat*, Université Paris VI Pierre et Marie Curie, 2001.

[31] E. C. Rosen, A. Viswanathan e R. Callon, "Multiprotocol Label Switching Architecture", *RFC 3031*, janeiro de 2001.

[32] D. Awduche, L. Berger, T. Li, V. Srinivasan e G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", *RFC 3209*, julho de 2001.

[33] E. Crawley, R. Nair, B. Rajagopalan e H. Sandick, "A Framework for QoS-based Routing", *RFC 2386*, agosto de 1998.

[34] L. H. M. K. Costa, S. Fdida e O. C. M. B. Duarte, "Distance-vector QoS-based Routing with Three Metrics", *In IFIP High Performance Networking - Networking 2000, Lecture Notes in Computer Science*, pp. 847–856, maio de 2000.

[35] H. Bai, M. Atiquzzaman e W. Ivancic, "Running Integrated Services over Differentiated Service Networks: Quantitative Performance Measurements", *In SPIE Quality of Service over Next-Generation Internet Conference*, julho de 2002.

[36] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski e E. Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", *RFC 2998*, novembro de 2000.

[37] G. Eichler, H. Hussmann, G. Mamais, I. Venieris, C. Prehofer e S. Salsano, "Implementing Integrated and Differentiated Services for the Internet with ATM Networks: A Practical Approach", *IEEE Communications Magazine*, pp. 132–141, janeiro de 2000.

[38] M. Yajnik, J. Kurose e D. Towsley, "Packet Loss Correlation in the MBone Multicast Network", *In Global Internet Conference*, novembro de 1996.

[39] R. Braden, L. Zhang, S. Berson, S. Herzog e S. Jamin, "Resource Reservation Protocol (RSVP) - Version 1 functional specification", *RFC 2205*, setembro de 1997.

[40] J. Wroclawski, "Specification of Controlled-Load Network Element Service", *RFC 2211*, setembro de 1997.

[41] S. Shenker, C. Partridge e R. Guérin, "Specification of Guaranteed Quality of Service", *RFC 2212*, setembro de 1997.

[42] A. K. Parekh e R. G. Gallanger, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case", *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, abril de 1994.

[43] S. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management", *In ACM/SIGCOMM Symposium on Communications Architectures and Protocols*, setembro de 1990.

[44] W. Song e H. Lutfiyya, "An Architecture for Policy-Based RSVP using Differentiated Services", *In IEEE Global Telecommunications Conference (Globecom)*, novembro de 2002.

[45] S. Shenker e L. Breslau, "Two Issues in Reservation Establishment", *In ACM/SIGCOMM Symposium on Communications Architectures and Protocols*, agosto de 1995.

[46] P. White, "RSVP and Integrated Services in the Internet: A Tutorial", *IEEE Communications Magazine*, pp. 100–106, maio de 1997.

[47] L. Delgrossi e L. Berger, "Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+", *RFC 1819*, agosto de 1995.

[48] T. Ballardie, P. Francis e J. Crowcroft, "Core Based Trees (CBT) - An Architecture for Scalable Interdomain Multicast Routing", *In ACM/SIGCOMM*

Symposium on Communications Architectures and Protocols, setembro de 1993.

[49] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, V. Jacobson, C. Liu, P. Sharma e L. Wei, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification", *RFC 2117*, julho de 1997.

[50] A. Adams, J. Nicholas e W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", *Internet Draft*, fevereiro de 2003.

[51] H. Schulzrinne, S. Casner, R. Frederick e V. Jacobson, "RTP : A Transport Protocol for Real-Time Applications", *RFC 1889*, janeiro de 1996.

[52] P. Sharma, D. Estrin, S. Floyd e V. Jacobson, "Scalable Timers for Soft State Protocols", *In IEEE Conference on Computer Communications (INFOCOM)*, abril de 1997.

[53] G. Gaines e M. Festa, "A Survey of RSVP/QoS Implementations Update 2", *RSVP IETF Working Group*, julho de 1998.

[54] A. Neogi, T. Chiueh e P. Stirpe, "Performance Analysis of an RSVP-Capable Router", *IEEE Network Magazine*, pp. 56–63, setembro/outubro de 1999.

[55] M. Baugher e S. Jarrar, "Test Results of the Commercial Internet Multimedia Trials", *In ACM/SIGCOMM Symposium on Communications Architectures and Protocols*, 1998.

[56] M. Karsten, "Design and Implementation of RSVP based on Object-Relationships", *In Networking 2000 Conference, Lecture Notes in Computer Science*, pp. 325–336, abril de 2000.

[57] R. Braden e L. Zhang, "Resource Reservation Protocol (RSVP) - Version 1 Message Processing Rules", *RFC 2209*, setembro de 1997.

[58] A. L. P. Schmidt e R. M. M. Leão, "Análise de Qualidade de Serviço na Internet usando Reserva de Recursos", *XVIII Simpósio Brasileiro de Telecomunicações (SBrT)*, setembro de 2000.

