Jefferson Manhães de Azevedo

Uma Proposta para o Tráfego *Multicast* em Redes de Serviços Diferenciados

Orientadores: Profª . Luci Pirmez, D.Sc.
Prof. Oswaldo Vernet de Souza Pires, D.Sc.
Prof. Luiz Fernando Rust da Costa Carmo, Dr. UPS.

Núcleo de Computação Eletrônica - NCE
Instituto de Matemática - IM
Universidade Federal do Rio de Janeiro - UFRJ

Rio de Janeiro, Março de 2001.

UMA PROPOSTA PARA O TRÁFEGO MULTICAST
EM REDES DE SERVIÇOS DIFERENCIADOS

Jefferson Manhães de Azevedo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE MATEMÁTICA/NÚCLEO DE COMPUTAÇÃO ELETRÔNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM INFORMÁTICA.

Aprovada por:

_____

Profª . Luci Pirmez, D.Sc.

_____

Prof. Oswaldo Vernet de Souza Pires, D.Sc.

_____

Prof. Luiz Fernando Rust da Costa Carmo, Dr. UPS.

_____

Prof. Jose Ferreira de Rezende, Dr.

_____

Prof. Ageu Cavalcanti Pacheco Junior, PhD.

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2001

FICHA CATALOGRÁFICA

AZEVEDO, JEFFERSON MANHÃES.

Uma Proposta para o Tráfego Multicast em Redes de Serviços Diferenciados [Rio de Janeiro] 2001

IX, 111 p. 29,7 cm (IM/NCE/UFRJ, M.Sc., Informática, 2001)

Dissertação - Universidade Federal do Rio de Janeiro, IM/NCE

1. Serviços Diferenciados
2. Tráfego *Multicast*

I. IM/NCE/UFRJ II. Título (série)

AGRADECIMENTOS

A Deus, pela sua presença constante em minha vida. “OBRIGADO SENHOR, porque és meu amigo, porque sempre contigo eu posso contar”.

À minha esposa Cristiane, pelo seu companheirismo, amor e paciência, e aos meus pais e familiares, pois sempre me incentivaram e apoiaram.

Aos meus padrinhos, Guilherme e Cláudia, e aos frades Dominicanos, por me acolherem em suas residências ao longo deste Mestrado. Em especial, gostaria de agradecer ao amigo Fr. Mário Taurinho, ao Fr. Sérgio Lobo, ao Fr. Bruno de Palma e a Inês.

Aos meus orientadores, Luci Pirmez, Oswaldo Vernet e Luiz Fernando Rust da Costa Carmo, pelas valiosas contribuições e apoio ao longo deste nosso trabalho. Em especial, gostaria de agradecer ao Prof. Oswaldo, pela paciência, amizade e dedicação.

Ao Prof. Antonio Carlos Gay Thomé e ao Prof. Astério Tanaka, pela confiança que depositaram em mim na recomendação ao Mestrado.

Aos colegas de Mestrado, pelo companheirismo, troca de informações e apoio nos momentos difíceis que passamos juntos. Em especial ao Eduardo Cunha, Fábio David, João Carlos Peixoto, José Duarte Queiroz e Márcio Leocádio.

À Universidade Candido Mendes e o Centro Federal de Educação Tecnológica de Campos, por todo apoio que me deram para que eu pudesse cursar o Mestrado.

Aos amigos que se envolveram e acompanharam ao longo do desenvolvimento deste trabalho, em particular, ao amigo Luiz Gustavo Lourenço Moura e a amiga Lígia Henriques.

Ao Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro, pela oportunidade de desenvolver este trabalho em uma universidade pública de

qualidade. Em especial, agradeço ao Prof. Júlio Salek (in memorian) pela sua presença amiga e ao Prof. Pedro Salenbauch por sua valiosa ajuda nas soluções relativas ao ajuste do núcleo do TROPIX para a implementação dos elementos da arquitetura proposta neste trabalho.

Em fim, a todos aqueles que me ajudaram...

Este trabalho é fruto de muitas mãos, muitas cabeças e, principalmente, de muitos corações.

RESUMO

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

Uma Proposta para o Tráfego Multicast em Redes de Serviços Diferenciados

Jefferson Manhães de Azevedo

Março/2001

Orientadores: Luci Pirmez, D.Sc.
Oswaldo Vernet de Souza Pires, D.Sc.
Luiz Fernando Rust da Costa Carmo, Dr. UPS.

Programa: Informática

Esta monografia descreve e valida uma arquitetura que possibilita o tráfego *multicast* em redes que oferecem serviços diferenciados visando à criação de uma infra-estrutura de Internet para projetos de educação a distância que fazem uso intensivo de aplicações multimídia e multiponto.

ABSTRACT

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

A Propose for Multicast Traffics in the Differentiated Services Networks

Jefferson Manhães de Azevedo

March/2001

Advisors: Luci Pirmez, D.Sc.
Oswaldo Vernet de Souza Pires, D.Sc.
Luiz Fernando Rust da Costa Carmo, Dr. UPS.

Department: Informatics

This work describes an architecture to use multicast with differentiated services networks. That architecture aims at creating an Internet infrastructure for distance learning projects where multimedia and multipoint applications, like multimedia teleconferences and retrieving multimedia documents, are used.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

Introdução

1.1 - Considerações Iniciais

A infra-estrutura de transporte de dados da Internet é baseada no protocolo IP. Este protocolo possui um modelo de serviços de entrega de pacotes pelo melhor esforço, conhecido como *best-effort* que não oferece garantias nem quanto à entrega dos pacotes de dados ao destino nem quanto ao atraso ou a variação do atraso (*jitter*) na entrega. Além disso, neste modelo, todos os fluxos de dados são tratados de uma mesma forma e, portanto, com uma mesma prioridade. Estas características do bem sucedido protocolo IP simplificam os aspectos referentes à infra-estrutura de funcionamento da Internet, além de torná-la bastante flexível. Por um outro lado, estas características do IP são extremamente prejudiciais para os fluxos dos aplicativos multimídia. Estes fluxos são muito sensíveis ao atraso e ao *jitter* sofrido pelos seus pacotes de dados, quando estes são transportados em uma rede. Em redes onde a utilização da largura de banda disponível é baixa, o desempenho do IP torna-se bastante adequado, o que não é verdade em redes sobrecarregadas, como a Internet.

A fim de propiciar diferentes níveis de QoS ao protocolo IP, o IETF propõe o modelo de Serviços Integrados (Intserv) [BCS94] e o modelo de Serviços Diferenciados (Diffserv) [BBC98]. Estes modelos não visam à substituição do atual modelo de serviços da Internet, mas acrescentar novos serviços e, conseqüentemente, diferentes níveis de QoS ao protocolo IP. No modelo Intserv, a garantia de QoS é dada a cada fluxo individualmente, necessitando armazenar nos roteadores informações de reserva para cada um dos fluxos. Já no modelo Diffserv, a atribuição de QoS é feita para os agregados de fluxos, que são fluxos de dados que devem receber um mesmo tratamento em seu percurso através de uma rede. Neste modelo, só há necessidade do armazenamento de informações para os agregados de fluxos, tornando-o mais adequado para redes de longo alcance (WAN), como a Internet, onde há uma grande quantidade de fluxos de dados simultâneos.Vale ressaltar que existem propostas de agregação de

fluxos para o modelo Intserv [BV98]. Neste trabalho o termo fluxo de dados refere-se ao conjunto de pacotes oriundos de uma mesma aplicação ou usuário.

Apesar de as aplicações tradicionais na Internet envolverem comunicações entre dois computadores, há algumas aplicações emergentes, como a tele-conferência de vídeo/áudio e a recuperação de documentos multimídia, que requerem comunicação multiponto. Para que estas aplicações possam ter um desempenho satisfatório na Internet, há a necessidade do uso do roteamento *multicast*, que permite a comunicação de uma fonte de dados e diversos receptores, sem a necessidade do envio de uma cópia de cada pacote para cada um dos receptores, otimizando o uso da largura de banda da rede. O uso do tráfego *multicast*, aliado a uma rede que oferece serviços diferenciados, pode melhorar bastante o desempenho das aplicações multimídia multiponto, fundamentais para projetos de educação à distância e tele-medicina.

Diversas propostas de implementação, com alguns experimentos e testes já em andamento, vêm sendo desenvolvidas para o tráfego *unicast* no modelo Diffserv. No entanto, há apenas uma proposta para o tráfego *multicast* neste modelo, feita por Bless e Wehrle [BW00], além das considerações apresentadas na RFC 2475 [BBC98] que define o modelo Diffserv. Na proposta de Bless e Wehrle, é mencionada a necessidade de um processo de admissão de um novo receptor, fundamental para a reserva de recursos, mas tal processo não é suficientemente detalhado. A proposta também apresenta um mecanismo de remarcação de pacotes *multicast* no interior da rede que possibilita a atribuição de classes de serviços de nível de exigência menor do que o necessário para o fluxo gerado pela fonte de um grupo *multicast*. Apesar de este mecanismo ser bastante flexível, ele apresenta uma maior complexidade no processo de policiamento dos pacotes (adequação dos pacotes ao novo tipo de serviço) nos roteadores do interior da rede. Além disso, a proposta apresenta a necessidade do uso de um novo tipo de tráfego, o tráfego LBE [BW99a], a fim de não comprometer o tráfego *best-effort*. Este tipo de tráfego parece desnecessário, visto que nenhuma garantia é oferecida para o tráfego *best-effort* no ambiente Diffserv.

O objetivo deste trabalho é propor uma arquitetura para o uso do tráfego multicast no Modelo de Serviços Diferenciados, com um nível de detalhamento maior do que o apresentado na proposta de Bless e Wehrle, além de apresentar um processo de remarcação de pacotes que torna mais simples o policiamento dos pacotes no interior da

rede. A proposta também permite a reserva de recursos antecipada, sendo bastante adequada a projetos de educação à distância, onde sessões *multicast* são normalmente agendadas com antecedência.

Alguns dos trabalhos importantes para a elaboração desta arquitetura merecem destaque. O primeiro foi o trabalho descrito em [NJZ99] que apresenta os conceitos de "*Bandwidth Brokers*" (corretores de bandas) e do serviço *premium*. O segundo, apresentado em [BLB98], descreve a arquitetura e o funcionamento de um Servidor de Reserva Antecipada de Recursos para o modelo Intserv. O último foi o Internet *draft* que apresenta a descrição do protocolo de roteamento *multicast* PIM-SSM [HC00].

1.2 - Organização do Trabalho

Este trabalho está estruturado em 5 capítulos. O primeiro capítulo apresenta algumas noções introdutórias, mostrando os objetivos do trabalho e trabalhos correlatos.

O segundo capítulo apresenta a base teórica que fundamenta a proposta, apresentando os conceitos do modelo Diffserv, do tráfego *multicast* e as considerações da sua implementação no modelo Diffserv, bem como das políticas de escalonamento e descarte, fundamentais para o tratamento diferenciado dos pacotes nos roteadores.

O terceiro capítulo apresenta a descrição da arquitetura proposta, o seu funcionamento e os detalhes de sua implementação.

O quarto capítulo apresenta o ambiente elaborado para os testes de desempenho do ambiente de serviços diferenciados que compõe a arquitetura, os detalhes da implementação da ferramenta especialmente desenvolvida para estes experimentos e os resultados obtidos.

E, finalmente, as perspectivas e conclusões mais relevantes são destacadas no quinto capítulo.

CAPÍTULO 2

Modelo de Serviços Diferenciados, Tráfego Multicast e Políticas de Escalonamento e Descarte

2.1- Introdução

O modelo de serviços *best-effort* do protocolo IP não oferece nenhum tipo de garantia quanto à entrega dos pacotes de dados ao seu destino. Em redes com baixa utilização, o desempenho do protocolo IP no transporte de fluxos de dados multimídia é bastante satisfatório. O mesmo não ocorre em redes sobrecarregadas, como a Internet.

O baixo desempenho do protocolo IP resulta do alto índice de congestionamento nos roteadores, que causam grandes atrasos de enfileiramento e uma alta taxa de descarte dos pacotes. No início de 1986, Van Jacobson desenvolveu um mecanismo [J88] para minimizar os congestionamentos ocorridos em redes sobrecarregadas, visando à melhoria do desempenho do transporte de seus dados. O mecanismo proposto por Jacobson torna a fonte de dados sensível a qualquer congestionamento que ocorre no percurso dos seus dados na rede. Assim que a fonte de dados percebe que está havendo perda de pacotes, ela diminui a sua taxa de transmissão e, aos poucos, torna a elevá-la. Este mecanismo é utilizado atualmente pelo protocolo TCP, sendo ele o responsável por não permitir que a Internet atual entre em um colapso total.

Apesar de importante, este mecanismo utilizado pelo TCP não é suficiente para permitir um bom desempenho do transporte de dados multimídia na Internet, pois ele está baseado apenas nos nós de extremidade da rede. Como complemento, é necessário que haja um tratamento específico e diferenciado dos fluxos multimídia nos nós intermediários da rede. Com este objetivo, o IETF propôs dois novos modelos de serviços de entrega: o modelo de serviços integrados (Intserv) e o modelo de serviços diferenciados (Diffserv). Estes modelos não visam à substituição do modelo de serviços *best-effort* da Internet atual, mas acrescentar a ele outros serviços.

Uma outra maneira de propiciar um melhor desempenho nas redes é o uso do roteamento *multicast.* O roteamento *multicast* permite a comunicação de uma fonte de

dados e diversos receptores, sem a necessidade de enviar uma cópia do pacote para cada um dos receptores, otimizando o uso da largura de banda da rede.

A primeira seção deste capítulo apresenta os conceitos e a descrição do modelo Diffserv. A segunda seção apresenta conceitos sobre roteamento *multicast* e algumas propostas para o tráfego *multicast* no Diffserv. As duas últimas seções descrevem as políticas de escalonamento e descarte, que possibilitam a obtenção de serviços de melhor qualidade que o *best-effort* do protocolo IP, sendo amplamente utilizados no modelo Diffserv.

2.2- Modelo de Serviços Diferenciados (Diffserv)

O modelo Diffserv possibilita diferentes níveis de QoS na Internet, sem a necessidade de dar um tratamento específico a cada fluxo de dados, como no modelo de serviços integrados (Intserv) [BCS94]. No modelo Intserv, o protocolo de reserva RSVP (*Resource ReSerVation Protocol*) [BZB97] torna as aplicações capazes de fazerem uma sinalização (troca de informações de controle) para cada um de seus fluxos. Esta sinalização armazena informações de estado para cada fluxo em todos os roteadores por onde estes fluxos trafegarão. Esta abordagem permite um tratamento mais individualizado para cada fluxo de dados que trafega na rede. Além disso, é possível atender com um alto grau de precisão os requisitos de fluxos de dados multimídia, como o atraso máximo que um pacote pode sofrer. Por outro lado, esta abordagem aumenta em muito a sobrecarga de processamento nos roteadores que estão no interior da rede, principalmente em redes de longo alcance, como a Internet. Nestas redes, o número de fluxos de dados que passam pelos roteadores dos seus *backbones* pode chegar à ordem de milhões de fluxos. Armazenar um estado de reserva para cada um destes fluxos, a fim de dar um tratamento individualizado aos mesmos, pode inviabilizar a oferta de serviços com QoS em redes de longo alcance.

Uma nova abordagem na reserva de recursos para a Internet é oferecida pelo modelo de serviços Diferenciados (Diffserv) [BBC98]. Neste modelo de serviços de entrega de pacotes, os fluxos são agrupados, formando agregados de fluxos, e identificados em algumas classes de serviços preestabelecidas (CoS - *Class-of-Service*). Desta maneira, somente as informações de estado destas classes de serviço são

armazenadas nos roteadores, permitindo que cada agregado de fluxos receba um tratamento diferenciado em cada roteador no interior de uma rede. A grande vantagem desta abordagem é o menor número de estados de reserva necessários nos roteadores, sendo mais adequado para redes de longo alcance. Uma desvantagem é a impossibilidade de garantias precisas no atendimento aos requisitos dos fluxos de dados. Em [BV98], é descrita uma proposta de agregação de estados de reserva para o protocolo RSVP, tornando o modelo Intserv apropriado às redes WAN.

A seção 2.1 apresenta os conceitos fundamentais do modelo Diffserv. Na seção 2.2 são descritos os mecanismos de classificação, condicionamento e escalonamento de pacotes. Na seção 2.3 são apresentados os serviços oferecidos pelo Diffserv.

2.2.1- Conceitos Fundamentais do Diffserv

Uma porção da Internet formada por uma ou mais redes cujos nós (roteadores e *hosts*) implementam uma mesma política de tratamento diferenciado aos agregados de fluxos é chamada de domínio DS (*Differentiated Service*). Um domínio DS possui seus limites definidos por nós de extremidade (roteadores de borda). Estes nós são os responsáveis pela política de classificação, condicionamento e agregação dos fluxos de dados que entram e saem de um domínio. Nestes nós, os fluxos recebem um policiamento e uma modelagem, caso estejam fora do perfil de tráfego, que é previamente estabelecido por um contrato de fornecimento de serviços (SLA - *Service Level Agreement*). Neste trabalho, consideramos um domínio DS sendo composto por uma rede provedora de serviços diferenciados e várias redes clientes. Um conjunto de domínios DS interligados forma uma região DS. A Figura 2.1 mostra uma região DS.

Os fluxos que entram em um domínio DS devem ter seus pacotes de dados identificados por um código, chamado *codepoint*. O *codepoint* permite que os pacotes de dados recebam um tratamento diferenciado em cada nó do domínio a fim de oferecer um determinado serviço. O *codepoint* é definido em um campo do cabeçalho IP, denominado "DS *Field*" [NBB98], que substitui as definições existentes do octeto Tipo de Serviço (*Type of Service*) do IPv4 [ISI81] e o *Traffic Class Octet* do IPv6 [DH97]. O campo *"DS Field"* vazio ou com o valor zero caracteriza o serviço de entrega *best-effort*.