[59] S. Floyd e V. Jacobson, "Link-sharing and Resource Management Models for Packets Networks", *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, agosto de 1995.

[60] R. Yavatkar, D. Pendarakis e R. Guerin, "A Framework for Policy-based Admission Control", *RFC 2753*, janeiro de 2000.

[61] R. Rajan, D. Verma, S. Kamat, E. Felstaine e S. Herzog, "A Policy Framework for Integrated and Differentiated Services in the Internet", *IEEE Network Magazine*, pp. 36–41, setembro/outubro de 1999.

[62] S. Herzog, "RSVP Extensions for Policy Control", *RFC 2750*, janeiro de 2000.

[63] Y. Bernet e R. Pabbati, "Application and Sub Application Identity Policy Element for Use with RSVP", *RFC 2872*, junho de 2000.

[64] J. Boyle, R. Cohen, D. Durham, R. Rajan, S. Herzog e A. Sastry, "The COPS (Common Open Policy Service) Protocol", *RFC 2748*, janeiro de 2000.

[65] J. Boyle, R. Cohen, D. Durham, R. Rajan, S. Herzog e A. Sastry, "COPS usage for RSVP", *RFC 2749*, janeiro de 2000.

[66] A. Ponnappan, L. Yang e R. Pillai, "A Policy Based QoS Management System for the IntServ/DiffServ Based Internet", *In 3rd IEEE International Workshop on Policies for Distributed Systems and Networks*, junho de 2002.

[67] A. K. Talukdar, B. R. Badrinath e A. Acharya, "On Accommodating Mobile Hosts is an Integrated Services Packet Network", *In IEEE Conference on Computer Communications (INFOCOM)*, abril de 1997.

[68] A. Terzis, M. Srivastava e L. Zhang, "A Simple QoS Signaling Protocol for Mobile Hosts in the Integrated Services Internet", *In IEEE Conference on Computer Communications (INFOCOMM)*, março de 1999.

[69] A. K. Talukdar, B. R. Badrinath e A. Acharya, "MRSVP: A Reservation Protocol for an Integrated Services Packet Network with Mobile Hosts", *ACM/Baltzer Journal of Wireless Networks*, vol. 7, Issue 1, janeiro de 2001.

[70] J.-M. Kim, W.-J. Kim e Y.-J. Suh, "RSVP Adaptation for QoS Guarantees in the 3rd Generation Network", *In International Conference on Third Generation Wireless and Beyond (3GWireless)*, pp. 629–634, maio de 2002.

[71] C. E. Perkins, "Mobile IP", *IEEE Communications Magazine*, pp. 84–89, maio de 1997.

[72] L. Delgrossi, R. G. Herrtwich, C. Vogt e L. C. Wolf, "Reservation Protocols for Internetworks: A Comparison of ST-II and RSVP", *In 4th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, novembro de 1993.

[73] D. J. Mitzel, D. Estrin, S. Shenker e L. Zhang, "An Architectural Comparison of ST-II and RSVP", *In IEEE Conference on Computer Communications (INFOCOM)*, junho de 1994.

[74] P. Pan e H. Schulzrinne, "YESSIR: A Simple Reservation Mechanism for the Internet", *ACM Computer Communication Review*, vol. 29, no. 2, pp. 89–101, abril de 1999.

[75] P. Pan e H. Schulzrinne, "Processing Overhead Studies in Resource Reservation Protocols", *In 17th International Teletraffic Congress*, pp. 89–101, dezembro de 2001.

[76] G. Feher, K. Nemeth, M. Maliosz, I. Cselenyi, J. Bergkvist, D. Ahlard e T. Engborg, "Boomerang - A Simple Protocol for Resource Reservation in IP Networks", *In IEEE Workshop on QoS Support for Real-Time Internet Applications*, junho de 1999.

[77] G. Feher, K. Nemeth e I. Cselenyi, "Performance Evaluation Framework for IP Resource Reservation Signalling", *Performance Evaluation Journal*, vol. 48(1-4), pp. 131–156, Elsevier Science B.V, maio de 2002.

[78] U. Roedig, C. M. Gortz, M. Karsten e R. Steinmetz, "RSVP as Firewall Signalling Protocol", *In 6th IEEE Symposium on Computers and Communications*, pp. 57–62, julho de 2001.

[79] O. Fourmeaux e S. Fdida, "Multicast for RSVP Switching", *Telecommunications Systems Journal*, vol. 11, pp. 85–104, março de 1999.

[80] C. Metz, "RSVP: General-Purpose Signaling for IP", *IEEE Internet Computing Magazine*, pp. 95–99, maio/junho de 1999.

[81] R. Greco, L. Delgrossi e M. Brunner, "Towards RSVP Version 2", *In 2nd International Workshop on QoS in Multiservice IP Networks*, pp. 704–716, fevereiro de 2003.

[82] X. Fu e C. Kappler, "Towards RSVP Lite: Light-weight RSVP for Generic Signaling", *In International Conference on Advanced Information Networking and Applications (AINA)*, março de 2003.

[83] R. Greco e M. Brunner, "Towards RSVP Version 2", *Internet Draft*, outubro de 2002.