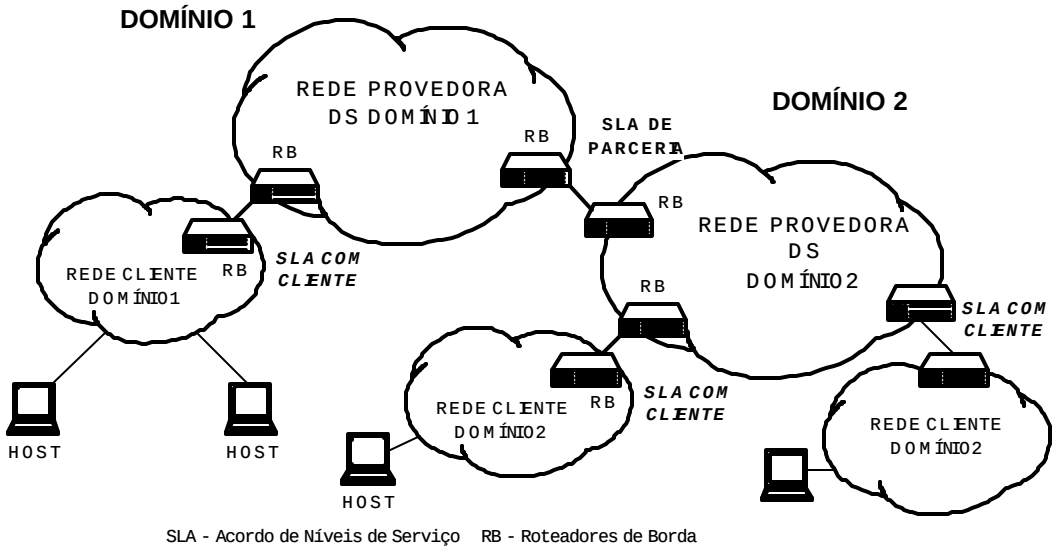


Figura 2.1 – Região DS

O *Per-Hop-Behavior* (PHB) é uma descrição do comportamento de envio dos nós de um Domínio DS aplicado aos pacotes de dados. O conceito de comportamento difere do conceito de serviço, pois o primeiro refere-se a um comportamento local (em cada nó de um Domínio DS), enquanto o segundo refere-se ao comportamento fim-a-fim (ao longo de todos os nós no percurso dos pacotes dentro de um domínio DS). A capacidade em oferecer diferentes PHBs está relacionada à quantidade de recursos que os nós disponibilizam para cada classe de serviço, das políticas de enfileiramento dos pacotes e das políticas de gerenciamento de *buffers* destas filas. As características comportamentais de um PHB são padronizadas e não estão associadas a tipos específicos de algoritmos ou mecanismos utilizados em sua implementação. Os PHBs podem ser especificados individualmente ou em grupos. O modelo Diffserv define ainda um PHB *default*. Este PHB é dado a todos os pacotes que possuem um *codepoint* desconhecido pelos nós do domínio DS. Geralmente, este PHB *default* é equivalente a um serviço *best-effort*.

Além de ter os seus pacotes identificados, os fluxos ou agregados de fluxos de dados que entram em um domínio DS devem ser condicionados, de tal forma que as suas características estejam de acordo com o perfil de tráfego previamente estabelecido no SLA. Um perfil de tráfego especifica as propriedades temporais que um determinado fluxo ou agregado de fluxos deve ter. Além disso, o perfil fornece as regras que determinam a pertinência de um pacote, bem como as diferentes ações de

condicionamento que podem ser aplicadas aos pacotes em cada uma destas duas condições. Os pacotes dentro do perfil podem receber permissão de entrar no domínio sem condicionamento ou, caso necessário, ter o valor do seu *codepoint* alterado (remarcados) a fim de se adequar aos valores de *codepoints* do novo domínio (um mesmo serviço pode ser identificado por diferentes valores de *codepoints* em domínios adjacentes). Por outro lado, os pacotes fora do perfil podem ser enfileirados, até que estejam de acordo com o perfil (modelagem), descartados (policiamento), marcados com um novo valor de *codepoint* ou transmitidos inalterados. Os pacotes fora de perfil podem ser mapeados para um serviço de entrega com qualidade inferior ou para o serviço de entrega *best-effort.*

A política de condicionamento é de responsabilidade dos nós extremos. Ela é estabelecida por intermédio de um Acordo de Condicionamento de Tráfego (TCA - *Traffic Conditioning Agreement*), que é parte do SLA. O TCA especifica a classificação dos pacotes de dados, a política de marcação/remarcação dos *codepoints*, os perfis (características) de tráfego permitidos, bem como as ações necessárias que são aplicadas aos fluxos que estão dentro ou fora destes perfis. Os nós de saída do domínio, de forma análoga, podem também aplicar um condicionamento aos fluxos saindo do domínio em direção a um domínio adjacente. A idéia por trás da arquitetura Diffserv é simplificar ao máximo os nós do interior do domínio (sem políticas de condicionamento), concentrando toda sua complexidade em nós extremos.

A seguir serão apresentados os elementos necessários à implementação dos PHBs nos nós de um domínio DS.

2.2.2- Classificação, Condicionamento e Escalonamento de Tráfego

Todos os fluxos ou agregados de fluxos de dados que entram em um domínio devem ter os seus pacotes classificados, condicionados e escalonados. A classificação identifica a classe de serviço à qual o pacote pertence a fim de encaminhá-lo ao módulo condicionador. O condicionador tem a função de adequar o fluxo ou o agregado ao perfil previamente estabelecido no TCA. O escalonador é o responsável pelo envio dos pacotes de acordo com as regras de escalonamento. A Figura 2.2 mostra o diagrama de blocos representando estes três módulos.

Figura 2.2 – Diagrama de Blocos de um Classificador, Condicionador e Escalonador de Tráfego

Estas funcionalidades devem ser implementadas nos nós de um domínio a fim de que o mesmo possa oferecer os serviços diferenciados. A descrição de cada módulo é descrita em seguida.

2.2.2.1- Classificadores

Os classificadores de pacotes selecionam os pacotes, que entram em um domínio DS, a partir do conteúdo de alguma porção do seu cabeçalho. São possíveis dois tipos de classificação. A primeira é a classificação baseada no valor do *codepoint* do campo "DS *Field*" do cabeçalho do pacote (BA - *Behavior Agregate*). A segunda baseia a seleção em um ou mais campos do cabeçalho do pacote (MF - *Multi Field*), tais como: endereço da fonte, endereço do destino, número da porta origem, número da porta de destino, etc. Geralmente, a classificação MF é feita nos nós de entrada dos domínios, enquanto a classificação BA é feita em seus nós interiores. Desta forma, há uma simplificação e, conseqüentemente, uma menor sobrecarga de processamento na classificação dos pacotes no interior dos domínios.

Após a seleção, os classificadores conduzem os pacotes selecionados para o módulo condicionador de tráfego. Os classificadores devem ser configurados por algum procedimento de administração, de acordo com o TCA acordado.

2.2.2.2- Condicionador de Tráfego

O condicionador de tráfego é o elemento responsável pela adequação de um fluxo ou agregado de fluxo de dados, entrando em um domínio, a um determinado perfil estabelecido no TCA. Ele é composto pelos seguintes elementos: medidor, marcador, modelador e descartador. Há situações onde alguns dos elementos do módulo

condicionador não são necessários, com é o caso dos roteadores internos do modelo Diffserv. Detalhes sobre a implementação dos roteadores Diffserv são apresentados no capítulo 3.

Após a classificação, os pacotes de um fluxo de dados são conduzidos para o módulo condicionador. Um medidor de tráfego pode ser utilizado para medir alguma característica temporal do fluxo ou agregado de fluxo de dados, a fim de verificar se ele está dentro ou fora do perfil estabelecido. O resultado desta comparação pode ser usado para uma ação de marcação/remarcação, descarte ou modelagem. A Figura 2.3 mostra o diagrama de blocos de um classificador e do condicionador de tráfego.

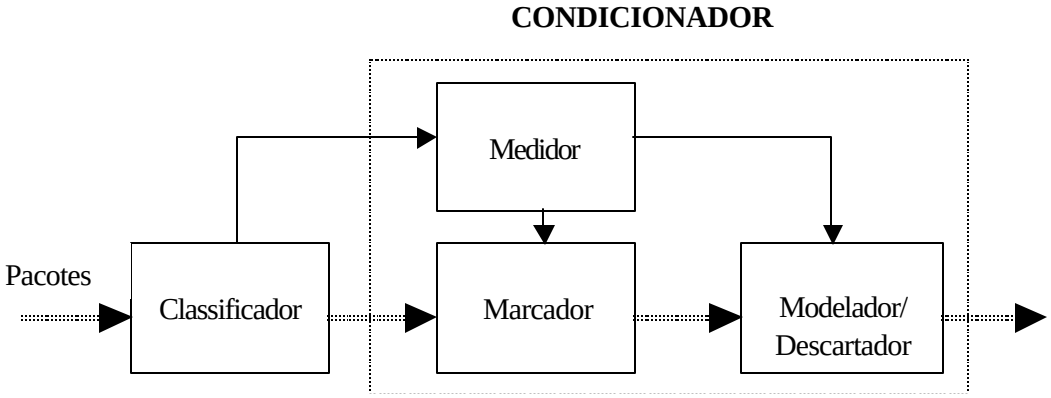


Figura 2.3 – Diagrama de blocos de um Classificador de Pacote e do Condicionador de Tráfego

O módulo marcador é o responsável pela atribuição de valores de *codepoints* aos pacotes dos fluxos e agregados de fluxos de dados, entrando em um domínio DS, segundo a classe de serviços para eles reservada. Três situações de marcação são possíveis. A primeira ocorre quando um fluxo ou agregado de fluxo de dados entra pela primeira vez em um domínio DS. Nesta situação, todos os pacotes pertencentes ao fluxo ou agregado de fluxos devem receber um valor de *codepoint*. Visando garantir que somente os fluxos ou agregados de fluxos com reservas sejam marcados (recebam um valo de *codepoint*), antes da classificação todos os pacotes devem ser desmarcados. Uma segunda situação de classificação ocorre quando um fluxo ou agregado de fluxo está fora do perfil contratado e, por isso, alguns de seus pacotes devem receber um valor de *codepoint* equivalente a um serviço de menor qualidade. A terceira situação de classificação ocorre quando dois domínios adjacentes utilizam valores de *codepoints* diferentes para um mesmo tipo de serviço. Logo, o marcador deverá atribuir o valor de

codepoint adequado ao fluxo antes que ele entre no próximo domínio. Nestas duas últimas situações o marcador faz a ação de remarcação.

O módulo modelador atrasa alguns ou todos os pacotes de um fluxo ou agregado de fluxo, a fim de ajustá-los a um determinado perfil. Geralmente, o modelador possui um *buffer* de tamanho finito, o que pode acarretar um possível descarte de pacotes. Isto ocorre quando não há espaço suficiente no *buffer* para manutenção do atraso desejado dos pacotes.

O módulo descartador é o responsável pelo descarte de alguns ou todos os pacotes de um fluxo ou agregado de fluxo, visando ajustá-los a um perfil desejado. Este procedimento é conhecido como "policiamento". Um descarte de pacotes pode ser implementado em um modelador com o tamanho do *buffer* ajustado para zero ou para um valor muito baixo.

2.2.2.3- Escalonador de Tráfego

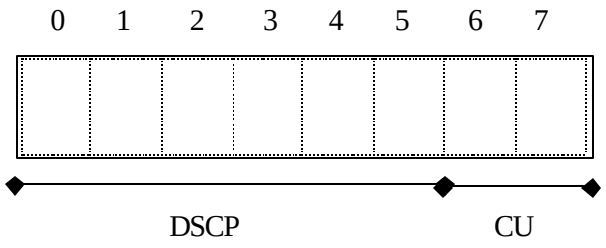
O escalonador é o principal elemento responsável para garantir que os pacotes tenham o devido PHB em cada nó de um domínio. O PHB define a maneira pela qual um nó aloca recursos para os fluxos ou agregados de acordo com sua classe de serviços, podendo ser especificados em termos de recursos alocados com relação a outros PHBs (ex., *buffer*, banda) ou em termos de características de tráfego (ex., *delay*, perda, *jitter*). O modelo Diffserv não define os mecanismos para a implementação dos escalonadores, mas apenas as características esperadas dos PHBs. Uma variedade de mecanismos pode ser adequada na implementação dos escalonadores. As seções 3 e 4 deste capítulo descrevem uma série destes mecanismos.

2.2.3- Serviços Oferecidos pelo Diffserv

Um PHB é selecionado em um nó de um domínio DS pelo *codepoint* de um pacote recebido. Como descrito em [NBB98], o *codepoint* é definido em um campo do cabeçalho IP, denominado "DS *Field*", sendo composto pelos primeiros seis bits do "DS F*ield*", como pode ser visto na Figura 2.4. Os outros dois bits (CU) não são usados e,

portanto, não devem ser utilizados como identificador de PHB pelos nós de um domínio DS.

Existe um conjunto de *codepoints* que são padronizados e relacionados a certos PHBs. Há um outro conjunto, sem padronização, que têm um significado e um mapeamento a nível local (particular a um domínio). Todos os *codepoints* com significado para um domínio DS devem estar mapeados para algum PHB, mas um mesmo PHB pode ser mapeado por diversos *codepoints* diferentes. Caso o *codepoint* de um pacote não tenha um mapeamento para um PHB, não tenha significado para aquele domínio DS, os nós do domínio deverão dar um tratamento *default* a este pacote (PHB *default*).



DSCP (*Differentiated Services Codepoint*): *codepoint* de Serviços Diferenciados

CU (*Currently Unused*): Atualmente sem uso

Figura 2.4- Estrutura do "DS *Field*"

O *codepoint* deve ser tratado como um índice que selecionará o PHB correspondente em cada nó por onde o pacote passar. Cada nó dentro do domínio deve ter uma tabela que relaciona um *codepoint* a um PHB. O *codepoint* não possui nenhuma estrutura hierárquica. Um *codepoint* pode estar mapeado para um determinado PHB dentro de um domínio DS e ser mapeado para um outro PHB em um domínio DS adjacente. Neste caso, para a compatibilização do tratamento dado aos pacotes que transitam entre domínios adjacentes, há a necessidade de uma remarcação dos *codepoints* nos nós de extremidade. Esta remarcação deve estar definida no TCA do SLA interdomínio.

A seguir são apresentados os grupos de PHBs padronizados, bem como o conjunto de *codepoints* reservados para manter a compatibilidade retroativa com o campos de precedência do protocolo IP.

2.2.3.1- Classe de Precedência

Em [NBB98] é definida uma classe de serviço 'xxx000' (o xxx refere-se aos valores do campo de precedência do cabeçalho IP), que visa a compatibilizar o uso corrente do campo de precedência do IP (IP *Precedence*) [ISI81]. Atualmente, o campo de precedência do IP possui um amplo uso, definindo tratamentos especiais que são dados aos pacotes por ele identificados [B95]. Este uso será preservado. O que não será mantido é a compatibilidade com os bits "DTR" definidos na [ISI81]. Na Tabela 2.1 é mostrada a codificação dos bits "IP *Precedence*" no TOS do IPv4:

Bits	Descrição
111	*Network Control*
110	*Internetwork Control*
101	CRITIC/ECP
100	*Flash Override*
011	*Flash*
010	*Immediate*
001	*Priority*
000	*Routine*

Tabela 2.1- Codificação dos Bits "IP *Precedence*" no TOS do Ipv4

A classe de precedência é uma composição dos três bits indicado na Tabela 2.1 com os outros três bits da definição da classe. Além disso, cada domínio deve ter um *codepoint* relacionado a um comportamento *default*. Normalmente, o comportamento *default* é o *best-effort* e tem *codepoint* recomendado igual a '000000'. Caso algum nó receba um *codepoint* inválido, o pacote deve receber um tratamento equivalente ao comportamento *default* do domínio. Neste caso, não deve haver uma remarcação do "*DS Field"* para o "*codepoint default*".

2.2.3.2- Grupo de PHB AF (*Assured Forwarding*)

O grupo de PHB AF [HBW99] permite a entrega de pacotes IP em quatro classes distintas, onde cada classe pode ter até três níveis diferentes de prioridade de descarte. A Tabela 2.2 apresentada uma tabela dos *codepoints* referentes às classes AF.

Precedência	**Classe**			
	1	2	3	4
Baixa	010000	011000	100000	101000
Média	010010	011010	100010	101010
Alta	010100	011100	100100	101100

Tabela 2.2- *Codepoints* Referentes as Classes AF

Em cada nó do domínio, o nível de certeza de entrega de um pacote depende dos seguintes itens:

- quantidade de recursos reservados para a mesma classe AF do pacote;
- carga atual da classe;
- e em caso de congestionamento, da prioridade de descarte do pacote.

O grupo de PHBs AF não asseguram nenhum tipo de quantificação temporal, como *delay* ou *jitter*. Além disso, os nós de um domínio não podem reordenar os pacotes de um fluxo, quando eles pertencerem a uma mesma classe AF.

2.2.3.3- Grupo de PHB EF (*Expedited Forwarding*)

O grupo de PHB EF [JNP99] pode ser usado para construir um serviço fim-a-fim de baixa perda, baixa latência, baixo *jitter* e com uma largura de banda garantida em um domínio. Este serviço emula uma linha privativa de comunicação de dados virtual, para os nós das extremidades da comunicação, como no serviço *Premium* descrito em [NJZ99]. Conseqüentemente, os fluxos que recebem um tratamento EF devem ter um *delay* de enfileiramento baixíssimo. Isto é, não deve haver filas de pacotes para este serviço nos nós do domínio DS ou, se existir, elas devem ser muito pequenas. Para isso:

- os nós devem estar configurados a fim de garantir de que os pacotes dos agregados de fluxos tenham uma taxa de partida mínima bem definida.
- o agregado de fluxos deve ser condicionado de maneira que a sua taxa de chegada em qualquer nó seja freqüentemente menor do que a taxa mínima de partida configurada no nó.

Apenas a primeira parte deve ser garantida pelo PHB EF. A segunda parte deve ser garantida pelos condicionadores de tráfego definido em [BBC98]. O *codepoint* recomendado para o PHB EF é "101110".

2.3- Disciplina de Escalonamento

A disciplina de escalonamento é o elemento principal para a obtenção de diferentes níveis de QoS em uma rede, sendo o responsável pelo tratamento diferenciado que os nós de um domínio DS dão aos pacotes de um fluxo ou agregado de fluxos de dados. Uma disciplina de escalonamento é composta por dois componentes. O primeiro, é o responsável pela ordem de envio dos pacotes (política de escalonamento), definindo a ordem em que os pacotes serão ordenados na fila de espera e pela escolha do próximo pacote que deve ser enviado. O tempo médio que um pacote leva na fila, esperando para ser enviado, é conhecido como atraso de enfileiramento (*queuing delay*). O segundo, é o responsável pela gerência dos *buffers* disponíveis para o armazenamento dos pacotes que estão esperando para ser enviados. Quando não há mais *buffers* para o armazenamento de um novo pacote que acaba de chegar, deve haver uma estratégia para descartar algum pacote. Esta estratégia é denominada política de descarte, sendo responsável pela taxa de descarte de pacotes (*loss rate*).

A disciplina de escalonamento é implementada no módulo de envio dos pacotes à rede (escalonador). Nesta seção são descritos as características das disciplinas de escalonamento, alguns exemplos de disciplinas de escalonamento sugeridas para a implementação do modelo Diffserv e algumas políticas de descarte de pacotes. No Diffserv, os recursos compartilhados são normalmente a largura de banda dos enlaces e os *buffers* nos roteadores. Portanto, as disciplinas de escalonamento discutidas a seguir

dão mais ênfase ao compartilhamento destes recursos, o que não inviabiliza sua aplicação no compartilhamento de outros tipos de recursos.