[84] V. Paxson, "End-to-End Routing Behavior in the Internet", *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 610–615, outubro de 1997.

[85] M. Handley, "An Examination of MBone Performance", *UCS/ISI Technical Report ISI/RR-97-450*, janeiro de 1997.

[86] M. Yajnik, J. Kurose e D. Towsley, "Measurement and Modelling of the Temporal Dependence Packet Loss", *In IEEE Conference on Computer Communications (INFOCOM)*, março de 1999.

[87] D. Loguinov e H. Radha, "End-to-End Internet Video Traffic Dynamics: Statical Study and Analysis", *In IEEE Conference on Computer Communications (INFOCOM)*, junho de 2002.

[88] "The Internet Traffic Report", *http://www.internettrafficreport.com*, 2003.

[89] "The Internet Weather Report", *http://www.internetweather.com*, 2003.

[90] H. Sanneck e G. Carle, "A Framework Model for Packet Loss Metrics Based on Loss Runlengths", *In SPIE/ACM SIGMM Multimedia Computing and Networking Conference (MMCN)*, janeiro de 2000.

[91] Y. Zhang, V. Paxson e S. Shenker, "The Stationary of Internet Path Properties: Routing, Loss, and Throughput", *ACIRI Technical Report*, maio de 2000.

[92] S. Floyd e V. Jacobson, "The Synchronization of Periodic Routing Messages", *IEEE/ACM Transactions on Networking*, vol. 2, pp. 122–136, abril de 1994.

[93] J. Bolot, "End-to-End Packet Delay and Loss Behavior in the Internet", *In ACM/SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 289–298, setembro de 1993.

[94] P. Ferguson e G. Huston, "What's is a VPN", *White-Paper, ftp://ftp.employees.org/ferguson/*, abril de 1998.

[95] L. H. M. K. Costa, S. Fdida e O. C. M. B. Duarte, "An Introduction to Virtual Private Networks: Towards D-VPNs", *Networking and Information Systems Journal*, vol. 3, no. 4, novembro de 2000.

[96] C. Perkins, "IP Encapsulation within IP", *RFC 2003*, outubro de 1996.

[97] C. Perkins, "Minimal Encapsulation within IP", *RFC 2004*, outubro de 1996.

[98] A. Conta e S. Deering, "Generic Packet Tunneling in IPv6 Specification", *RFC 2473*, dezembro de 1998.

[99] G. Booch, J. Rumbaugh e I. Jacobson, *The Unified Modeling Language - User Guide*. Addison-Wesley, 1999.

[100] P. C. S. Vidal e O. C. M. B. Duarte, "Agregação de Estados do Protocolo RSVP para Fluxos *Multicast*", *XVIII Simpósio Brasileiro de Telecomunicações (SBrT)*, setembro de 2000.

[101] P. C. S. Vidal e O. C. M. B. Duarte, "Multicast Flow Aggregation in IP Tunnels with Resource Reservation", *In International Information Technology Symposium (I2TS)*, outubro de 2002.

[102] T. Bates, R. Chandra, D. Katz e Y. Rekhter, "Multiprotocol Extensions for BGP-4", *Internet-Draft*, janeiro de 1998.

[103] K. Nichols, S. Blake, F. Baker e D. L. Black, "Definition of the Differentiated Services Field (DS field) in the IPv4 and IPv6 Headers", *RFC 2474*, dezembro de 1998.

[104] "Network Simulator - NS (version2)", *http://www-mash.cs.berkeley.edu/ns/*.

[105] M. Greis, "RSVP/ns: An Implementation of RSVP for the Network Simulator ns-2", *http://titan.cs.uni-bonn.de/~greis/rsvpns/rsvpns.ps.gz*, 1999.

[106] P. C. S. Vidal e O. C. M. B. Duarte, "Estabelecimento de Reservas de Recursos para Fluxos *Multicast* em Túneis IP-sobre-IP", *V Simpósio Brasileiro de Sistemas Multimídia e Hypermídia (SBMIDIA)*, pp. 252–259, outubro de 2002.

[107] P. C. S. Vidal e O. C. M. B. Duarte, "Reducing the Reservation Establishment Time in IP Tunnels by Using Staged Refresh Timers", *In 2nd IEEE International Symposium on Network Computing and Applications*, pp. 389–396, abril de 2003.

Apêndice A

O Simulador de Redes *ns*

O simulador *ns* é um simulador de protocolos de rede em desenvolvimento no projeto *Virtual InterNet Testbed* (VINT), uma colaboração entre UC Berkeley, LBL, USC/ISI e Xerox PARC. O *ns* é um simulador orientado a objetos escrito em linguagem C++. O simulador fornece suporte à arquitetura TCP/IP, comunicação *multicast*, redes sem fio, roteamento, satélite, etc. Tem facilidades de *tracing*, que é a coleta e registro de dados de cada evento da simulação para análise porterior. Possui um visualizador gráfico para animações da simulação (*nam - network animator*), temporizadores, escalonadores, modelos para controle de erro e algumas ferramentas matemáticas como gerador de números aleatórios e integrais para cálculos estatísticos. Inclui também uma ferramenta de plotagem, o *xgraph*, e vários tipos de geradores de tráfego.