A seção 2.3.1 apresenta a Lei de Conservação das disciplinas de escalonamento. Na seção 2.3.2 são descritos os requisitos para as disciplinas de escalonamento. Na seção 2.3.3 são apresentadas algumas considerações sobre o projeto das disciplinas. Na seção 2.3.4, são apresentadas alguns exemplos de disciplinas, onde a maior parte delas foram implementadas nos roteadores que compõem a arquitetura proposta neste trabalho. Por último, na seção 2.3.5 serão apresentadas algumas estratégias para o descarte dos pacotes. Outras informações podem ser obtidas em [K98].

2.3.1- Lei da Conservação

Uma disciplina de escalonamento bem conhecida é a *First-Come-First-Serve* (FCFS), em que pacotes são transmitidos na ordem de sua chegada, descartando os que chegam quando a fila estiver cheia. Por isso, a disciplina FCFS não possibilita diferenciar o tratamento de pacotes dos fluxos de diferentes conexões, conseqüentemente, não permite diferenciar o atraso sofrido pelos pacotes destes fluxos. Apesar de outras disciplinas de escalonamento permitirem tal diferenciação, o teorema da Lei de Conservação afirma que a soma da média dos atrasos de enfileiramento sofridos por um conjunto de fluxos multiplexados, ponderada pelas suas proporções de carga no enlace, é independente da disciplina de escalonamento utilizada. Em outras palavras, uma disciplina de escalonamento só poderá reduzir o atraso de enfileiramento de um determinado fluxo de dados, comparado com a disciplina FCFS, às custas de outros fluxos.

2.3.2- Requisitos para as Disciplinas de Escalonamento

As disciplinas de escalonamento devem satisfazer os seguintes requisitos:

- **Facilidade de Implementação**

Em redes de alta velocidade, o tempo gasto para tomar a decisão de qual é o próximo pacote a ser enviado é muito pequeno. Por isso, a disciplina de escalonamento de pacotes deve ser constituída de poucas e simples operações, devendo ser a mais independente possível do número de fluxos servidos simultaneamente e, se possível, de fácil implementação em *hardware*.

- **Justiça**

Uma disciplina de escalonamento aloca uma parte dos recursos compartilhados para cada fluxo que ela está servindo. Esta alocação é justa quando é atendido o critério de alocação "max-min". Este critério tem por objetivo tornar o compartilhamento dos recursos insuficientes (entre fluxos que possuem os mesmos direitos, mas demandas diferenciadas) o mais justo possível. Os fluxos com menor demanda de recursos serão atendidos completamente. O restante dos recursos é dividido entre os fluxos com maior demanda. Resumidamente, o compartilhamento justo "max-min" é um tipo de compartilhamento, onde:

- os recursos são alocados em ordem crescente de demanda;
- nenhum fluxo recebe uma porção do recurso maior do que necessita;
- os fluxos com demandas não satisfeitas recebem uma parte igual de recurso.

A alocação de recursos justa para um conjunto de fluxos de dados é um objetivo global, uma vez que a disciplina de escalonamento toma somente decisões locais. A fim de transladar uma decisão local para uma decisão global, cada fluxo deve ter limitado o uso de recursos à menor alocação justa local ao longo de todo o seu percurso.

No modelo Diffserv, a disciplina de escalonamento deve ser justa para os fluxos de dados pertencentes a um mesmo agregado e injustas para os fluxos pertencentes a diferentes agregados, visto que cada agregado de fluxo possui requisitos diferenciados.

- **Proteção**

Uma disciplina de escalonamento que oferece proteção aos fluxos das conexões deve impedir que um fluxo mal comportado de uma conexão influencie na alocação de recursos das demais conexões. Na disciplina FCFS isto não é possível, pois uma rajada de um fluxo aumenta o atraso de enfileiramento de todos os outros fluxos, além de causar um aumento na taxa de descarte de seus pacotes. Uma disciplina de escalonamento que oferece proteção aos fluxos das conexões garante um mínimo de quantidade de recursos para todas as conexões. No modelo Diffserv a proteção se dá entre os diversos agregados de fluxos no interior de um domínio DS e as conexões individuais em alguns roteadores de borda.

- **Limites de Desempenho**

Uma disciplina de escalonamento deve permitir que determinadas conexões possam arbitrariamente, de acordo com o administrador da rede, ter limites de desempenhos diferenciados para seus fluxos (estas diferenciações estão limitadas pela Lei de Conservação). Tais limites são possíveis através de reservas de recursos. Uma vez que a quantidade de recursos reservados para os fluxos de uma conexão depende da sua intensidade, os fluxos com reservas devem respeitar o uso destes recursos. Para isso, um acordo deve ser estabelecido entre o usuário da aplicação que gera o fluxo e o administrador da rede. Este acordo deve estipular um limite do tráfego para o usuário. Se este limite for respeitado, a rede garante o desempenho acordado. No modelo Diffserv este acordo deve ser obtido através do SLA.

- Facilidade e Eficiência no Controle de Admissão

O processo de controle de admissão deve garantir que a aceitação de uma nova conexão não comprometa o desempenho já estabelecido para os fluxos das conexões já admitidas. Esta decisão deve ser tomada com base no uso corrente dos recursos e no descritor do fluxo da nova conexão. Além disso, a disciplina de escalonamento não deve permitir uma má utilização dos recursos da rede.

2.3.3- Projetos de Disciplinas de Escalonamento

A seguir são apresentados alguns graus de liberdade possíveis no projeto de disciplinas de escalonamento.

- **Níveis de Prioridade**

Nos projetos de escalonadores com prioridade, os fluxos de dados de cada conexão possuem níveis de prioridade associados. Desta forma, o escalonador pode oferecer aos pacotes dos fluxos das conexões de maior prioridade um menor atraso de enfileiramento, em detrimento dos pacotes das conexões com níveis de prioridade menores. Um escalonador pode ter diversos níveis de prioridade. Na prática, este número está relacionado ao número de classes de atraso de enfileiramento que o administrador da rede deseja oferecer.

Um projeto de escalonador com prioridades pode permitir que os fluxos de dados de usuários mal comportados em conexões de alta prioridade comprometam os recursos disponíveis para as conexões de menor prioridade. Dependendo da intensidade destes fluxos de dados poderá haver um crescimento do atraso de enfileiramento e uma diminuição da largura de banda disponível para as conexões de menor prioridade. Com isso, pode ser criada uma situação de adiamento indefinido, em que o escalonador nunca envia os pacotes das conexões de menor prioridade enquanto houver pacotes nas conexões de níveis mais altos de prioridade. Portanto, para que esta situação seja evitada, em escalonadores com prioridades, são críticos o controle de admissão e o policiamento das conexões**.**

- **Disciplinas de Trabalho Conservativo versus Trabalho não Conservativo**

As disciplinas de escalonamento de trabalho conservativo são aquelas em que o escalonador só está em estado ocioso (*idle*) quando não houver ninguém na fila de espera. Por outro lado, as disciplinas de trabalho não conservativo podem entrar em

estado ocioso mesmo quando houver requisições de serviço na fila de espera. À primeira vista não há sentido em projetar escalonadores não conservativos. Mas em muitos casos estas disciplinas são fundamentais, visto que proporcionam a modelagem e reconstrução dos fluxos, tornando-os mais previsíveis. Assim, pode-se reduzir tanto o tamanho dos *buffers* nas filas dos equipamentos na direção descendente do fluxo (*downstream*) quanto a variação do atraso (*jitter*) experimentado pelos pacotes.

2.3.4- Exemplos de Disciplina de Escalonamento

Nesta seção serão apresentados alguns exemplos de disciplinas de escalonamento. A implementação destas disciplinas corresponde ao módulo escalonador apresentado na estrutura funcional dos roteadores de borda e internos do modelo Diffserv. As disciplinas descritas são: Compartilhamento de Processador Genérico, *Weighted Round-Robin*, *Deficit Round-Robin*, *Weighted Fair Queuing, Self-Clocked Fair Queuing* e *Start-Time Fair Queuing*.

2.3.4.1- Compartilhamento de Processador Genérico (GPS)

A disciplina de escalonamento que atende completamente ao critério de justiça "max-min" é a disciplina de Compartilhamento de Processador Genérico (GPS – *Generalized Processor Sharing*). Esta disciplina serve cada pacote como se ele estivesse em uma fila logicamente separada, visitando cada uma das filas não vazias de cada vez e servindo cada pacote na fila em uma porção de tempo infinitesimal. A disciplina GPS é utilizada no escalonamento de processos em sistemas operacionais. O problema é que processos podem ser servidos em um pequeno intervalo por vez. Esta abordagem é inviável para o serviço de transmissão de pacotes, pois estes devem ser servidos em sua totalidade.

A disciplina GPS permite que cada conexão tenha um peso, de forma que os fluxos das conexões com maior peso tenham um maior tempo de serviço. Além de ser justa, a disciplina GPS também oferece proteção aos fluxos. Infelizmente, não é possível implementar esta disciplina, mas ela serve como uma referência de comparação com relação ao nível de justiça e proteção para as demais disciplinas de escalonamento.

2.3.4.2- *Weighted Round-Robin* (WRR)

Uma simulação simples da disciplina GPS é a disciplina *Roundin-Robin* (RR), que serve um pacote de cada vez das filas que não estão vazias. Cada fila corresponde a um fluxo de uma conexão ou agregado de fluxos. A porção de tempo de serviço infinitesimal da disciplina GPS é substituída pelo serviço de um pacote por vez.

A disciplina WRR é a disciplina RR usando filas ponderadas (utilizam intervalos de tempo de serviços diferenciados), permitindo que o atendimento dos pacotes seja proporcional aos pesos dados a cada fila. Se os pacotes das filas possuem tamanhos diferentes, o que acontece na Internet, o servidor da disciplina de escalonamento WRR faz a divisão de cada peso, correspondente a cada fila, pela média do tamanho dos pacotes na fila. Com isso, obtém-se um conjunto normalizado de pesos. Por exemplo, supondo que haja três classes de serviço com pacotes de um mesmo tamanho, mas com pesos respectivamente igual a '0,5', '0,75' e '1,0'. Normalizando os pesos para que eles se tornem inteiros, o número de pacotes servido de cada vez nas filas será respectivamente 2,3 e 4. Supondo também que a média do tamanho de pacotes seja respectivamente 50, 500 e 1500 bytes. Dividindo o peso pela média do tamanho dos pacotes, obtêm-se os pesos normalizados iguais a 0,01, 0,0015 e 0,000666. Normalizando os pesos para valores inteiros, obtêm-se respectivamente 60, 9 e 4. Assim, o servidor irá servir 60 pacotes da primeira fila, 9 da segunda e 4 da terceira em cada rodada.

A WRR apresenta dois problemas quando trata de fluxos com pacotes de tamanhos diferentes. O primeiro corresponde à dificuldade de prever antecipadamente a média do tamanho dos pacotes das filas, podendo causar uma alocação de banda injusta. O segundo problema é que a WRR só é justa para intervalos de tempo muito maiores do que o intervalo de tempo de cada rodada. Em pequenos intervalos de tempo, algumas filas podem obter mais recursos que outras.

2.3.4.3- *Deficit Round-Robin* (DRR)

A disciplina DRR modifica a WRR a fim de possibilitar a manipulação de filas com tamanhos de pacotes variados, sem a necessidade de conhecer estes tamanhos antecipadamente. Na DRR, cada fila possui um contador (*deficit counter*) inicializado em 0. O servidor da disciplina de escalonamento DRR visita uma fila de cada vez, servindo uma quantidade de bits (*quantum*). O primeiro pacote da fila visitada é servido caso ele seja menor ou igual ao tamanho do *quantum*. Se o pacote é maior, o *quantum* é acrescido ao contador da fila. Se o servidor visitar uma fila e a soma do contador da fila e do *quantum* for maior ou igual ao tamanho do pacote, o pacote será servido e o contador da fila será diminuído de um valor igual ao tamanho do pacote. A disciplina DRR também possui uma versão que utiliza pesos, em que o peso de cada fila é multiplicado pelo valor do *quantum*. Para que seja garantido que pelo menos um pacote seja servidor por vez, o tamanho do *quantum* deve ser igual ao maior tamanho de pacote possível.

A grande vantagem desta disciplina é a sua facilidade de implementação. Contudo, como a WRR, esta disciplina é injusta em pequenos intervalos de tempo.

2.3.4.4- *Weighted Fair Queuing* (WFQ)

A disciplina de escalonamento de pacotes WFQ é similar à disciplina de escalonamento conhecida como *Packet-by-packet Generalized Processor Sharing* (PGPS). A PGPS é uma aproximação do funcionamento do escalonamento GPS, sem a necessidade de servir um pacote em um intervalo de tempo infinitesimal. Ela também não necessita do conhecimento antecipado do tamanho médio dos pacotes, no caso de fluxos com tamanhos de pacotes variados.

A disciplina WFQ calcula o tempo que cada pacote levaria para ser servido em uma disciplina GPS. Este tempo é chamado de *finishing time* (tempo de término). A ordem de serviço dos pacotes segue a ordem do *finishing time*. Para desvincular o *finishing time* da sua relação com o tempo de serviço, a disciplina WFQ chama-o de *finishing number*. O *finishing number* é considerado um rótulo utilizado para representar

um índice de ordenação no serviço dos pacotes. Este rótulo induz uma seqüência de serviço, sem nenhuma relação com o tempo-real em que os pacotes são servidos.

O cálculo do *finishing number* depende de uma variável conhecida como *round number* (número de rodadas). Supondo pesos iguais para cada fluxo e o intervalo de tempo infinitesimal de serviço da disciplina GPS equivalente a um bit, o *round number* é igual ao número de rodadas de serviço que um escalonador RR bit-a-bit (servindo um bit por vez) completa em um determinado tempo. Por exemplo, se em um certo momento o *round number* é igual a 6, isto equivale a dizer que o escalonador RR está na sexta rodada.

Na disciplina WFQ uma conexão pode estar ativa ou inativa. Uma conexão é considerada ativa se o *finishing number* de um de seus pacotes, ou do último pacote servido por ela, é maior que o *round number* corrente, e inativa caso contrário. Uma das dificuldades na implementação da disciplina WFQ está relacionada ao cálculo do *round number* corrente, devido a um problema conhecido como deleção iterativa (*iterated deletion*). Este fenômeno ocorre quando uma conexão torna-se inativa, gerando uma desativação em cascata das outras filas.

O cálculo do *finishing number*, conhecendo-se o *round number*, ocorre da seguinte forma:

- Quando um pacote chega a uma conexão inativa, o *finishing number* é a soma do valor corrente do *round number* e do tamanho do pacote em bits. Isto equivale ao número de vezes que um servidor RR bit-a-bit necessita para servir este pacote. Por exemplo, se o *round number* for 3, quando o pacote de tamanho de 10 bits chega, o *finishing number* deste pacote será igual a 13, o que equivale a dizer que este pacote é completamente servido na décima terceira rodada do servidor RR.
- Se um pacote chega a uma conexão considerada ativa, o seu *finishing number* é calculado através da soma do maior *finishing number* dos pacotes da conexão (ou do último pacote servido na conexão) e o tamanho do pacote em bits. O maior *finishing number* da conexão é chamado de *connection finishing number*. Por exemplo, se o tamanho do pacote que acaba de chegar é de 10 bits, e o *connection finishing number* é 20, o seu *finishing number*

será igual a 30. Isto equivale a dizer que o pacote é completamente servido quando o servidor RR tiver com seu *round number* em 30.

O número de rodadas cresce a uma taxa inversamente proporcional ao número de conexões ativas. Se o *round number* for considerado apenas um número utilizado para o cálculo do *finishing number*, perdendo a sua correlação com o número de rodadas de um servidor RR bit-a-bit, a disciplina WFQ emula uma disciplina GPS, ao invés de um escalonamento RR bit-a-bit. A taxa de crescimento do *round number* é o inverso do número de conexões ativas. Por exemplo, caso haja quatro conexões ativas, o *round number* terá uma taxa de crescimento de 0,25.

Até agora os cálculos apresentados foram realizados considerando todas as conexões com um mesmo peso. Para conexões com pesos diferentes, o que caracteriza a disciplina WFQ, o cálculo do *finish number* dos pacotes é um pouco diferente, visto que deve ser considerado o peso atribuído à fila a qual o pacote pertence. A alteração no cálculo ocorre no valor do tamanho do pacote que será somado ao *round number*. Neste caso, o tamanho do pacote é dividido pelo peso. Por exemplo, suponha que um pacote de tamanho 1 chegue a uma fila de peso 50. O tamanho do pacote ponderado é de 1/50, que é igual a 0,02. Este é o valor adicionado ao *round number* corrente, obtendo com isso um *finish number* menor do que aquele que seria obtido na versão sem pesos.

Uma outra alteração necessária é no cálculo do valor do *round number* corrente. Na versão não ponderada, ele é obtido através da divisão de 1 pelo somatório das conexões ativas. Na versão ponderada, divide-se 1 pela soma dos pesos das conexões ativas. Por exemplo, se há quatro conexões ativas com os valores de pesos igual a 1, 4, 5 e 10 respectivamente, o valor do crescimento do *round number* corrente é igual a 0,05.

A disciplina WFQ também prescreve uma política de descarte. Se um pacote chega a um escalonador com a fila cheia, um ou mais pacotes devem ser descartados, em ordem decrescente do *finish number*, a fim de propiciar um espaço suficiente para o pacote que acabou de chegar.

Vale destacar três importantes características da disciplina. A primeira refere-se à proteção entre os fluxos, ou agregados de fluxos, em função da aproximação da disciplina WFQ com a disciplina GPS. Segundo, em certas circunstâncias na disciplina WFQ, um fluxo pode alcançar um pior caso de atraso de enfileiramento fim-a-fim,

independente do número de nós por onde ele passa e do comportamento dos outros fluxos. Por último, a disciplina WFQ força os usuários a implementarem um rigoroso controle de fluxo, visto que, se os seus fluxos forem mal comportados, a taxa de descarte será elevada.

2.3.4.5- Variantes da WFQ

Em função da dificuldade do cálculo do *round number* corrente da disciplina WFQ, foram desenvolvidas algumas variantes. Estas variantes modificam este cálculo, tornando bem mais simples a sua implementação, porém mantêm boa parte do seu desempenho.

A primeira delas é a *Self-Clocked fair Queuing* (SCFQ). Nela, quando um pacote chega a uma conexão inativa, ao invés de ser utilizado o *round number* corrente para o cálculo do *finishing number* do pacote, utiliza-se o *finish number* do pacote que está sendo servido. Apesar de ser bastante simples de implementar, em pequenos períodos de tempo esta disciplina torna-se injusta.