Os usuários criam *scripts* na linguagem OTcl que são interpretados e executados pelo simulador. O simulador suporta uma hierarquia de classes em C++ e uma hierarquia de classes similar para o interpretador OTcl. Os usuários criam novos objetos do simulador através do interpretador que instancia estes objetos e gera um espelho em C++, criando um objeto correspondente.

As topologias são criadas no *ns* através de nós (Classe *Node*) e da conexão destes nós por enlaces (Classe *Link*). O *ns* permite a utilização de dois tipos de nós, ponto-a-ponto e multiponto. A função de um nó, quando recebe um pacote, é de examinar seus campos, normalmente o campo de endereço destino, e mapear os valores em um objeto de interface de saída que servirá de encaminhamento para o próximo receptor deste pacote.

Os enlaces no *ns* são compostos por objetos chamados conectores pertencentes a Classe *Connector*. Os conectores apenas geram dados para um receptor. O pacote pode ser enviado para este receptor ou pode ser descartado. Um enlace básico é criado com características específicas de banda passante e atraso.

Os agentes do *ns* representam pontos onde os pacotes são gerados ou consumidos e são usados para a implementação de protocolos de várias camadas. Geralmente, um usuário querendo criar uma nova fonte ou receptor de pacotes da camada de rede deve criar uma classe derivada da classe *Agent*.

A.1 Modelos de Erros

Esta seção apresenta dois modelos de erros utilizados nas simulações do mecanismo de agregação *multicast* que estão implementados no simulador *ns* através da classe *ErrorModel* e da classe *ErrorModel/Multistate*.

O modelo de erro simula erros ou perdas no enlace marcando um *flag* de erro no pacote ou descarregando o pacote em um agente de descarte de pacotes. Nas simulações, os erros podem ser gerados de um modelo simples como a taxa de perdas de pacotes, ou modelos estatísticos mais complexos. Para fornecer suporte a diversos tipos de erros, a unidade de erro pode ser especificada em termos de pacote, bits ou baseado no tempo.

A classe base para modelagem das perdas é a classe *ErrorModel* que é derivada da classe *Connector*. A classe *ErrorModel* define o método *unit* para especificar a unidade de erro e a variável *ranvar* para especificar a variável randômica para gerar erros. Se não especificada, a unidade de erro será em pacotes, e a variável randômica será uniformemente distribuída entre 0 e 1. Um exemplo de criação de modelo de erro com taxa de erros de pacotes de 1% (0,01) em um enlace é apresentado abaixo:

```
# cria um modulo_perda e uma taxa de erro de 1%
set modulo_perda [new ErroModel]
$modulo_perda set rate_ 0.01

# opcional: configura a unidade de perda e a variável aleatória
$em unit pkt
$em ranvar [new RandomVariable/Uniform]

# instalação do módulo de erro no enlace
$ns lossmodel $modulo_perda $n(0) $n(1)

# configura o alvo para os pacotes descartados
$modulo_perda drop-target [new Agent/Null]
```

O modelo de erro multiestado implementa uma máquina de estados. As transições para o próximo estado ocorrem no final da duração do estado corrente. O próximo estado de erro é então selecionado usando a matriz de transições de estados.

Para criar um modelo multiestado, devem ser fornecidos os seguintes parâmetros: um arranjo de estados, o período de duração de cada estado, a matriz de transição de estados, a unidade de erro, o tipo de transição entre os estados (baseado em tempo por pacote), o número de estados e o estado inicial.

O *script* abaixo mostra um exemplo de uso do modelo para 3 (três) estados:

```
# Criação dos 3 estados (modelos de estados)
set e0 [new ErrorModel/Uniform  0 pkt]
set e1 [new ErrorModel/Uniform .9 pkt]
set e2 [new ErrorModel/Uniform .5 pkt]

# Arranjo de 3 estados
set m_estados [list $e0 $e1 $e2]
# Duração de cada estado em segundos, e0, e1 and e2, respectivamente
```

```
set m_periodos [list 0 .0075 .00375]
# Matriz de transições dos modelos de estados
set m_matriz { {0.95 0.05 0} {0   0   1} {1   0   0} }
set m_unid pkt
# Transição de estados baseado no tempo
set m_tipotrans time
set m_estados 3
# Estado 0 inicia o processo
set m_estinicial [lindex $m_estados 0]

# Criação do modelo multiestado
set em [new ErrorModel/MultiState $m_estados $m_periodos $m_matriz
         $m_unid $m_tipotrans $m_nestados $m_estinicial]

# instalação do módulo de erro no enlace
$ns lossmodel $em $n(0) $n(1)
```

A.2 Comandos em OTcl do RSVP Túnel

Esta seção descreve os comandos desenvolvidos na linguagem OTcl para usar o RSVP Túnel em *scripts* de simulação. Os comandos *duplex-rsvp-link*, *add-rsvp-agent* e *session* foram obtidos do RSVP/ns [105].