Uma outra variação da disciplina WFQ é a disciplina *Start-Time Fair Queuing* (STFQ), que supera algumas das limitações da SCFQ. Na disciplina STFQ, além do cálculo do *finish number*, há o cálculo do *start number*, que é o número relativo ao momento da chegada do pacote. O *start number* de um pacote que chega em uma conexão inativa é o *round number* corrente. No caso de o pacote chegar em uma conexão ativa, o valor do seu *start number* é igual ao valor do *finish number* do pacote da mesma conexão que o antecede na fila. O *finish number* de um pacote é a soma do *start number* do pacote ao tamanho do pacote dividido pelo peso da conexão. O *round number* corrente recebe o valor do *start number* do pacote que está em serviço. Se não há mais nenhum pacote a ser enviado, o *round number* recebe o valor do pacote que teve o maior *finish number* até aquele momento. Os pacotes são servidos na ordem do seu *start number*.

2.3.5- Políticas de Descarte de Pacotes

O módulo escalonador armazena os pacotes que estão esperando para serem servidos em um *pool* de *buffers*. Quando não há mais *buffers* disponíveis para os pacotes que estão chegando é necessária uma política de descarte de pacotes. Um pacote só deverá ser descartado em caso de absoluta necessidade. Enquanto a política de escalonamento é a responsável pela escolha do próximo pacote que será servido, a política de descarte é a responsável pela escolha do próximo pacote que deverá ser descartado, definindo a taxa de descarte (*loss ratio*) dos pacotes. Juntamente com a política de escalonamento, a política de descarte pode estabelecer diferentes classes de QoS a diferentes usuários, alocando diferentes atrasos de enfileiramento e taxas de descarte para suas requisições.

A política de escolha do próximo pacote a ser descartado deverá ser justa, visto que ela poderá privilegiar um fluxo, ou uma classe de fluxos, em detrimento de outro. No caso da disciplina de fila FCFS, quando não há *buffers* disponíveis, todos os pacotes que estiverem chegando serão descartados. Esta política é conhecida como *tail drop*. Caso um fluxo seja mal comportado, ele provavelmente irá ocupar um maior número de *buffers*, enquanto os fluxos bem comportados terão um maior descarte de pacotes. Portanto, a política de descarte deverá proteger os fluxos bem comportados daqueles mal comportados.

No modelo Diffserv esta proteção deverá ser entre os agregados (classes) de fluxos. Os fluxos pertencentes a uma mesma classe não terão proteção entre si.

As políticas de descarte podem ser classificadas de acordo com quatro características: nível de agregação, escolha de prioridades de descarte, descarte antecipado ou em sobrecarga e posição de descarte. A seguir estas características serão descritas.

2.3.5.1- Nível de Agregação

Assim como na política de escalonamento, a política de descarte pode estar baseada no tratamento dos fluxos de conexões individuais ou agrupados em classes. No primeiro caso, o algoritmo de descarte armazena um estado para cada conexão,

ocasionando um grande número de informações. No segundo, os estados armazenados são relativos às classes de fluxos, que demandam um menor número de estados armazenados. Por outro lado, o tratamento individual das conexões permite um maior isolamento entre os fluxos destas conexões, e conseqüentemente uma maior proteção de descarte, o que é conseguido apenas em nível de classe no segundo caso. Fluxos mal comportados em uma mesma classe podem causar danos aos fluxos bem comportados da mesma classe.

Se os pacotes são enfileirados por conexão ou por classe e compartilham um mesmo *pool* de *buffers*, o algoritmo de descarte estará de acordo com a alocação justa "min-max" de *buffers* quando descartar o pacote da fila mais longa. Isto se deve ao fato de que, havendo *buffers* disponíveis, os fluxos alocam a quantidade de *buffers* que precisam.

2.3.5.2- Prioridades de Descarte

Uma política de descarte pode estar baseada no descarte preferencial de alguns pacotes sobre outros. Por exemplo, alguns pacotes de menor relevância de um fluxo podem receber uma marcação. Quando a rede estiver com baixa carga, estes fluxos trafegarão normalmente pela rede, atingindo o seu destino. Caso haja uma sobrecarga na rede, ou em um determinado segmento da rede, estes pacotes serão preferencialmente descartados. Desta forma, os pacotes que não foram marcados possuem uma maior garantia de alcançarem o seu destino. A marcação dos pacotes poderá ser feita pela fonte de dados ou por elementos na entrada da rede, conhecido como policiadores.

Um segundo método utilizado nas políticas de descarte são aqueles baseados no descarte de células de um mesmo pacote. Em uma rede onde a unidade de transmissão são as células e onde um único pacote se divide em diversas células, o descarte de uma das células do pacote corresponde ao descarte do pacote inteiro. Portanto, há mecanismos que possibilitam descartar preferencialmente células pertencentes a um mesmo pacote.

Um outro método pode estar baseado no descarte de pacotes dos fluxos que possuem origem mais próxima do local de congestionamento. Este método considera

que os fluxos de origem mais distante já usaram uma grande quantidade de recursos da rede e, caso sofram descarte, haverá um maior desperdício de recursos.

2.3.5.3- Descarte Antecipado ou em Sobrecarga

As técnicas de descarte em sobrecarga, que descartam os pacotes só quando as filas estão completamente cheias, são as mais usuais e largamente utilizada na Internet. Elas possuem dois grandes inconvenientes. O primeiro é conhecido como *lock-out*, fenômeno que ocorre quando um fluxo ou um agregado de fluxos monopolizam o espaço de *buffers* disponíveis nas filas. O segundo deve-se ao fato de que as disciplinas de descarte em sobrecarga podem manter as filas cheias por um longo período de tempo, visto que a sinalização de congestionamento (descarte de pacotes) só ocorre quando as filas já estão completamente cheias. Em função dos fluxos TCP chegarem geralmente em rajadas nos roteadores, se a fila estiver cheia ocorrerá uma rajada de descarte, o que é extremamente prejudicial para o desempenho da transmissão.

A manutenção de filas pequenas nos roteadores não está relacionada apenas aos baixos *delays* pretendidos nas transmissões, mas principalmente a necessidade de ter espaço nas filas para possibilitar a absorção das rajadas dos fluxos, a fim de não descaracterizá-los. A solução deste problema é a utilização de disciplinas que descartem pacotes antes de as filas tornarem-se cheias. Com este objetivo, foram desenvolvidas políticas com descarte antecipado, onde o descarte de pacotes ocorre antes do preenchimento dos *buffers*. Estas políticas são as mais adequadas para as redes onde as fontes de dados são sensíveis a perda de pacotes, como em redes TCP. O descarte antecipado previne os congestionamentos.

Os dois métodos mais conhecidos de descarte antecipado para filas de agregados de fluxos são: *early random drop* e o *random early detection* (RED) [BCC98]. O *early random drop* faz o descarte dos pacotes que estão chegando com uma probabilidade fixa, assim que a fila atinge um tamanho que excede um determinado nível de descarte. A idéia por trás deste mecanismo é de que os fluxos mal comportados enviam mais pacotes e, sendo o descarte aleatório, estes fluxos terão um maior número de pacotes descartados.

Ao contrário das políticas de gerenciamento de filas tradicionais, a política de descarte RED faz parte da classe de políticas de gerenciamento de filas ativo. Nesta política, o descarte dos pacotes que estão chegando à fila se dá de forma probabilística. A probabilidade de descarte aumenta quando a estimativa da média do tamanho da fila aumenta. A resposta ao crescimento da fila não é instantânea, mas baseada em uma média do tamanho da fila em um determinado período de tempo. Isto previne que mudanças repentinas em um "passado recente" alterem de forma brusca a taxa de descarte de pacotes.

O algoritmo RED consiste em duas partes. A primeira parte faz uma estimativa do tamanho médio da fila e a segunda parte toma a decisão de descartar ou não um pacote que está chegando. Há dois parâmetros para o descarte: um limite mínimo e outro máximo. Se o tamanho da fila está abaixo do limite mínimo, nenhum pacote é descartado. Se o tamanho da fila é maior que o limite máximo, todos os pacotes são descartados. Caso o tamanho da fila esteja entre os limites mínimos e máximos, haverá um descarte probabilístico dos pacotes, variando linearmente entre zero e um valor máximo. A decisão de descarte de um pacote que está chegando pode estar no modo pacote, que ignora o tamanho do pacote, ou no modo *byte*, que leva em consideração o tamanho do pacote.

As políticas de descarte antecipado são conhecidas como "gerenciamento de filas ativo". Com estas políticas, o número de pacotes descartados nos roteadores é reduzido, pois a média do tamanho das filas torna-se pequena e, com isso, as rajadas de dados são absorvidas sem descarte de pacotes. Além disso, com filas menores o *delay* dos fluxos diminui. Com espaço nas filas, o fenômeno *lock-out* também é evitado, pois sempre haverá espaço para os pacotes dos diversos fluxos que estão chegando.

2.3.5.4- Posição de Descarte

O pacote que será descartado não é necessariamente o último de uma fila. Ele pode ser o primeiro da fila ou ocupar uma posição aleatória. Uma fila pode também ter todos os seus pacotes descartados. Cada um dessas estratégias terá vantagens e desvantagens.

O descarte de pacotes ao final da fila, conhecido como *tail drop*, é o mais comum e fácil de implementar, visto que o descarte se dá em um pacote que está chegando, não alterando os ponteiros de início e fim da fila. O descarte de pacotes que estão no início da fila possui uma implementação mais difícil, mas permite que a fonte perceba mais rápido o congestionamento. Este método de descarte é bastante adequado para o TCP que é sensível ao descarte, permitindo uma retransmissão mais rápida. O descarte de pacotes em posições aleatórias permite que fluxos mal comportados tenham uma maior quantidade de pacotes perdidos. Isto se deve ao fato de que os pacotes de fluxos mal comportados estão em maior número nos *buffers*, conseqüentemente, possuindo maior probabilidade de serem descartados.

O método de descarte da fila mais longa (toda a fila é descartada) penaliza os fluxos mal comportados, além de liberar uma grande quantidade de *buffers*. Este método pressupõe que as filas mais longas pertencem aos fluxos mal comportados.

2.4- Tráfego *Multicast*

As aplicações tradicionais de redes envolvem comunicações entre dois computadores. No entanto, algumas aplicações emergentes, tais como LAN TV, vídeo/áudio conferência, *broadcast* corporativo e computação colaborativa requerem comunicação simultânea entre grupos de computadores. Estas comunicações são classificadas como comunicações multiponto.

Há três maneiras para se estabelecer a comunicação multiponto. A primeira delas é a comunicação *unicast*. Nela, as aplicações enviam uma cópia de cada pacote para cada membro do grupo multiponto. Esta técnica é simples de implementar, mas tem problemas relativos a escalabilidade quando há grupos com um grande número de computadores. Além disso, há necessidade de grande largura de banda, pois a mesma informação é carregada múltiplas vezes, uma cópia para cada membro do grupo, mesmo quando alguns membros compartilham um mesmo enlace. A segunda forma de comunicação multiponto é a comunicação *broadcast*. Nela, as aplicações enviam uma cópia de cada pacote para um endereço de *broadcast*. Esta técnica é mais simples do que a que usa comunicação *unicast*. No entanto, se esta técnica é usada a rede deve ou limitar o *broadcast* aos limites da LAN, evitando *broadcast storms*, ou enviar o

broadcast para todas as redes. Quando há um pequeno grupo de computadores que necessitam receber os pacotes da aplicação multiponto, o envio de pacotes *broadcast* para todas as redes usa uma quantidade desnecessária de recursos de rede. A terceira forma de comunicação multiponto é a comunicação *multicast* [D89]. Nela, a aplicação envia apenas uma cópia de cada pacote para o endereço *multicast* e apenas os computadores que fazem parte do grupo multiponto recebem os pacotes. Os pacotes são enviados somente para as redes que possuem membros do grupo multiponto, havendo uma otimização do uso dos recursos de rede.

A seção 5.1 apresenta uma introdução ao *multicast* do protocolo IP. Na seção 5.2 são apresentados conceitos sobre roteamento *multicast*, bem como alguns exemplos de protocolos multicast. Nesta seção é apresentado o protocolo de roteamento *multicast* utilizado na arquitetura proposta. Na seção 5.3 são apresentadas algumas considerações sobre o tráfego *multicast* no Diffserv. Além disso, nesta seção são apresentadas as considerações encontradas na RFC 2475 que define o modelo Diffserv, bem como um resumo da proposta de Bless e Wehrle sobre tráfego *multicast* no Diffserv.

2.4.1- IP *Multicast*

O IP multicast é uma técnica de roteamento onde o tráfego de dados IP é enviado de uma fonte ou múltiplas fontes a múltiplos receptores. O IP multicast é baseado no conceito de grupos. Um grupo multicast é um grupo de computadores que possuem interesse em receber tráfego de dados de uma determinada aplicação. As aplicações enviam os pacotes para grupos *multicast*, identificados por um endereço *multicast*. Um grupo de computadores *multicast* não possui qualquer limite físico ou geográfico. Os computadores interessados em receber tráfego de um determinado grupo devem fazer uma requisição explícita para se juntar ao grupo, por meio do *Internet Group Management Protocol* (IGMP) [D89].

2.4.1.1- Endereço IP *Multicast*

O endereço IP *multicast* especifica um grupo de computadores que desejam receber o tráfego enviado para este grupo. O *Internet Assigned Numbers Authority*

(IANA) controla a atribuição de endereços *multicast*, destinando-lhes os endereços IP Classe D.

O IANA reservou os endereços entre 224.0.0.0 e 224.0.0.255 para serem usados por protocolos de rede em um segmento de rede local. Por exemplo, o protocolo OSPF utiliza os endereços 224.0.0.5 e 224.0.0.6 para trocar informações de estados entre os roteadores. Os pacotes com os endereços reservados não são retransmitidos pelos roteadores.

A faixa entre 224.0.1.0 até 238.255.255.255 contém os chamados endereços de escopo global [I01], que podem ser usados para troca de dados *multicast* entre organizações e através da Internet. Alguns deles foram reservados para algumas aplicações, como, por exemplo, o endereço 224.0.1.1 é reservado para o *Network Time Protocol* (NTP).

A faixa entre 239.0.0.0 até 239.255.255.255 abrange os endereços de escopo local ou administrativo [M98a], destinados a grupos locais ou de organizações. Os roteadores são configurados para evitar que pacotes com estes endereços saiam de um determinado *Autonomous System* (AS) ou um determinado domínio, permitindo que estes endereços sejam reutilizados.

A RFC 2770 [ML00] propõe que a faixa 233.0.0.0/8 seja reservada para endereços definidos estaticamente para organizações que já tenham um número de AS reservado. O número de AS do domínio é usado como o segundo e o terceiro octeto da faixa 233.0.0.0/8. Por exemplo, o AS 62010 é escrito em hexadecimal como F23A. Os dois octetos F2 e 3A possuem valor decimal 242 e 58. Então, o AS 62010 tem disponível os endereços de grupo *multicast* 233.242.58.0/24.

2.4.1.2- Mapeamento de Endereço IP *Multicast* no *Ethernet*

Um endereço *Ethernet* é composto por seis octetos (48 bits). O IANA definiu que os endereços Ethernet iniciados com 01:00:5E (em hexadecimal) são destinados para endereçamento *multicast*, restando 23 bits para identificar um endereço IP de um grupo *multicast*. Os 23 bits de menor ordem de um endereço IP *multicast* são utilizados para completar o endereço *Ethernet*. Como os 5 bits de maior ordem do endereço IP *multicast* são desprezados neste mapeamento, o endereço *Ethernet* resultante não

identifica univocamente um grupo *multicast*. Na verdade, cada endereço *Ethernet multicast* está associado a 32 endereços IP *multicast*.

2.4.1.3- *Internet Group Management Protocol* (IGMP)

O IGMP faz parte da pilha de protocolos TCP/IP, devendo ser implementado em todos os *hosts* que desejam receber tráfego *multicast* e usado para fazer dinamicamente o registro de um *host* em um grupo *multicast* em uma rede local. A fim de se tornarem membros de um determinado grupo *multicast*, os *hosts* enviam uma mensagem IGMP para o roteador *multicast* de sua sub-rede. Os roteadores executando o IGMP, além de receberem requisições IGMP, verificam quais grupos estão ativos ou inativos em uma determinada sub-rede. Isto é feito por meio do envio periódico de consultas (*queries*) as sub-redes.

O funcionamento básico do IGMP pode ser resumido segundo os procedimentos descritos abaixo:

- um *host* envia uma mensagem IGMP *report* ao roteador *multicast* de sua sub-rede indicando o desejo de se juntar (tornar-se membro) a um determinado grupo *multicast*. Esta mensagem indica ao roteador *multicast* que há um receptor para aquele grupo *multicast* naquela sub-rede. O *host* programa o seu dispositivo de rede para receber tráfego do grupo *multicast*.
- o roteador registra que a sub-rede possui um membro de um determinado grupo *multicast* e prepara-se para repassar todo o tráfego destinado ao grupo para a sub-rede.
- algum tempo depois, o roteador começa periodicamente a enviar mensagens IGMP de consulta para verificar se ainda há membros do grupo *multicast* na sub-rede. Se o *host* ainda deseja receber tráfego do grupo, ele irá enviar ao roteador uma mensagem IGMP informando que ele ainda é membro daquele grupo. Quando o *host* não deseja mais receber tráfego do grupo *multicast*, ele reprograma o seu dispositivo de rede para tal.

- após três envios consecutivos de mensagens sem resposta, o roteador considera o grupo inativo e pára de repassar tráfego para este grupo na sub-rede.

Usando *queries* e *reports*, um roteador *multicast* mantém uma tabela de suas interfaces que possuem pelo menos um *host* em um grupo *multicast*. Quando o roteador recebe um pacote *multicast* de um determinado grupo, ele envia o pacote somente para as suas interfaces que possuem membros para aquele grupo *multicast*. O protocolo IGMP não determina a maneira como os pacotes são enviados nos roteadores. Isto é feito de acordo com as regras do protocolo de roteamento *multicast* que está sendo executado no roteador.

Na versão 1 do IGMP [D89], os *hosts* enviam mensagens do tipo *Membership Report* para indicar o desejo de se tornarem membros de um determinado grupo *multicast* e responder que ainda são parte do grupo. Os roteadores enviam periodicamente mensagens IGMP do tipo *Membership Query* para verificar se há membros de um grupo *multicast* em uma sub-rede. Juntar-se a um grupo *multicast* é bastante rápido na versão 1 do IGMP, mas o deixar o grupo pode ser bastante vagaroso, pois somente após três mensagens consecutivas sem resposta é que o roteador detecta que não há mais membros de um determinado grupo *multicast* em uma sub-rede. Apesar de o *host* não estar mais recebendo o tráfego *multicast* (o dispositivo de rede é reprogramado para não mais receber tráfego daquele grupo *multicast*), o tráfego destinado ao grupo continua sendo enviado para a sub-rede.