Os comandos *sender* e *reserve* foram adaptados para o RSVP Túnel e os comandos *add-entry-router* e *add-border-routers* foram criados para o RSVP Túnel.

Configurando um enlace RSVP

Um enlace RSVP entre os nós n1 e n2 é criado pelo comando

```
ns duplex-rsvp-link <n1> <n2> <lb> <at> <pr> <lbr> <tf> <cad> <est>
```

onde 'ns' é a instância do simulador. Os outros argumentos são:

- lb - largura de banda do enlace;
- at - o atraso do enlace;
- pr - a porção da largura de banda do enlace que pode ser usada por mensagens RSVP;
- lbr - a largura de banda (em bps) que é reservada para mensagens RSVP;
- tf - o tamanho da fila para a classe de serviço de melhor-esforço;
- cad - o tipo de controle de admissão (baseado em parâmetros ou em medidas);
- est - o estimador usado pelo controle de admissão baseado em medidas.

Criação e configuração de agentes RSVP

Um agente RSVP é associado a um nó com o comando:

```
<no> add-rsvp-agent
```

Este comando retorna um valor associado ao agente. O exemplo seguinte associa um agente RSVP para o nó *n* e armazena o valor na variável *agente_rsvp*.

```
set agente_rsvp [$n add-rsvp-agent]
```

Configurando um agente RSVP como roteador de entrada:

```
<no> add-entry-router <agente-rsvp> <rentrada>
```

O parâmetro <rentrada> será 1(um) para indicar que o agente é um roteador de entrada e zero caso o contrário.

Configurando um agente RSVP como roteador de fronteira:

```
<no> add-border-routers <agente-rsvp> <fronteira>
```

O parâmetro <fronteira> será 1(um) para indicar que o agente é um roteador de fronteira e zero caso o contrário.

Envio de mensagens de controle

Criando uma sessão individual ou uma sessão túnel:

```
<agente-rsvp> session <destino> <id>
```

O argumento <destino> é o identificador do nó e o argumento <id> pode ser um identificador do fluxo ou um identificador do túnel. Este comando retorna um identificador da sessão túnel ou individual que será usado nos comandos de envio de mensagens de controle.

Enviando uma mensagem PATH túnel:

```
<agente-rsvp> sender <id-sessão> <taxa> <bucket> <ttl> <tipo> <obj>
```

Os argumentos deste comando são:

- id-sessão - identificador da sessão relacionado com o comando "session";
- taxa - largura de banda da fonte;
- bucket - tamanho do *token bucket*;
- ttl - valor que representa o tempo de vida (*ttl - time to life*) da mensagem;
- tipo - mensagem de criação do estado do caminho ou mensagem de atualização;
- obj - número do objeto PATH_CONFIRM.

Enviando uma mensagem RESV túnel:

```
<agente-rsvp> reserve <id-sessão> <taxa> <bucket> <origem> <obj>
```

Os argumentos deste comando são:

- id-sessão - identificador da sessão relacionado com o comando "session";
- taxa - largura de banda da fonte;
- bucket - tamanho do *token bucket*;
- origem - endereço do emissor da mensagem PATH túnel;
- obj - número do objeto RESV_CONFIRM.

A.3 Exemplo de um *script* de simulação

Esta seção apresenta um *script* de simulação do mecanismo de agregação *multicast* para uma árvore binária de 4(quatro) nós com enlace de gargalo utilizando o modelo de perdas periódicas no enlace de gargalo e nos demais enlaces sendo configurados com uma taxa média de perdas de pacotes.

O objetivo do *script* é obter o tempo de configuracao das reservas no túnel na presença de perdas de mensagens de controle.

O *script* está organizado pelas seguintes etapas:

- leitura de parâmetros externos no inicio da execução do *script*;
- abertura dos arquivos para *tracing*;
- criação da topologia e dos agentes RSVP;
- criação dos modelos de perdas de pacotes;
- envio das mensagens PATH túnel e RESV túnel;
- finalização do *script*.