A versão 2 do IGMP [F97] acrescenta à versão 1 a possibilidade de um de um *host* solicitar a sua saída de um determinado grupo *multicast*, através do envio da mensagem *Leave Group* ao roteador. Ao receber tal mensagem, o roteador envia à sub-rede uma mensagem para verificar se ainda há membros daquele grupo. Se não houver resposta, o roteador para de enviar tráfego daquele grupo a sub-rede. Este procedimento reduz em muito a latência de retirada de um membro de um grupo *multicast* comparado com a versão 1 do IGMP, agilizando o processo de parada de envio de tráfego desnecessário a sub-redes que não possuem mais membros de um determinado grupo *multicast*.

A versão 3 do IGMP [CDF01], que ainda está em desenvolvimento, permitirá que um *host* possa se juntar ou deixar um grupo, recebendo tráfego de uma fonte específica daquele grupo. Esta versão é adequada aos protocolos de roteamento *multicast* baseados em árvores de distribuição para cada fonte, com o PIM-SSM [HC00]. Tais protocolos de roteamento *multicast* são utilizados na arquitetura proposta neste trabalho e serão discutidos a seguir.

2.4.1.4- Árvores de Distribuição *Multicast*

Os roteadores *multicast* criam árvores de distribuição que formam os caminhos por onde tráfego IP *multicast* passa pela rede, levando o tráfego a todos os receptores. Os dois tipos básicos de árvores de distribuição são as árvores baseadas na fonte e as árvores compartilhadas.

A árvore de distribuição baseada na fonte é o tipo de árvore mais simples, onde a raiz da árvore é a fonte de dados do grupo *multicast* e os seus ramos formam uma árvore de espalhamento (*spanning tree*) pela rede até os receptores. Como esta árvore usa os menores caminhos para interligar a fonte e os receptores, ela também é chamada de árvore de menor caminho (S*hortest Path Tree* - SPT). Uma notação especial (S,G), muitas vezes referida como par (S,G), indica uma SPT onde S é o endereço IP da fonte e G é o endereço IP do grupo *multicast*. Esta notação implica em uma árvore de distribuição separada para cada fonte em um determinado grupo.

Diferentemente das árvores de distribuição que têm como raiz a fonte de dados, as árvores de distribuição compartilhadas usam um ponto comum como a raiz, localizado em algum lugar na rede. Esta raiz compartilhada é chamada de *Rendezvous Point* (RP). Em grupos *multicast* onde a árvore é compartilhada, as fontes devem enviar o seu tráfego de dados para a raiz, sendo este repassado para os ramos da árvore até alcançar todos os receptores. Uma vez que todas as fontes utilizam uma mesma árvore de distribuição, uma notação do tipo (*,G) é usada para representar a árvore de um determinado grupo. Neste caso, o "*" significa todas as fontes e G representa o grupo *multicast*.

Os membros de um grupo *multicast* podem se juntar ou deixar o grupo a qualquer momento; portanto a(s) árvore(s) do grupo devem ser dinamicamente

atualizadas. Quando todos os receptores ativos em um determinado ramo da árvore não desejarem mais receber o tráfego de dados do grupo, os roteadores podam este ramo, interrompendo o repasse do tráfego de dados do grupo pelo ramo. Se um receptor neste ramo torna-se ativo e requisita o tráfego *multicast*, o roteador dinamicamente modifica a árvore de distribuição e se inicia o repasse do tráfego.

As árvores de menor caminho têm a vantagem de criar um caminho ótimo entre a fonte de dados e os receptores, garantindo uma quantidade mínima de latência na rede para o envio do tráfego *multicast*. Esta otimização possui um preço. Os roteadores devem manter informações (estado) sobre o caminho para cada fonte. Em redes com milhares de fontes e milhares de grupos, isto pode comprometer o desempenho dos roteadores.

As árvores compartilhadas têm como vantagem o menor uso da quantidade de estados em cada roteador, exigindo menor quantidade de memória. A desvantagem é que, em certas circunstâncias, os caminhos entre as fontes de dados e os receptores podem não ser os menores, introduzindo maior latência na entrega dos pacotes. A localização do RP na rede é fundamental para o desempenho da entrega de dados em um grupo.

2.4.2- Roteamento *Multicast*

No roteamento *unicast*, o tráfego é roteado através da rede por um único caminho da fonte até o destino. Um roteador *unicast* não se preocupa com o endereço fonte, mas com o endereço destino e em como encaminhar o tráfego em direção a este destino. Ao receber um pacote, o roteador procura em sua tabela de roteamento *unicast* o caminho (interface) em direção ao destino por onde ele enviará o pacote recebido.

No roteamento *multicast*, uma fonte envia tráfego para um determinado grupo de *hosts*, representado por um endereço de grupo *multicast*. O roteador *multicast* deve determinar se a direção do fluxo é ascendente ou descendente. Se houver múltiplos caminhos de descida, o roteador replicará o pacote recebido da fonte, repassando cada cópia por estes caminhos de descida.

2.4.2.1- *Reverse Path Forwarding* (RPF)

O conceito de repasse de tráfego *multicast* baseado na fonte, ao invés dos receptores, é chamado de Encaminhamento pelo Caminho Reverso (*Reverse Path Forwarding*). O RPF é um conceito fundamental no roteamento *multicast*. Ele permite que os roteadores encaminhem corretamente o tráfego *multicast* em direção aos receptores na árvore de distribuição. O RPF usa a tabela de roteamento *unicast* para determinar os vizinhos tanto no sentido ascendente quanto no sentido descendente. Um roteador somente repassa um pacote *multicast*, se ele o recebeu pela interface na direção ascendente. Esta verificação RPF ajuda a garantir uma árvore de distribuição livre de *loops*.

Sempre que um pacote *multicast* chega a um roteador é feita a verificação RPF no pacote.Se a verificação RPF é bem-sucedida, o pacote é repassado. Caso contrário, ele é descartado. Para um tráfego descendo em uma árvore de distribuição baseada na fonte, a verificação RPF funciona da seguinte forma:

- assim que um pacote *multicast* chega ao roteador, o roteador procura o endereço fonte do pacote na sua tabela de roteamento *unicast*, verificando se o pacote chegou pela interface que é o caminho reverso de retorno a fonte;
- se o pacote chegou pela interface do caminho de retorno a fonte, a verificação RPF é bem-sucedida e o pacote é repassado adiante;
- se a verificação RPF foi mal-sucedida, o pacote é descartado.

2.4.2.2- DVMRP

O protocolo de roteamento *multicast* DVMRP [WDP88] é um protocolo intradomínio que usa a técnica RPF. Quando um roteador recebe um pacote *multicast*, ele o repassa para todas as interfaces, exceto aquela por onde o pacote foi recebido. Este procedimento é conhecido como inundação (*flood*). A inundação permite o tráfego de dados alcançar todas as redes e, possivelmente, alcançar uma mesma rede mais de uma vez. Se um roteador possui várias sub-redes conectadas que não possuem membros de um determinado grupo *multicast*, o roteador pode enviar uma mensagem de "poda"

(*prune*) de volta pela árvore de distribuição. Com isso, evita-se que o tráfego de um grupo *multicast* chegue a redes que não possuem membros do grupo.

O DVMRP irá re-inundar periodicamente a rede a fim de alcançar um novo *host* que deseja receber tráfego de um determinado grupo *multicast*. Há uma relação direta entre o tempo que um receptor leva para receber o tráfego de um grupo e a freqüência da inundação. O DVMRP implementa seu próprio protocolo de roteamento *unicast* a fim de determinar qual é a interface de retorno à fonte do tráfego de dados. Este protocolo de roteamento é muito parecido com o RIP e sua métrica é o numero de saltos (*hop counts*). Assim, o caminho por onde o tráfego *multicast* segue pode não ser o mesmo pelo qual o tráfego *unicast* segue.

O DVMRP tem relevantes problemas relativos a escalabilidade, em vista de constantes inundações. Esta limitação é exacerbada, pois as primeiras versões do DVMRP não implementavam a "poda" dos ramos da árvore de distribuição. Ele tem sido usado no MBONE (*backbone multicast* de pesquisa na Internet), interligando redes institucionais através de túneis entre roteadores que têm habilitado o DVMRP. O MBONE é usado amplamente pela comunidade de pesquisa para a transmissão de diversas conferências.

2.4.2.3- Extensão *Multicast* para o OSPF (MOSPF)

O protocolo de roteamento *multicast* MOSPF [M94] é um protocolo interdomínio. Ele foi definido como uma extensão do protocolo de roteamento *unicast* OSPF (*Open Shortest Path First*) [M98]. Sendo um protocolo de estado de enlace, cada roteador que compõe a rede divulga o estado de cada um dos seus enlaces diretamente conectados a todos os demais roteadores pertencentes a uma rede ou a uma área da rede, como no OSPF. O protocolo OSPF pode dividir uma rede em diversas áreas, sendo todas estas áreas interligadas através de uma área denominada área de *backbone*. Os roteadores de uma mesma área divulgam suas rotas entre eles. As áreas são interligadas por meio de roteadores chamados roteadores de borda de área. A Figura 2.5 apresenta uma rede com roteamento OSPF dividida em áreas.

A divulgação das rotas ocorre sempre que o estado de um dos enlaces é alterado. A partir do recebimento destas informações, cada roteador tem conhecimento de todos

os enlaces que formam a sua área da rede, tendo assim, uma visão exata da topologia da rede ou da área da rede onde se encontram. Com base nestas informações, cada roteador executa o algoritmo de menor caminho (*Diskjtra Algorithm*), calculando as rotas para todos os possíveis destinos a partir dele. O MOSPF inclui informações *multicast* nas informações de atualização do OSPF trocada entre os roteadores. Assim, os roteadores MOSPF têm conhecimento sobre os grupos *multicast* ativos em cada rede. Além disso, o MOSPF constrói a árvore de distribuição *multicast* para cada par fonte/grupo (S,G) e calcula a árvore para cada fonte ativa que está enviando tráfego para o grupo. O estado com informações sobre a árvore é posto em uma área de *cache* e as árvores são recalculadas sempre que houver uma mudança de estado de um enlace ou quando o tempo do *cache* se expira.

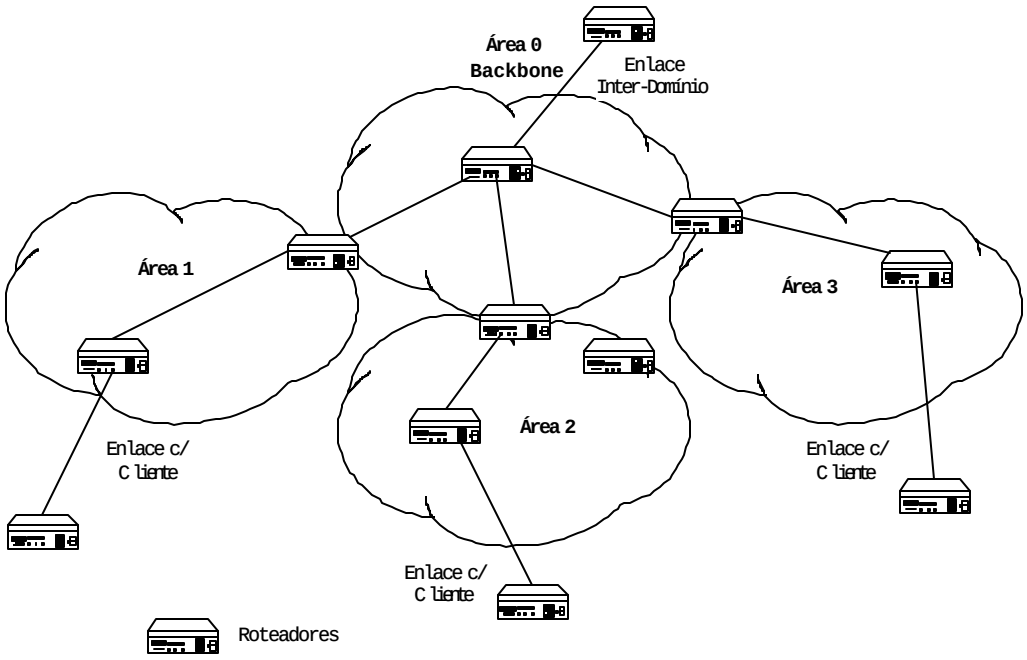


Figura 2.5- Rede OSPF Dividida em Áreas

O MOSPF é mais adequado para ambientes que têm poucos pares fontes/grupos ativos em um determinado momento, tendo um baixo desempenho em ambientes com grupos com muitas fontes de dados e com enlaces instáveis. Ele somente é usado em redes que utilizam o OSPF como protocolo de roteamento *unicast*.

2.4.2.4- *Protocol Independent Multicast* (PIM)

O protocolo de roteamento *multicast* PIM usa as informações de roteamento *unicast* para executar as funções de envio *multicast*, mas independe do tipo de protocolo de roteamento *unicast* usado. O PIM trabalha com qualquer um dos protocolos de roteamento *unicast* usuais, como o OSPF [M98] e BGP [LR91], e também com o roteamento estático. Ele usa a tabela de roteamento *unicast* para executar as funções de verificação RPF. O PIM suporta dois tipos diferentes de padrões de distribuição de tráfego multiponto: denso e esparso.

- **PIM Modo Denso (PIM-DM)**

O PIM-DM [DEF98] usa um modelo do tipo "*push*" para inundar o tráfego *multicast* para toda a rede. Este é um método de "força bruta" para a entrega de dados aos receptores, mas em certas circunstâncias ou aplicações este pode ser um eficiente mecanismo. Por exemplo, quando houver membros de um grupo em todas as sub-redes de uma rede.

O PIM-DM inicialmente inunda o tráfego *multicast* por toda a rede. Os roteadores que não tem nenhum vizinho na direção descendente do fluxo enviam mensagens de "poda" de retorno, na direção ascendente, para não receber tráfego desnecessário. Este processo se repete periodicamente.

O mecanismo de inundação e poda é a maneira pela qual os roteadores armazenam as suas informações de estado, ao receberem o tráfego de dados. Este tráfego contém a informação de fonte e grupo a fim de que os roteadores na direção da descida do fluxo possam criar suas tabelas de roteamento *multicast*. O PIM-DM cria a sua árvore de distribuição baseada na fonte (S,G).

O modo denso é mais adequado quando:

- as fontes e os receptores estão próximos uns dos outros;
- há poucas fontes e muitos receptores;
- o volume de tráfego *multicast* é intenso;
- o tráfego *multicast* é constante.

- **PIM Modo Esparso (PIM -SM)**

O PIM-SM [EFH98] usa um modelo to tipo "*pull*" para a entrega de tráfego *multicast*. Somente segmentos de rede que têm receptores ativos que requisitaram os dados receberão o tráfego de dados do grupo *multicast*. O PIM-SM assume que nenhum *host* deseja receber tráfego *multicast*, a menos que explicitamente solicite.

O PIM-SM usa uma árvore de distribuição compartilhada. Dependendo da configuração dos roteadores, o tráfego pode permanecer na árvore compartilhada ou ser enviado por uma árvore de distribuição baseada na fonte otimizada. O tráfego inicialmente é enviado pela árvore compartilhada e, então, os roteadores ao longo do caminho determinam se há um caminho melhor para a fonte. Se houver um melhor caminho, o roteador designado (o roteador mais próximo do receptor) envia uma mensagem de *"join"* em direção a fonte e então o tráfego passa a ser enviado pela nova rota.

Uma vez que o PIM-SM utiliza uma árvore de distribuição compartilhada, pelo menos inicialmente, existe o conceito de *Rendezvous Point* (RP). O RP é um roteador que deve ser administrativamente configurado na rede. As fontes de dados do grupo se registram no RP e então enviam dados pela árvore compartilhada para os receptores. Se a árvore compartilhada não é o melhor caminho entre a fonte e o receptor, os roteadores dinamicamente criarão uma árvore de distribuição baseada na fonte, deixando de enviar o tráfego pela árvore compartilhada.

O modo esparso é mais adequado quando:

- há poucos receptores em um grupo;
- as fontes de dados e os receptores são separados por enlaces WAN;
- o tipo de tráfego é intermitente.

O PIM-SM é otimizado para ambientes onde há muitos fluxos de dados multiponto (fontes de dados) e cada fluxo é direcionado para um pequeno número de sub-redes na rede. Para estes tipos de grupos a técnica de Encaminhamento pelo Caminho Reverso desperdiça muita largura de banda.

2.4.2.5- PIM-*Source Specific Multicast* (PIM-SSM)

O *Multicast* de Fonte Específica (*Source Specific Multicast*) é uma extensão do multicast IP onde o tráfego de dados é repassado aos receptores a partir de uma única fonte, para a qual os receptores explicitamente se juntaram. Para grupos *multicast* configurados para o SSM, somente árvores de distribuição *multicast* de fonte específica são criadas. O SSM é um modelo de entrega de pacotes que melhor suporta aplicações um-para-muitos, também conhecida como aplicações *broadcast*.

O PIM-SSM é um protocolo de roteamento que suporta a implementação do SSM e é derivado do PIM-SM. A atual infraestrutura *multicast* IP na Internet e em muitas intranets empresariais é baseada no protocolo PIM-SM e no protocolo MSDP [FRM00] (o protocolo MSDP permite que os RPs de diversos domínios se comuniquem, permitindo assim que receptores de um domínio receba tráfego *multicast* de outro domínio). Estes protocolos oferecem muita confiabilidade e eficiência. No entanto, eles estão limitados pela complexidade e limitações funcionais do modelo de serviços *multicast* padrão da Internet (ISM) [D89]. Por exemplo, com o ISM as redes devem manter o conhecimento sobre quais *hosts* estão ativamente enviando tráfego *multicast*. Com o SSM, esta informação é oferecida pelos receptores por meio do endereço da fonte passado ao roteador mais próximo aos receptores. Com isso, não é necessário nenhum protocolo para a descoberta das fontes, como o MSDP, bem como torna-se desnecessário o uso de mecanismos como a inundação (*flood*) ou *Rendezvous Point* (RP). Como o receptor conhece a fonte de dados da qual deseja receber o tráfego *multicast*, a rede pode enviar a solicitação de "*join*" diretamente a fonte. A comunicação entre o receptor e o roteador deve ser feita por meio do IGMP versão 3, que permite a identificação do grupo e da fonte *multicast* que o receptor deseja se associar.

O ISM consiste na entrega de datagramas IP de uma fonte para um grupo de receptores chamado de grupo *multicast*. O tráfego de dados para o grupo *multicast* consiste em datagramas com um endereço IP *unicast* arbitrário da fonte S e um endereço de grupo *multicast* G, como o endereço de destino. Os *hosts* receberão estes tráfegos quando tornarem-se membros do grupo.