A codificação do *script* está descrita abaixo:

```
# Criação da instância do objeto Simulator.
set ns [new Simulator]

# Leitura de parâmetros externos: nr de sessoes, duração da simulação,
# duração do estado sem erro, duração do estado com erro
# e percentual de perdas de pacotes.
set NR_SESSOES  [lindex $argv 0]
set simtime     [lindex $argv 2]
set tgood       [lindex $argv 3]
set tbad        [lindex $argv 4]
set perc        [lindex $argv 5]
set taxa_erro   [expr $perc / 100.0]

# Abertura dos arquivos para trace e análise posterior.
set arq_trace atraso_tg${tgood}b${tbad}ns${NR_SESSOES}P${perc}.tr
set f0 [open $arq_trace w]
set f1 [open n_path_tg${tgood}tb${tbad}ns${NR_SESSOES}P${perc}.tr w]
set f2 [open n_resv_tg${tgood}tb${tbad}ns${NR_SESSOES}P${perc}.tr w]

# Outros parametros de simulacao: largura de banda inicial (bps)
# e incremento da largura de banda a ser reservada (bps)
set BW 8000
set deltabw 100
# Intervalo médio de chegada entre msgs RSVP (em segundos)
set fint 0.1

# Criação dos agentes RSVP
for {set i 0} {$i < 4} {incr i} {
        set n($i) [$ns node]
        set rsvp($i) [$n($i) add-rsvp-agent]
}
```

```
$n(0) add-entry-router   $rsvp(0) 1
$n(1) add-entry-router   $rsvp(1) 0
$n(2) add-entry-router   $rsvp(2) 0
$n(3) add-entry-router   $rsvp(3) 0

$n(0) add-border-routers $rsvp(0) 0
$n(1) add-border-routers $rsvp(1) 0
$n(2) add-border-routers $rsvp(2) 1
$n(3) add-border-routers $rsvp(3) 1

# Criação da topologia e das sessões tuneis
$ns duplex-rsvp-link $n(0) $n(1) 100Mb 0s 1.0 200 5000 Param Null
$ns duplex-rsvp-link $n(1) $n(2) 100Mb 0s 1.0 200 5000 Param Null
$ns duplex-rsvp-link $n(1) $n(3) 100Mb 0s 1.0 200 5000 Param Null

$rsvp(0) session $n(0) $n(2) 0
$rsvp(0) session $n(0) $n(3) 1

# Criação do modelo de perdas periódicas, modelo de multiestados
set good [new ErrorModel/Uniform   0   pkt]
set bad  [new ErrorModel/Uniform   1   pkt]
set m_states  [list $good $bad]
set m_periods [list $tgood $tbad]
set m_transmx { { 0.9 0.1 } { 0.8  0.2 } }
set m_trunit  pkt
set m_sttype  time
set m_nstates 2
set m_nstart  [lindex $m_states 0]

set em01 [new ErrorModel/MultiState $m_states $m_periods
```

```
 $m_transmx $m_trunit $m_sttype $m_nstates $m_nstart]
$ns lossmodel $em01 $n(0) $n(1)
$em01 drop-target [new Agent/Null]

set em10 [new ErrorModel/MultiState $m_states $m_periods
 $m_transmx $m_trunit $m_sttype $m_nstates $m_nstart]
$ns lossmodel $em10 $n(1) $n(0)
$em10 drop-target [new Agent/Null]

# Criação do modelo ErrorModel nos enlaces n1-n2 e n1-n3
set em12 [new ErrorModel]
$em12 set rate_ $taxa_erro
$ns lossmodel $em12 $n(1) $n(2)
$em12 drop-target [new Agent/Null]

set em13 [new ErrorModel]
$em13 set rate_ $taxa_erro
$ns lossmodel $em13 $n(1) $n(3)
$em13 drop-target [new Agent/Null]

set fvar [new RandomVariable/Exponential]
$fvar set avg_ $fint
set i 1
set inicio 0.0
# Envio da mensagem PATH para o tunel 0 e tunel 1.
while { $i <= $NR_SESSOES } {
  set valor [$fvar value]
  set inicio [expr $inicio + $valor]
  $ns at $inicio "$rsvp(0) sender 0 $BW 5000 32 0 $i"
  $ns at $inicio "$rsvp(0) sender 1 $BW 5000 32 0 [expr $i + 10000]"
  set BW [expr $BW + $deltabw]
```

```
  incr i
}


# Redefine a função upcall-path para automaticamente
# enviar o pedido de reserva quando receber a msg PATH.
Agent/RSVPTunel instproc upcall-path {sid rate bucket sender objpath} {
     if { $self == $rsvp(2) } || { $self == $rsvp(3) } {
         $self reserve $sid FF $rate $bucket $sender $objpath
     }
}


$ns at $simtime "finalizar"


proc finalizar {} {
        global  f0 f1 f2
        close $f0 $f1 $f2
        exit 0
}


$ns run
```

Apêndice B

Classes do Mecanismo de Agregação

Este Apêndice mostra a especificação das principais classes utilizadas na implementação do mecanismo de agregação de estados do protocolo RSVP para fluxos *multicast* em túneis IP-sobre-IP. As seguintes classes são apresentadas: RSVPTunel, RSVPConector, RSVPLink, Encap e Decap.

B.1 Classe RSVPTunel

```
/* Associação da Classe RSVPTunel em C++ e OTcl  */
static class RSVPTunelClass : public TclClass {
public:
  RSVPTunelClass() : TclClass("Agent/RSVPTunel") {}
  TclObject* create(int, const char*const*) {
  return (new RSVPTunel());
  }
} class_RSVPTunel;

class RSVPTunel : public Agent {
 public:
   RSVPTunel();
```

```
   int command(int argc, const char*const* argv);
   void recv(Packet *p, Handler* h);
   void give(Packet *p, RSVPConector *ret, int encap, int decap);
 protected:

/* --------- Métodos para gerenciar fluxos agregados  ------------*/

void       check_resv_tunnel(session_t *s, rsb_t *r, RSVPMsg *msg);
session_t *create_session_tunnel(nsaddr_t dst, int fid, char local);
session_t *find_session_tunnel(nsaddr_t in_t, nsaddr_t out_t);
int        find_tunnel_sessions_list(session *si, long sid);
int        find_sids_list_st_si(session_t *st, long stid);
int        insert_st_si(session_t *st, session *si);
psb_t      *new_psb_tunnel(session_t *s, double rate, long bucket,
           nsaddr_t sender, nsaddr_t phop);
rsb_t      *new_rsb_tunnel(session_t *s, FLOWSPEC_M *fl, char *style,
           nsaddr_t in_t, nsaddr_t out_t, nsaddr_t fromhop);
rsb_t      *new_rsb_tunnel_ri(session_t *s, FLOWSPEC_M *fl,
           char *style, nsaddr_t in_t, nsaddr_t out_t,
           nsaddr_t fromhop);
session_t *new_session_tunnel(session_t *s, nsaddr_t in_t,
           nsaddr_t out_t);
session_t *new_session_tunnel_ri(nsaddr_t in_t, nsaddr_t out_t);
void       proc_reserve_request(session_t *s, int argc, char* argv);
void       proc_path_tunnel_message(RSVPMsg *msg, int iface);
void       proc_resv_tunnel_message(RSVPMsg *msg, nsaddr_t fromhop);
void       proc_tunnel_rconf_message(RSVPMsg *msg, nsaddr_t fromhop);
void       send_path_tunnel_message(session_t *s, psb_t *p);
int        send_resv_tunnel_message(session_t *s, session *s);
int        send_resv_tunnel_message_ri(session_t *s, rsb_t *r);
void       send_tunnel_resv_conf_message(session_t *s, rsb_t *r);
```

```
void      trigger_path_encap_msg(session *s, psb *p,
          int session_assoc, nsaddr_t out_t);
void      trigger_path_tunnel_msg(session *s, psb *p, nsaddr_t out_t);
void      trigger_resv_tunnel_msg(session_t *st, session *si);
char      update_psb_tunnel(session_t *st, session *s, psb *p,
          double rate, long bucket, nsaddr_t sender);
void      update_rsb_tunnel(session_t *s, nsaddr_t sender,
          char *style, FLOWSPEC_M *fl, nsaddr_t nhop);
int       update_tunnel_traffic_control(session_t *s, rsb_t *r);

/* ---------- Métodos para gerenciar fluxos individuais ----------*/

char    check_path(session *s);
char    check_resv(session *s);
psb     *find_psb(session *s, nsaddr_t sender);
rsb     *find_rsb(session *s, nsaddr_t sender, nsaddr_t nhop);
session *find_session_dst(nsaddr_t dst, int fid);
session *find_session_sid(long sid);
psb     *new_psb(session *s, double rate, long bucket, char ttl,
nsaddr_t sender, nsaddr_t phop, int encap);
rsb     *new_rsb(session *s, psb *p, nsaddr_t sender,
        FLOWSPEC *fl, char *style, nsaddr_t nhop, nsaddr_t fromhop);
int     new_session_sid(nsaddr_t dst, int fid);
session *new_session_p(nsaddr_t dst, int fid);
void    proc_ff_request(session *s, int argc, char* argv );
void    proc_path_message(RSVPMsg *msg, int encap);
void    proc_path_message_decap(RSVPMsg *msg, int iface);
void    proc_resv_message(RSVPMsg *msg, nsaddr_t fromhop);
void    proc_resv_message_decap(RSVPMsg *msg, nsaddr_t fromhop);
void    release_session(session *s);
void    send_path_message(session *s, psb *p, int encap,
```

```
        int encap_with_session_assoc, nsaddr_t out_t);
int     send_resv_message(session *s, rsb *r);
char    update_psb(session *s, psb *p, double rate, long bucket,
        nsaddr_t sender, nsaddr_t phop, int iface, int encap);
void    update_rsb(session *s, rsb *r, psb *p, nsaddr_t sender,
        FLOWSPEC_M *fl, char *style, nsaddr_t nhop, nsaddr_t fromh);
};

/* ------- Método command da Classe RSVPTunel ------*/
int RSVPTunel::command(int argc, const char*const* argv)
{
 Tcl& tcl = Tcl::instance();
 /* session dest fid */
 if (strcmp(argv[1], "session") == 0)  {
    if (argc != 4) {
      return (TCL_ERROR);
    }
    if (argv[2][0] == '_') {
      tcl.evalf("%s id", argv[2]);
      tcl.resultf("%d", new_session_tunnel(atoi(tcl.result()),
                  atoi(argv[3]), 1));
    }
     else {
     tcl.resultf("%d",new_session_tunnel(strtol(argv[2],NULL,0),
                 atoi(argv[3]), 1));
    }
  return (TCL_OK);
 }
/* sender session-id rate bucket-size ttl */
 if (strcmp(argv[1], "sender") == 0) {
   if (argc != 6)  {
```

```
      return (TCL_ERROR);
   }
   session *s = find_session_tunnel(atoi(argv[2]));
   if (s == NULL)     {
      tcl.resultf("%s RSVP: ID da sessao desconhecida %s",
                  name(), argv[2]);
      return(TCL_ERROR);
   }
   psb *p;
   if ((p = find_psb_tunnel(s,
      addr_ >> Address::instance().NodeShift_[1])) == NULL) {
      new_psb_tunnel(s, atof(argv[3]), atoi(argv[4]),
       atoi(argv[5]),addr_ >> Address::instance().NodeShift_[1]);
      num_psb_++;
   }
    else {
    update_psb_tunnel(s, p, atof(argv[3]), atoi(argv[4]),
     atoi(argv[5]), addr_ >> Address::instance().NodeShift_[1]);
   }
   return (TCL_OK);
 }
 /* reserve session-id style <flow descriptor list> */
  if (strcmp(argv[1], "reserve") == 0)   {
     if (argc < 6)  {
        return (TCL_ERROR);
     }
     session *s = find_session_tunnel(atoi(argv[2]));
     if (s == NULL)    {
        if (noisy_ & UPC_RESVERR)   {
           tcl.evalf("%s upcall-resv-error %s 4 0 %d", name(),
             argv[2],addr_ >> Address::instance().NodeShift_[1]);
```

```
        }
      }
       else
      {
       /* Existe algum PSB para esta sessao ? */
       if (((p = find_psb_tunnel(s,
          addr_ >> Address::instance().NodeShift_[1])) == NULL)) {
          if (noisy_ & UPC_RESVERR)  {
             tcl.evalf("%s upcall-resv-error %s 3 0 %d", name(),
              argv[2], addr_ >> Address::instance().NodeShift_[1]);
          }
       }
        else   {
          if (strcasecmp(argv[3], "ff") != 0)  {
             if (noisy_ & UPC_RESVERR)  {
                tcl.evalf("%s upcall-resv-error %s 6 0 %d", name(),
                 argv[2],addr_ >> Address::instance().NodeShift_[1]);
             }
          }
           else {
              process_reservation_request(s, argc, argv);
          }
       }
      }
   return (TCL_OK);
  }
}
```