No SSM, a entrega de datagramas IP é baseado em canais (S,G). O tráfego para um canal (S,G) consiste em datagramas com um endereço IP *unicast* da fonte S e o

endereço IP do grupo *multicast* como endereço de destino. Os *hosts* irão receber este tráfego tornando-se membros do canal (S,G). Tanto no SSM quanto no ISM, nenhuma sinalização é necessária para um *host* tornar-se uma fonte de dados. No entanto, no SSM os receptores devem se inscrever ou se desinscrever para um canal (S,G) a fim de, respectivamente, receber ou não receber o tráfego de uma determinada fonte em um grupo. Em outras palavras, no SSM os receptores podem receber tráfego somente do canal (S,G) que eles se inscreveram, enquanto que no ISM os receptores não necessitam saber do endereço das fontes de dados das quais recebem o tráfego *multicast*. No SSM, para se inscrever no canal é necessária a versão 3 do IGMP que, como visto, permite a identificação do grupo e da fonte da qual o receptor deseja receber o tráfego.

O SSM pode coexistir com os serviços ISM existentes, aplicando o modelo de entrega SSM a um subconjunto configurado de faixa de endereços de grupo *multicast* IP. O IANA reservou a faixa de endereço 232.0.0.0 até 232.255.255.255 para aplicações e protocolos SSM.

2.4.3- Tráfego *Multicast* no Diffserv

O tráfego *multicast* no modelo Diffserv ainda é tema de estudos no *Diffserv Work Group* do IETF. Este tipo de tráfego possui duas características que dificultam a sua implementação no Diffserv: a replicação de pacotes e o dinamismo dos grupos *multicast*. Estas duas características podem comprometer os recursos reservados para o tráfego *unicast*, pois quando um pacote *multicast* chega a um nó de ingresso de um domínio, ele pode se replicar e sair do domínio por vários nós. Este comportamento do tráfego *multicast* pode ocasionar sobrecarga em alguns enlaces do domínio e violações de alguns SLAs. As violações ocorrem quando um determinado SLA não prevê o tráfego *multicast* (não houve um acordo para este tipo de tráfego) ou quando o tráfego *multicast* ultrapassa o limite acordado no SLA.

A fim de evitar as sobrecargas nos enlaces e violações dos SLAs, antes de aceitar um novo receptor para um grupo *multicast* (processo de admissão), torna-se necessário fazer uma verificação dos recursos disponíveis ao longo do novo caminho por onde este fluxo vai passar (novo ramo da árvore de distribuição de dados do grupo *multicast* para alcançar o novo receptor). Este processo consiste em verificar se nenhum

SLA será violado e se há recursos suficientes nos roteadores ao longo do caminho. A verificação deve ocorrer tanto para os SLAs intradomínio (entre o provedor de serviços Diffserv e seus clientes) quanto para os SLAs interdomínios (entre os provedores de serviços Diffserv).

O processo de admissão de um novo receptor é complexo, pois os grupos *multicast* são dinâmicos. A qualquer momento um receptor pode solicitar a sua participação em um grupo *multicast*, passando a receber os fluxos de dados daquele grupo. Assim, torna-se difícil saber antecipadamente quais serão os caminhos por onde os fluxos *multicast* irão passar e a quantidade de recursos para um grupo *multicast*.

Uma outra dificuldade ocorre quando há uma divisão do fluxo *multicast* no interior do domínio. Como um fluxo *multicast* pode entrar por um nó de extremidade do domínio e sair por vários outros, em algum roteador do interior do domínio, haverá uma divisão do fluxo. Neste caso, o fluxo *multicast* se replicará e seguirá por pelo menos dois caminhos diferentes. É possível que um dos caminhos não tenha recursos suficientes para o fluxo *multicast* daquele grupo, causando sobrecarga dos enlaces e violações de SLAs.

A seguir serão apresentadas as considerações feitas na RFC 2475 e a proposta feita por Blesser e Wehrle para a implementação do tráfego *multicast* no modelo Diffserv.

2.4.3.1- Considerações sobre o Tráfego *Multicast* Apresentadas na RFC 2475

A partir das dificuldades encontradas para a implementação do tráfego *multicast* no modelo Diffserv, a RFC 2475 [BBC98] apresenta duas considerações sobre esta implementação a fim de evitar que os recursos reservados para o tráfego *unicast* sejam comprometidos pelo tráfego *multicast*.

A primeira consideração é a reserva de um conjunto de *codepoints* e PHBs para uso exclusivo do tráfego *unicast*, de tal maneira que haja uma isolamento do tráfego *multicast*. A segunda é estabelecer SLAs de parcerias para o tráfego *multicast* diferente dos SLAs de parceria para o tráfego *unicast*, objetivando evitar violações de SLAs. Um conjunto de *codepoints* deve ser destinado ao tráfego *multicast* ou mecanismos de

classificação e condicionamento de tráfego devem ser implementados nos nós de saída do domínio DS, evitando tais violações e protegendo o tráfego *unicast*.

2.4.3.2- Proposta de Bless e Wehrle

Uma importante contribuição para a implementação do tráfego *multicast* no Diffserv foi a proposta feita por Bless e Wehrle [BW00]. Eles propõem o uso das tabelas de roteamento *multicast* dos roteadores para auxiliar no processo de remarcação dos pacotes de dados *multicast*, possibilitando que os receptores de um grupo *multicast* recebam tráfego *multicast* com uma melhor qualidade de serviço. Como o processo de roteamento dos pacotes *multicast* IP sempre consulta a tabela de roteamento *multicast* do roteador, a fim de determinar por quais enlaces que um pacote IP *multicast* deve ser enviado, pouca sobrecarga de processamento será acrescida devido ao procedimento de remarcação dos pacotes. Apesar da recomendação da RFC 2475, esta proposta não distingue o conjunto de *codepoints* destinado ao tráfego *unicast* e *multicast*.

Segundo a proposta, um novo campo deve ser adicionado à tabela de roteamento *multicast*. Este campo deve conter o valor do *codepoint* para a remarcação do *"DS Field"* dos pacotes oriundos de uma determinada fonte *multicast*. Inicialmente, todas as entradas da tabela *multicast* devem ter este novo campo contendo o valor do *codepoint* equivalente ao serviço LBE (*Lower than Best-Effort*) [BW99a], que é um serviço inferior ao serviço *best-effort*. Desta maneira, todos os pacotes de dados de um grupo *multicast* terão inicialmente o seu *"DS Field"* remarcado para o serviço "LBE" ao passarem por qualquer roteador pertencente à árvore de distribuição de dados do grupo.

Quando um receptor desejar fazer parte de um grupo *multicast*, recebendo tráfego *multicast* de uma determinada fonte com um serviço de melhor qualidade, ele deverá enviar uma mensagem solicitando este serviço. Esta solicitação desencadeia um processo de remarcação das tabelas de roteamento *multicast* ao longo do ramo da árvore *multicast* à qual o receptor pertence. Quando um roteador recebe esta mensagem, ele deve solicitar a uma entidade de gerenciamento ("*Bandwidth Broker*") a admissão de um novo fluxo. Como os roteadores interiores de um domínio devem ser mais simples (com menos sobrecarga de processamento), esta tarefa deve ser feita pelos roteadores de borda. Após o processo de admissão, a entrada da tabela de roteamento *multicast*

referente ao novo ramo terá o valor de *codepoint* alterado para o valor de *codepoint* do serviço de melhor qualidade solicitado pelo receptor. Algumas novas mensagens devem ser acrescidas ao conjunto de mensagens do protocolo de roteamento *multicast.*

Algumas considerações devem ser feitas quanto à proposta de Bless e Wehrle. Eles mencionam a necessidade de um processo de admissão de um novo receptor, mas não detalham a sua implementação. Os procedimentos utilizados para a descoberta dos nós (roteadores interiores) onde ocorrem as divisões do tráfego *multicast* também não são claramente mencionados. Eles citam a necessidade do uso de um novo tipo de tráfego, o tráfego LBE, a fim de não comprometer o tráfego *best-effort*, o que parece ser desnecessário, pois nenhuma garantia é oferecida para o tráfego *best-effort* no ambiente Diffserv.

CAPÍTULO 3

Arquitetura para Tráfegos *Multicast* no Diffserv e sua Implentação

3.1- Introdução

Este capítulo apresenta a arquitetura que tem por objetivo possibilitar o tráfego *multicast* em redes que implementam os serviços diferenciados e os detalhes relevantes da sua implementação. A arquitetura é composta por um sistema de reserva de recursos de rede e de roteadores que implementam serviços diferenciados. O sistema de reservas é baseado em Servidores de Reserva de Recursos (SRR), similares aos "*Bandwidth Brokers*" [NJZ99], que se comunicam a fim de verificar a disponibilidade de recursos (largura de banda) ao longo da árvore de distribuição *multicast*. Os SRRs são os responsáveis pela configuração dos roteadores dos domínios Diffserv por onde se estende a árvore de distribuição *multicast*. Os receptores que desejarem receber tráfego *multicast* com QoS devem solicitar uma reserva de recursos a um SRR. As reservas podem ser para sessões *multicast* futuras (reservas antecipadas) ou para sessões *multicast* em andamento (reservas imediatas). Em [BLB98] é descrita uma arquitetura para reservas antecipadas para o modelo Intserv/RSVP.

Os roteadores implementam o serviço "*multicast Premium*" utilizando o PHB *Expedited Forwarding* [JNP99]. O serviço *premium* [NJZ99] é um serviço fim-a-fim, de baixo atraso e baixo *jitter*, onde a alocação de recursos é feita pela da taxa de pico. Apesar das recomendações propostas na RFC 2475 [BBC98], neste trabalho são usados o mesmo *codepoint* e SLA (*Service Level Agreement*) para os fluxos *unicast* e *multicast*.

A arquitetura proposta utiliza uma simplificação do mecanismo de remarcação dos pacotes proposta por Bless e Wehrle [BW00]. Ao invés de utilizar um *byte* com o valor do *codepoint*, apenas um bit é acrescido à tabela *multicast* dos roteadores, indicando a necessidade de remarcar ou não o "*DS Field*" dos pacotes que estão sendo roteados. O sistema de reserva de recursos desta arquitetura também pode ser considerado um complemento à proposta de Bless e Wehrle, pois na proposta deles é citado, mas não descrito, o uso de um sistema de gerenciamento de recursos para a admissão de um novo receptor. Na arquitetura proposta neste capítulo também é

possível descobrir a exata localização dos nós interiores onde ocorre a divisão da árvore *multicast*, onde um de seus ramos não tem recursos suficientes para oferecer um serviço de melhor qualidade. Este processo não é bem detalhado na proposta de Bless e Wehrle. A arquitetura não usa o tráfego LBE proposto por Bless e Wehrle, mantendo o serviço *best-effort* como o serviço sem nenhuma garantia, assim como é originalmente feito pelo protocolo IP.

A seção 2 deste capítulo apresenta os detalhes da arquitetura proposta e a seção 3 apresentas os aspectos relevantes da implementação dos elementos que constituem tanto o sistema de reserva de recursos quanto os que constituem o ambiente de serviços diferenciados. Como o protocolo *multicast* PIM-SSM, proposto para ser usado na arquitetura, e o protocolo IGMPv3 ainda estão em fase de proposição do IETF, alguns módulos dos elementos da arquitetura não foram implementados.

3.2- Arquitetura para Tráfegos *Multicast* no Diffserv

Esta seção apresenta os detalhes da arquitetura proposta, mostrando a sua descrição, os protocolos *multicast* usados, os procedimentos de reserva, a estrutura funcional dos seus componentes e a interoperabilidade entre eles.

3.2.1- Descrição da Arquitetura

A arquitetura proposta é composta por um sistema de reservas de recursos e roteadores que implementam os serviços diferenciados. O sistema de reserva de recursos é composto por um conjunto de SRRs, responsáveis pela verificação e a reserva dos recursos nos enlaces que formam a árvore de distribuição de um grupo *multicast*. A arquitetura considera que um Domínio DS (Domínio de Serviços Diferenciados) é formado por uma rede provedora de serviços diferenciados e diversas redes clientes. Geralmente, a rede provedora é um Provedor de Serviços Internet (ISP), enquanto as redes clientes são redes empresariais ou redes dos provedores de acesso à Internet. Cada uma destas redes possui pelo menos um SRR. Os SRRs conhecem a topologia da rede à qual pertencem e executam o mesmo algoritmo de roteamento *multicast* usado nelas (protocolo *multicast* intradomínio), o que possibilita a

comunicação com os demais SRRs de seu Domínio, bem como a descoberta dos caminhos por onde o tráfego *multicast* passa naquele Domínio. A Figura 3.1 mostra a comunicação entre os SRRs em um Domínio DS.

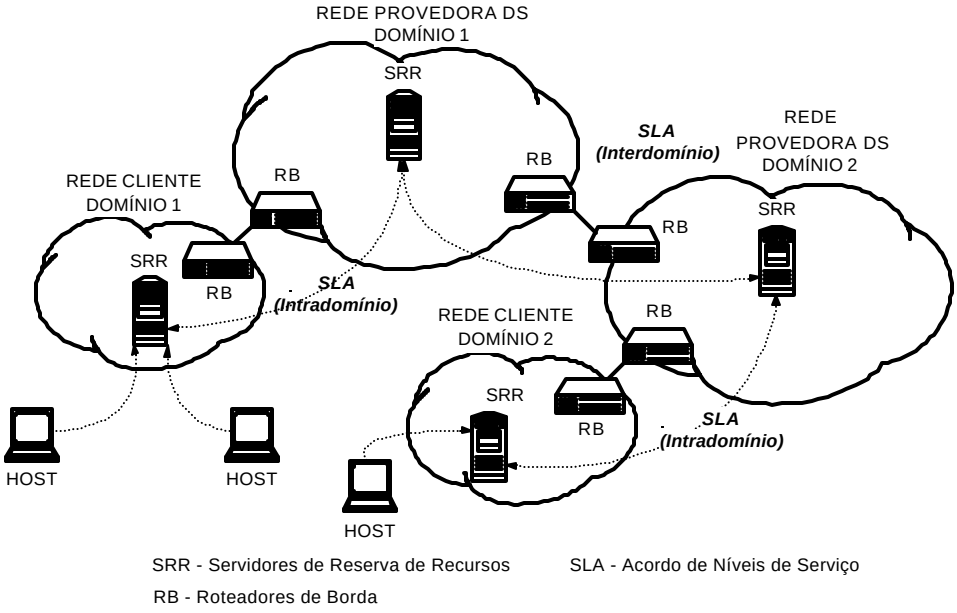


Figura 3.1 – *Comunicação* entre SRRs em um Domínio DS

Quando a árvore de distribuição *multicast* de um grupo se espalha por diversos domínios (uma região DS), os SRRs destes domínios devem se comunicar a fim de fazer as verificações dos enlaces e SLAs interdomínios. Para isso, os SRRs também devem participar do roteamento *multicast* interdomínio, executando o mesmo protocolo de roteamento *multicast* utilizado na Região DS (Região de Serviços Diferenciados). Esta participação permitirá que os SRRs de um domínio se comunique com o SRRs do domínio adjacente, por onde a árvore de distribuição *multicast* se estende.

Os SRRs são responsáveis também pela configuração dos roteadores pertencentes aos domínios DS com membros do grupo *multicast.* Existem três tipos de roteadores: roteadores folha, roteadores de borda e roteadores internos. Os roteadores folha são os roteadores mais próximos dos *hosts* que estão enviando fluxo *multicast* (fontes de dados do grupo), geralmente estão na mesma sub-rede. Além de darem um tratamento diferenciado aos fluxos que entram no Domínio DS, tais roteadores fazem o condicionamento deste tráfego. Os roteadores de borda e internos apenas desmarcam os

fluxos *multicast* que não obtiveram reservas, mas sem manter um estado descritor de reserva para cada fluxo, como nos roteadores folha. Em algumas situações os roteadores de borda podem dar um tratamento aos fluxos individuais.

Como visto, a reserva de recursos para o tráfego de um grupo *multicast* é uma tarefa conjunta dos vários SRRs da Região DS, que possuem a incumbência de verificar a disponibilidade de recursos nos Domínios DS por onde se estende a árvore de distribuição *multicast* do grupo. Para isso, os SRRs devem fazer a verificação de cada SLA e dos caminhos por onde o tráfego *multicast* passa, a fim de configurar todos roteadores que compõe os ramos da árvore de distribuição *multicast* do grupo.

Na arquitetura proposta, cada grupo *multicast* pode ter apenas um transmissor de fluxo *multicast* com QoS (somente o tráfego deste transmissor é marcado com o *codepoint* de tráfego *multicast*), apesar de um grupo IP *multicast* best-effort poder ser composta por diversos receptores e transmissores de fluxos *multicast.* Os fluxos de dados dos demais transmissores, caso existam, recebem um tratamento *best-effort* (os pacotes dos fluxos de dados destes transmissores têm valor de *codepoint* vazio ou igual a zero). Esta abordagem, apesar de parecer inicialmente uma grande limitação, torna-se bastante adequada a muitas aplicações em ensino à distância onde a necessidade de qualidade de transmissão é maior na direção do instrutor para os alunos do que no sentido inverso. Além disso, muitas aplicações necessitam apenas de um transmissor, como as aplicações denominadas *broadcast*. Tendo em vista tais aplicações, o IETF esta propondo novas especificações para grupos *multicast* com fonte específica (SSM), como pode ser visto no capítulo 2 deste trabalho. Muitos pesquisadores vêm apontando o SSM como uma forte tendência, devido a sua simplicidade com relação às demais abordagens de implementação *multicast*. Portanto, a nossa proposta estará baseada em árvores de distribuição SSM. Isto não impede o uso das demais abordagens, sendo a nossa arquitetura facilmente adaptável as mesmas. Além disso, na abordagem SSM cada *host* deve fazer explicitamente uma solicitação para se juntar a uma fonte de um grupo *multicast*, o que permite uma mais fácil adequação a nossa proposta.

Cada grupo *multicast* deve possuir um perfil que o descreva. Este perfil é composto pelas seguintes informações: endereço *multicast* do grupo, endereço do *host* transmissor (fonte), taxa de transmissão (Kbps), tipo de serviço, horário inicial e horário final. O endereço *multicast* identifica o endereço IP do grupo *multicast*. O endereço do

host identifica o endereço IP *unicast* da fonte daquele grupo (os fluxos desta fonte recebem o tratamento diferenciado). O tipo de serviço identifica o tipo de tratamento que o tráfego *multicast* deve receber em seu percurso (em nossa arquitetura é o serviço "*multicast premium*"). Os horários inicial e final identificam a data e a hora do início e o fim da sessão *multicast* (utilizado para reservas antecipadas). Este perfil fica armazenado nos SRRs (Servidor de Reserva de Recursos) do Domínio DS provedor onde se encontra a fonte de dados do grupo.