B.2 Classe RSVPConector e Classe RSVPLink

```
/* Associação da Classe RSVPLink em C++ e OTcl */
```

```
static class RSVPLinkClass : public TclClass {
public:
  RSVPLinkClass() : TclClass("RSVPLink") {}
  TclObject* create(int, const char*const*) {
    return (new RSVPLink());
  }
} class_RSVPLink;

class RSVPConector : public Connector {
 public:
  RSVPConector();
  void give(Packet *p);
  void recv(Packet* p, Handler* h);
 protected:
  int off_ip_;
  int off_rsvpm_;
  nsaddr_t src_;
  nsaddr_t dst_;
};

class SALink : public Connector {
public:
        SALink();
        int command(int argc, const char* argv);
        void trace(TracedVar* v);
protected:
        void recv(Packet *,Handler *);
        ADC *adc_;
        int RTT;
        int off_ip_;
        int off_resv_;
```

```
        pending pending_[NFLOWS];
        int lookup(int);
        int get_nxt();
        TracedInt numfl_;
        Tcl_Channel tchan_;
        int onumfl_;   /* XXX: store previous value of numfl_ */
        int src_;      /* id of node we're connected */
        int dst_;      /* id of node at end of the link */
        int last_;     /* previous ac decision on this link */
};

/* Associação da Classe RSVPConector em C++ e OTcl */
static class RSVPConectorClass : public TclClass {
public:
  RSVPConectorClass() : TclClass("RSVPConector") {}
  TclObject* create(int, const char*const*) {
    return (new RSVPConector());
  }
} class_RSVPConector;

class RSVPLink : public SALink {
 public:
  RSVPLink();
 protected:
  double besteffort_;
};
```

B.3 Classe Encap e Classe Decap

```
/*  Associação da Classe Encap em C++ e OTcl    */
static class EncapClass : public TclClass {
```

```
public:
        EncapClass() : TclClass("Agent/Encap") {}
        TclObject* create(int, const char*const*) {
                return (new Encap());
        }
} class_encap;

class Encap : public Agent {
public:
    Encap();
    void recv_encap(Packet* p, RSVPConector* con, nsaddr_t in_t,
                    nsaddr_t out_t, nsaddr_t group, int is_path);
protected:
    nsaddr_t addr_;
    int off_encap_;
    int off_ip_;
};

/* Associação da Classe Decap em C++ e em OTcl    */
static class DecapClass : public TclClass {
public:
        DecapClass() : TclClass("Agent/Decap") {}
        TclObject* create(int, const char*const*) {
                return (new Decap());
        }
} class_decap;

class Decap : public Agent {
public:
    Decap();
    void recv_decap(Packet* p, RSVPTunel* ag, RSVPConector* con);
```

```
protected:
    int decapsulated_;
    static int off_encap_;
    int off_ip_;
};
```