Cada participante de um grupo *multicast* que desejar receber tráfego *multicast* com QoS, deve fazer uma explicita solicitação de reserva de recursos de acordo com o perfil do grupo *multicast*. Esta reserva garante que o tráfego *multicast* que chega aos membros do grupo, a partir do transmissor com QoS, recebe um tratamento segundo o tipo de serviço estabelecido no perfil do grupo *multicast* ao longo de todo o caminho percorrido na rede. As reservas sempre são feitas na direção da fonte para o receptor. Em caso de protocolos *multicast* que criam uma árvore de distribuição para cada transmissor, a reserva é feita para os ramos da árvore de distribuição da fonte com QoS que tenham receptores com reserva. No caso de protocolos que usam uma árvore de distribuição compartilhada, a reserva é feita na direção da fonte com QoS para os receptores.

Os participantes de um grupo *multicast* podem estar espalhados por diversas redes e, até mesmo, por diversos domínios de serviços diferenciados. Logo, a árvore de distribuição *multicast* pode ter ramos no domínio DS da fonte (intradomínio) e ramos espalhados por diversos outros domínios (interdomínios). Os SRRs das diversas redes, onde há pelo menos um membro do grupo *multicast*, comunicam-se a fim de estabelecer as reservas intradomínios e interdomínios.

Caso haja falha na reserva, os pacotes dos fluxos *multicast*, passando pelo ramo da árvore *multicast* do receptor que não obteve a reserva, devem ser remarcados para receberem um tratamento *best-effort*. Por motivos de segurança e garantia do correto uso dos recursos, todos os pacotes de fluxos *multicast* devem ter seu *"DS Field"* (campo do cabeçalho IP que contém o *codepoint*) desmarcado ao passar por um roteador nas bordas e no interior de um domínio. Se o ramo da árvore de distribuição tem um receptor com uma reserva para um determinado grupo *multicast*, todos os roteadores pertencentes àquele ramo devem receber uma configuração especial. Esta

configuração vai impedir que os pacotes enviados pelo transmissor com QoS do grupo *multicast* sejam desmarcados, garantindo assim que estes pacotes recebam um tratamento diferenciado, como mostra a Figura 3.2.

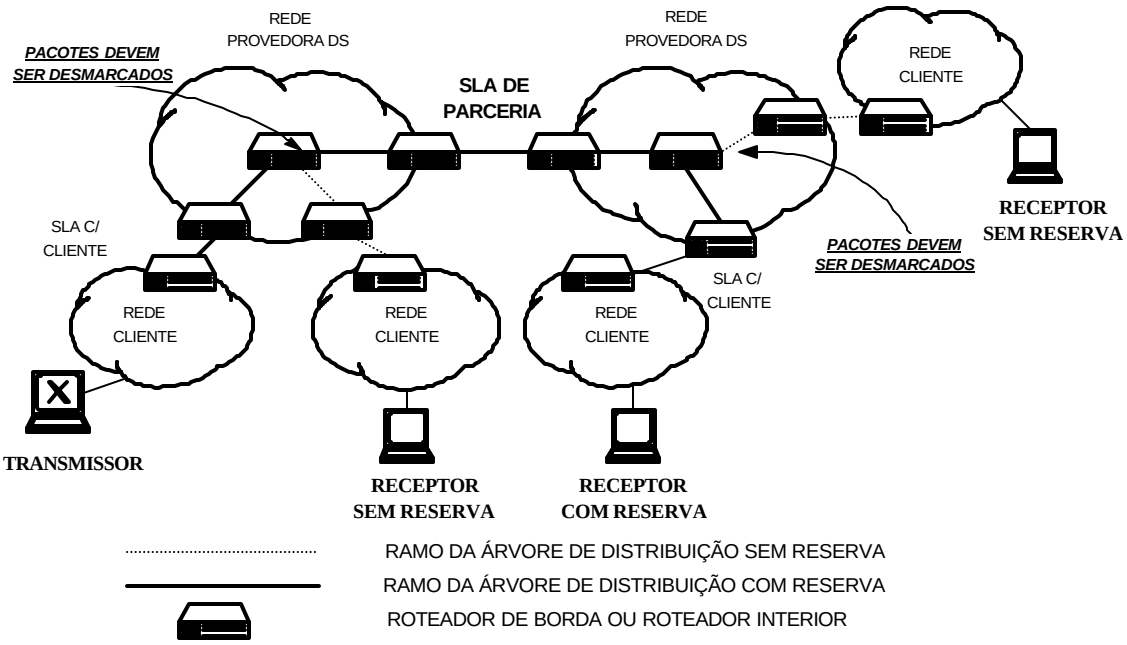


Figura 3.2 – Árvore de uma Sessão *Multicast* com alguns Ramos sem QoS

A configuração especial citada consiste em criar um indicador especial nas entradas da tabela de roteamento *multicast* dos roteadores. As entradas de uma tabela *multicast* consistem em pares fonte/grupo (S,G) ou só de endereços do grupo *multicast* (*,G) para cada interface de rede do roteador. O tipo de entrada na tabela depende do tipo de protocolo de roteamento utilizado (árvore baseada na fonte ou árvore compartilhada). Estas entradas indicam para quais interfaces de rede um determinado pacote *multicast* deve ser enviado. Inicialmente, todas as entradas da tabela possuem um indicador especial "desligada" (valor zero), indicando que o tráfego que é repassado para um determinado fonte/grupo ou grupo por aquela interface deve ser desmarcado. Quando o indicador de uma entrada estiver "ligada", há uma indicação de que os pacotes enviados a partir de uma determinada fonte/grupo ou de um grupo *multicast* através daquela interface não devem ser remarcados recebendo assim um tratamento diferenciado segundo o seu *codepoint*.

Esta abordagem pretende minimizar o processamento nos roteadores nas bordas e no interior dos domínios. Dependendo de sua localização e taxa de transmissão, os roteadores de borda podem usar a mesma abordagem dos roteadores folha, como os roteadores de borda de redes clientes. Por outro lado, se os receptores cujas requisições de reserva não tenham sido completamente atendidas continuassem capazes de receber fluxos de dados do grupo em um nível de qualidade inferior, diferente do tratamento *best-effort*, os roteadores interiores aos Domínios teriam uma maior complexidade. Isto, pois, nos domínios onde o tráfego *multicast* se divide, há a necessidade de remarcação dos pacotes para uma classe de serviço mais adequada. Conseqüentemente, pode ser preciso um policiamento destes fluxos garantindo a sua adequação à nova classe de serviço e, para tal, torna-se necessário armazenar informações de estado para cada um dos fluxos nos nós interiores, como pode ser visto em Berson [BV98].

As reservas podem ser antecipadas (para sessões *multicast* futuras) ou imediatas (para sessões *multicast* em andamento). Os dois tipos de reserva fazem o mesmo processo de verificação dos recursos. A diferença está no momento em que os roteadores são configurados. Nas reservas antecipadas, as informações sobre as reservas bem sucedidas são armazenadas nos SRRs e, logo que a sessão de transmissão *multicast* inicie, os SRRs configuram os roteadores. Já nas reservas imediatas, o processo de configuração dos roteadores é feito logo após a verificação da disponibilidade dos recursos. As reservas antecipadas são muito importantes para projetos de educação à distância, pois possibilita o agendamento de teleconferências entre instrutores e alunos.

3.2.2- Protocolos *Multicast*

Inicialmente, a arquitetura proposta utilizava dois tipos de protocolos de roteamento *multicast*: um intradomínio e outro interdomínio. O protocolo de roteamento *multicast* intradomínio com a responsabilidade pela formação dos caminhos por onde os pacotes de dados *multicast* serão conduzidos no interior de um Domínio DS e o protocolo de roteamento *multicast* interdomínio com a responsabilidade pela formação dos caminhos por onde os pacotes de dados *multicast* serão conduzidos entre os domínios de uma Região DS. Os protocolos de roteamento *multicast* utilizados eram o DVMRP, como protocolo intradomínio, e o protocolo MASC/BGMP, como protocolo

interdomínio [APO00]. A escolha do DVMRP se deu em função do seu amplo uso na Internet, principalmente no MBONE (*backbone multicast* de pesquisa na Internet). Algum tempo depois, o protocolo proposto para a arquitetura foi o MOSPF, substituindo o protocolo DVMRP.

A abordagem de usar dois tipos de protocolos multicast distintos parecia ser a mais adequada à Internet, visto que não obrigava os diversos Domínios DS interligados em uma Região DS a usarem um mesmo tipo de protocolo *multicast* intradomínio. Cada domínio podia fazer uso daquele protocolo *multicast* mais adequado à sua topologia. Em contrapartida, o uso destes protocolos criava algumas complexidades na troca de informações de controle entre os elementos da arquitetura, que foram simplificadas com o uso da abordagem de protocolos *multicast* SSM. Com ele, os receptores tornam-se membros de um canal (S,G), definindo a fonte e o grupo ao qual eles desejam se juntar a fim de receber o tráfego *multicast*. Isto facilitou muito o processo de reserva de recursos, elemento fundamental para o funcionamento dos SRRs. O protocolo *multicast* proposto para uso é o PIM-SSM, cuja principal vantagem é funcionar com qualquer protocolo de roteamento *unicast*.

A arquitetura também pressupõe o uso dentro dos domínios de roteamento *unicast* dinâmico de estado de enlace. Inicialmente, utilizávamos o RIP, um protocolo de roteamento *unicast* de vetor de distância (*distance-vector*). O RIP criava uma série de dificuldades na descoberta dos enlaces por onde os ramos das árvores de distribuição *multicast* se estendiam, que foram sanadas com o uso do roteamento de estado de enlace. Os roteadores usando roteamento *unicast* de estado de enlace divulgam o estado de cada um dos seus enlaces diretamente conectados a todos os demais roteadores pertencentes à rede. Esta divulgação ocorre sempre que o estado de um dos enlaces é alterado. A partir do recebimento destas informações, cada roteador tem conhecimento de todos os enlaces que formam a rede, tendo assim, uma visão exata da topologia da rede. Com base nestas informações, cada roteador executa o algoritmo de menor caminho (*Diskjtra Algorithm*), calculando as rotas para todos os possíveis destinos a partir dele. Na arquitetura proposta, o protocolo de roteamento *unicast* escolhido foi o OSPF. Um resumo do seu funcionamento foi descrito na seção 2.5.2.3 sobre o protocolo *multicast* MOSPF no capítulo 2 deste trabalho. Basicamente, o OSPF divide a rede de um Domínio em diversas áreas, sendo todas estas áreas interligadas através de uma área

denominada área de *backbone*. Os roteadores de uma mesma área divulgam suas rotas entre eles. As áreas são interligadas por meio de roteadores chamados roteadores de borda de área. Quando bem projetado, um Domínio usando OSPF se interliga a outro Domínio através de seus roteadores de borda de Domínio, que estão na área de *backbone* do Domínio. Cada área de rede do OSPF deve ter ao menos um SRR. Os SRRs na área de backbone serão denominados SRRs de borda. Estes SRRs, assim como os roteadores da área de *backbone*, possuem uma visão de toda a rede do Domínio. Além disso, cada rede cliente tem o seu próprio SRR. A Figura 3.3 mostra a disposição dos SRRs em um Domínio DS.

Os procedimentos de divulgação e atualização das informações dos enlaces do protocolo OSPF permitem aos SRRs receberem as informações sobre os enlaces que compõe a área da rede a qual pertencem, conhecendo com exatidão a sua topologia. O conhecimento exato da topologia da área possibilita aos SRRs calcularem os caminhos por onde o tráfego dos grupos *multicast* trafegam por meio da execução do mesmo algoritmo utilizado pelo OSPF.

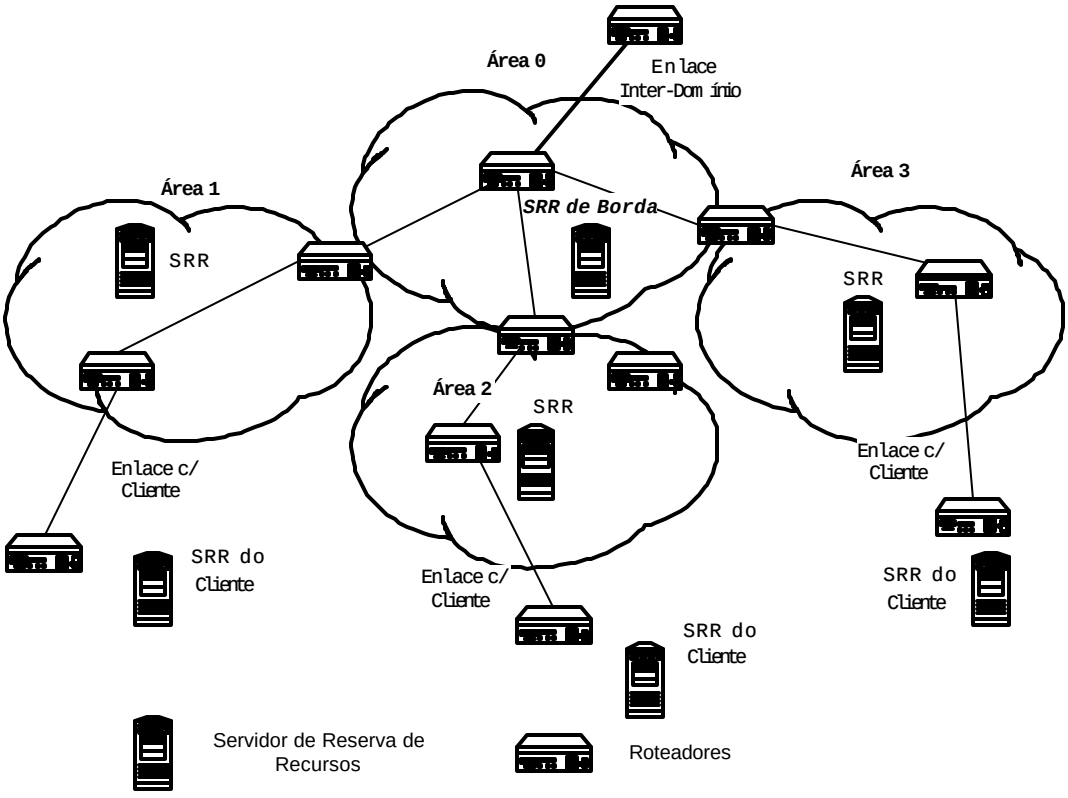


Figura 3.3- Rede OSPF Dividida em Áreas e os SRR de cada Área

Os SRRs devem executar os mesmos procedimentos para a formação da árvore de distribuição *multicast* usados pelo protocolo PIM-SSM. Dessa forma, os SRRs podem fazer uma verificação de sobrecarga do tráfego entre as áreas dentro do domínio, bem como fazer a verificação de violações dos SLAs com suas redes clientes (SLAs intradomínios). Além disso, é possível localizar os pontos de divisão do fluxo *multicast* no interior do domínio.

A falta de recursos em um dos ramos da árvore *multicast* torna necessária uma remarcação dos pacotes para um tratamento *best-effort*. Quando um ramo da árvore de distribuição *multicast* se estende para outros Domínios, os SRRs do interior do Domínio devem entrar em contato com o SRR de borda, para que este possa entrar em contato com os SRRs de borda dos Domínios adjacentes. Os SRRs devem possuir também as informações necessárias para o roteamento interdomínio, da mesma maneira que possuem as informações intradomínio, bem como executar os mesmos procedimentos para descoberta de rotas usados pelos roteadores de borda. Em resumo, os SRRs de borda devem participar do roteamento interdomínio, assim como participam do roteamento intradomínio. O protocolo interdomínio proposto é o BGP, devido ao seu amplo uso na Internet. A partir de um endereço IP de destino, um roteador executando o BGP sabe para qual Domínio (enlace) ele deve encaminhar o datagrama IP. Caso haja mais de um roteador BGP dentro de um Domínio, e cada um deles com enlaces para diferentes Domínios, eles sabem qual deles possui o melhor caminho para um determinado destino.

3.2.3- Procedimentos de Reserva

O processo de reserva de recursos é iniciado a partir da criação de um perfil com as características da transmissão de um canal (S,G) *multicast*. Este perfil é armazenado no(s) SRR(s) de borda do Domínio DS da fonte *multicast.*

Para que os receptores associados a um canal possam receber tráfego *multicast* com QoS, eles devem fazer uma requisição de reserva, com base no perfil daquele canal, ao SRR de sua rede. O SRR da rede do receptor, ao receber a requisição, faz uma autenticação do receptor, verificando se ele tem permissão para fazer uma reserva. Se houver, o SRR faz o controle de admissão, verificando se o receptor tem disponível a

quantidade de recursos solicitados. Em caso afirmativo, o SRR envia uma mensagem de requisição a um SRR do provedor (ISP) do Domínio.

O SRR do provedor, ao receber a mensagem de solicitação de reserva, faz a autenticação e o controle de admissão na rede provedora do SRR solicitante (o SRR da rede do receptor). O controle de admissão na rede provedora verifica se não há violação do SLA intradomínio, entre a rede cliente e a rede provedora. A seguir, o SRR da rede provedora deverá entrar em contato com os demais SRRs da rede provedora (SRR de borda pertencente à área do *backbone* OSPF ou a um SRR de uma rede cliente), a fim de verificar se há recursos suficientes nos enlaces que compõem o ramo que interliga o novo receptor ao ramo mais próximo da árvore de distribuição da fonte/grupo *multicast*. Como visto, o SRR conhece a topologia da rede através das informações divulgadas pelo OSPF e pela execução do algoritmo de menor caminho (*Diskjtra Algorithm*). Tais caminhos são verificados para saber se há alguma violação de SLAs intradomínio (entre o provedor de serviço e seus clientes) ou se há sobrecarga nos enlaces que interligam as áreas OSPF do domínio. Se o ramo do receptor se estende para outro Domínio (isto ocorre quando o ramo mais próximo da árvore *multicast* da fonte/grupo está em outro domínio), o SRR de borda do Domínio deve repassar a solicitação de reserva para o SRR do Domínio adjacente em direção ao ramo mais próximo da árvore de distribuição *multicast* da fonte/grupo. Para isso, o SRR de borda executa o protocolo de roteamento *unicast* interdomínio, descobrindo o Domínio adjacente (enlace). Então, o SRR de borda consulta a tabela com os endereços IP dos SRRs de borda dos domínios adjacentes, a fim de descobrir o endereço IP do SRR para o qual deve ser repassada a solicitação de reserva.

O SRR de borda do domínio adjacente, ao receber a mensagem de requisição de reserva faz a verificação do SLA (*Service Level Agreement*) interdomínio relativo ao domínio anterior, de forma que este não seja violado. Ele também irá verificar se não há sobrecarga nos caminhos que compõem o ramo do receptor dentro do Domínio, em conjunto com os demais SRRs do seu Domínio. Os enlaces que interligam as áreas OSPF do domínio e os SLAs intradomínios com as redes clientes são verificados por intermédio dos procedimentos descritos anteriormente. Isto ocorre até que um SRR que tenha um ramo da árvore de distribuição *multicast* da fonte/grupo na sua área OSPF seja atingido. Quando isto ocorrer, uma mensagem de confirmação retorna a todos os SRRs

que participaram da requisição da reserva até alcançar o SRR da rede do receptor requisitante. Este mesmo procedimento ocorre caso seja negada a requisição de reserva.

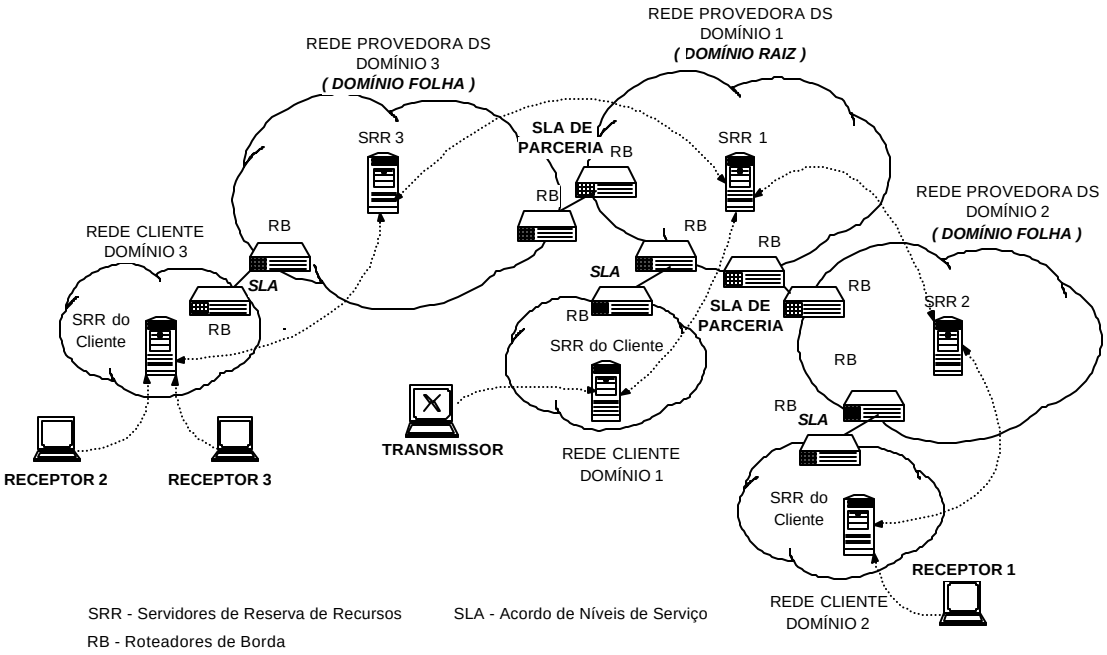


Figura 3.4 - Árvore *Multicast* Intradomínio e Interdomínio

A Figura 3.4 mostra um esboço de uma árvore de interligação de SRRs de um canal (S,G) *multicast* cujas reservas foram bem sucedidas. Nela, estamos chamando de Domínio Raiz o domínio da fonte do grupo *multicast* que está transmitindo fluxos que devem ter um tratamento segundo o serviço "*multicast premium*" para aqueles receptores que obtiveram sucesso na reserva de recursos. Todos os demais Domínios DS serão considerados Domínios Folhas. Esta é uma nomenclatura utilizada apenas como uma referência neste trabalho, não tendo qualquer relação com as especificações do PIM-SSM.

Caso haja falha na reserva, os pacotes dos fluxos *multicast*, passando pelo ramo da árvore *multicast* do receptor que não obteve a reserva, devem ser remarcados para receberem um tratamento *best-effort*, como foi apresentado na descrição da arquitetura.

3.2.4- Estrutura Funcional do SRR

Os SRRs são os responsáveis pelas reservas de recursos intradomínios e interdomínios. Cada rede deve ter pelo menos um SRR de borda e um SRR para cada

área do protocolo de roteamento OSPF. A sua estrutura funcional é composta pelos seguintes módulos: módulo autenticador, módulo de controle de admissão, módulo de comunicação, módulo de configuração e o módulo de descoberta de rotas. A Figura 3.5 mostra a estrutura funcional de um SRR.

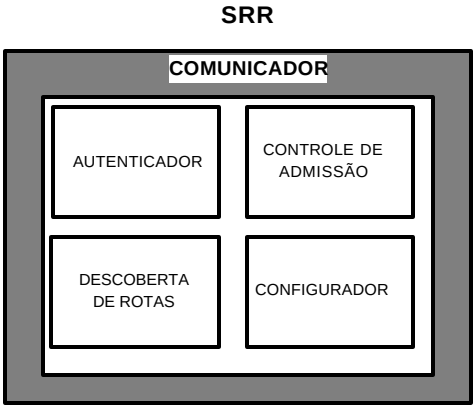


Figura 3.5 - Estrutura Modular de um SRR

A seguir cada um destes módulos serão descritos.

- **Módulo Autenticador**

O módulo autenticador verifica se o computador que está solicitando a reserva tem permissão para fazer a reserva. Quando o SRR é parte de uma rede cliente, ele fará a autenticação dos *hosts* de sua rede. Quando o SRR é parte de uma rede provedora, ele fará a autenticação dos SRRs das suas redes clientes, de outras áreas OSPF do Domínio ou de SRRs de borda dos domínios DS adjacentes. Em qualquer um dos casos, o administrador da rede deverá cadastrar o endereço IP *unicast* de cada *host* ou SRR que tiver permissão para solicitar uma reserva.

- **Módulo de Controle de Admissão**

O módulo de controle de admissão verifica a disponibilidade de recursos da rede para aceitar uma nova reserva. Para isso, duas verificações são necessárias. A primeira é verificar se o computador que está fazendo a requisição de reserva tem disponível a quantidade de recursos solicitada. A segunda é verificar se os caminhos por onde o

fluxo *multicast* irá passar dentro da área de responsabilidade do SRR (ex. área de rede OSPF onde se encontra o SRR) têm disponíveis as quantidades de recursos solicitadas. Esta verificação é feita com o auxílio do módulo de descoberta de rotas. O controle de admissão é um processo complexo e um tema onde há diversas propostas e pesquisas em andamento. Neste trabalho, usaremos a abordagem encontrada na definição dos "*Bandwidth Brokers*" [NJZ99], que utiliza a taxa de pico do fluxo de dados como base para o controle de admissão.

No caso dos SRRs de redes clientes, o administrador da rede deve cadastrar a quantidade de recursos disponíveis para cada *host* em cada PHB (atualmente o único PHB disponível é o "*multicast premium*"). Ele também deve cadastrar a quantidade total de recursos em cada PHB para toda rede, definida no SLA com a rede provedora. Desta forma, o módulo de admissão poderá verificar se o *host* que está fazendo a requisição tem disponível a quantidade de recursos solicitada em um determinado PHB e se com esta nova requisição não é ultrapassado o limite total deste PHB permitido na rede cliente (a quantidade contratada ao provedor).

No caso dos SRRs pertencentes às áreas do Domínio esta verificação é mais complexa. O administrador deve cadastrar todos as quantidades de recursos em cada PHB disponíveis para cada SRR das redes clientes interligadas aos roteadores da área do SRR. Estas quantidades são definidas nos SLAs com os clientes. Também devem ser cadastradas as quantidades de recursos em cada PHB permitido entre a área do SRR e a área de *backbone* do Domínio.

Os SRRs de borda, pertencentes à área de *backbone* do Domínio, devem possuir as quantidades de cada PHB para cada SRR de borda dos Domínios adjacentes na Região DS. Estas quantidades são definidas nos SLAs de parceria com os domínios adjacentes. Além disso, ele deve ter cadastrada as quantidades de recursos disponível para cada SRR das áreas OSPF que compõe o seu Domínio DS.

O processo de admissão consiste em duas etapas. A primeira é verificar a quantidade de recursos disponíveis em um determinado PHB para o *host* ou SRR solicitador. A segunda é verificar se os caminhos por onde o fluxo *multicast* irá passar dentro da rede provedora têm esta quantidade de recursos disponível.

Cada SRR é o responsável pelo processo de admissão de novas reservas dentro de sua rede ou de sua área OSPF no Domínio. No caso de árvores de distribuição

multicast que se espalham por várias redes ou vários domínios DS é necessário que os SRRs de borda destas redes se comuniquem. Nestes casos, o processo de reserva é uma tarefa conjunta dos vários SRRs. Este processo de admissão conjunto depende das informações geradas pelo módulo de descoberta de rotas descrito a seguir.

- **Módulo de Descoberta de Rotas**

O módulo de descoberta de rotas é o responsável pela descoberta dos caminhos (rotas) por onde o fluxo *multicast* irá passar dentro de um domínio, bem como do próximo SRR neste caminho. Estas informações são indispensáveis para o módulo de admissão.

Este módulo é muito simples nos SRRs das redes clientes, pois possuem apenas o endereço do SRR da área OSPF na rede provedora ao qual a rede cliente esta interligada.

Nos SRRs das áreas OSPF no interior do Domínio, este módulo possui uma maior complexidade. Ele recebe as informações de rotas através do anúncio feito pelos roteadores da sua área OSPF e as armazena em um banco de dados. Estes SRRs possuem os endereços dos SRRs das diversas redes clientes interligada aos roteadores da sua área e o endereço do SRR da área de *backbone* do Domínio. Além disso, o módulo implementa o algoritmo de descoberta do menor caminho, usado pelo protocolo OSPF, para descobrir o caminho para onde se estende à árvore *multicast*, bem como os enlaces que farão parte deste caminho.

Os SRRs de borda, pertencente à área de *backbone* do Domínio possuem os endereços dos SRRs de bordas dos domínios adjacentes, bem como os endereços dos SRRs de todas as áreas OSPF do seu Domínio. Neste SRR, todas as rotas do Domínio são armazenadas, pois o roteador OSPF desta área recebe um sumário das rotas das diversas áreas. Além disso, ele armazena as informações de rotas do BGP. Neste módulo são implementados tanto o algoritmo do protocolo OSPF quanto o algoritmo do protocolo BGP.

A partir do endereço IP do receptor que solicitou a reserva e do IP da fonte/grupo *multicast*, os algoritmos de roteamento são executados e como resultado é

gerado um vetor contendo as informações dos caminhos por onde o fluxo *multicast* irá passar e o próximo SRR neste caminho para onde deve ser encaminhada a reserva.

- **Módulo Configurador**

O módulo configurador é o responsável pela configuração dos roteadores que fazem parte da árvore de distribuição *multicast* de uma fonte/grupo. O roteador folha é o primeiro roteador por onde o fluxo *multicast* do transmissor do grupo entra em um Domínio DS. Ele pode ser, por exemplo, o roteador de borda da rede do transmissor. A configuração é feita dinamicamente pelos SRRs destas redes antes do horário inicial do grupo (para reservas antecipadas) ou logo após a requisição da reserva (reservas imediatas). Somente serão configurados os roteadores dos ramos da árvore de distribuição onde haja receptores que conseguiram fazer a reserva. As configurações foram especificadas na descrição da arquitetura.

- **Módulo Comunicador**

O módulo comunicador tem a responsabilidade de implementar a comunicação entre o SRR e todos os outros elementos da arquitetura: os hosts, outros SRRs e os roteadores. Para isso, ele deve utilizar um protocolo que atenda aos requisitos de cada uma destas comunicações, como o RSVP. Nós implementamos um protocolo específico, que está descrito nos itens 3.3.1.6 e 3.3.2.

3.2.5- Estrutura Funcional dos Roteadores

Como visto, há três tipos de roteadores em um ambiente de serviços diferenciados: roteadores folha, roteadores de borda e roteadores internos. As estruturas funcionais destes roteadores são similares e podem ser vistas na Figura 3.6. A estrutura funcional dos roteadores é composta pelos seguintes módulos: classificador, condicionador, escalonador, configurador e comunicador. Os roteadores internos não utilizam o módulo condicionador.

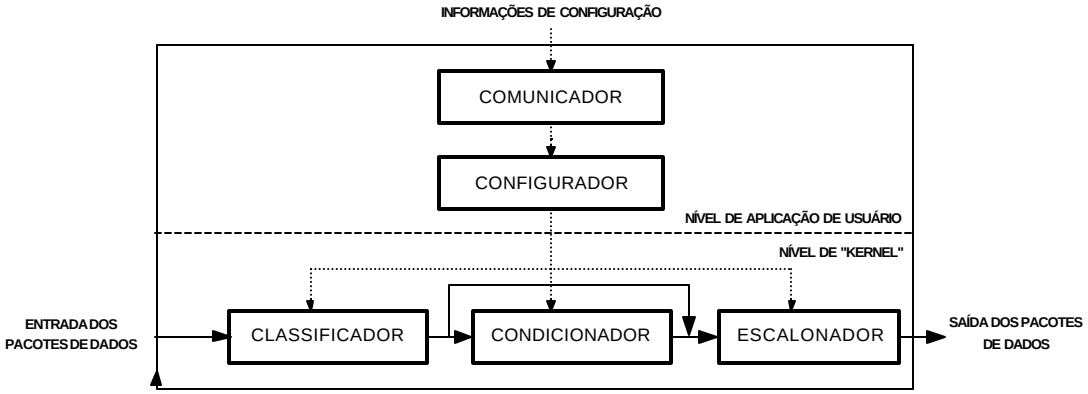


Figura 3.6 – Estrutura Funcional dos Roteadores

O módulo classificador seleciona os pacotes em um fluxo de tráfego baseado no conteúdo de alguma porção do cabeçalho do pacote. Podem-se definir dois tipos de classificadores: o classificador BA (*Behavior-Aggregate*) e o classificador MF (*Multi-Field*). Este módulo encaminha os pacotes que combinam com alguma regra especifica para um elemento condicionador de tráfego, de forma que ele sofra um processamento adicional. A classificação MF é geralmente utilizada nos roteadores de entrada na rede de serviços diferenciados, podendo ser um roteador folha. Ela utiliza um conjunto de quatro campos do cabeçalho do pacote (para fluxos individuais): endereço IP origem, endereço IP destino, porta do aplicativo origem e porta do aplicativo destino. A classificação BA é geralmente utilizada nos roteadores internos e nos roteadores de borda. Alguns roteadores de borda podem utilizar a classificação MF, quando eles são o roteador de entrada de uma rede Diffserv.

O módulo condicionador dos roteadores folha e de borda é compostos pelos sub-módulos marcador, modelador e descartador. O sub-módulo marcador atribui ou modifica um valor de *codepoint* de um pacote. O sub-módulo modelador atrasa alguns ou todos os pacotes em um fluxo ou agregado de dados a fim de adequar o fluxo a um perfil de tráfego, enquanto o sub-módulo descartador descarta os pacotes que não estão de acordo com o perfil (policiamento). O módulo condicionador faz o condicionamento tanto dos fluxos individuais quanto dos agregados de fluxos.

O módulo escalonador implementa disciplinas de escalonamento visando à diferenciação do tratamento dado aos pacotes pertencentes as diferentes classes de serviço.

O módulo configurador é o responsável pela configuração dos módulos de classificação, de condicionamento e escalonamento. Este módulo recebe informações a partir dos SRRs ou agentes de configuração, através do módulo comunicador, por intermédio de um protocolo de comunicação desenvolvido especialmente para este ambiente Diffserv.

3.2.6- Esquema da Interoperabilidade entre os Componentes da Arquitetura

A Figura 3.7 mostra o esquema da interoperabilidade entre os componentes da arquitetura proposta para o tráfego *multicast* com QoS.

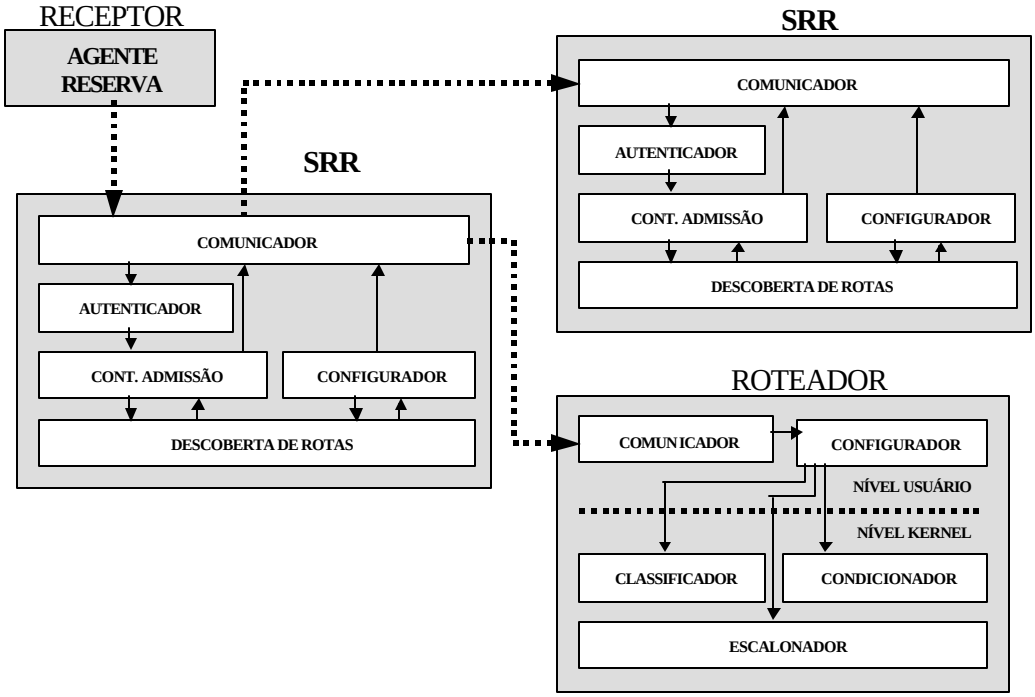


Figura 3.7 – Esquema da Interoperabilidade entre os Componentes da Arquitetura

A fim de permitir que os receptores possam solicitar o perfil e a reserva de um grupo *multicast* aos SRRs, foi necessário criar um novo componente, que é o Agente para Reserva. Os aspectos mais relevantes acerca da implementação serão mostrados na próxima seção.

3.3- Implementação

Esta seção aborda alguns aspectos relevantes na implementação dos elementos que constituem tanto o ambiente de serviços diferenciados quanto o sistema de reserva de recursos. Todos os módulos responsáveis pelo tratamento diferenciado dos pacotes foram desenvolvidos, assim como diversos módulos dos SRRs. Também foram desenvolvidos protocolos de comunicação que permitem a troca de informações entre os elementos da arquitetura. Estes elementos são a base para o tratamento do tráfego *multicast*. Os módulos responsáveis pelo tratamento do tráfego *multicast* não foram desenvolvidos, visto que tanto a especificação do PIM-SSM quanto a especificação do IGMP versão 3 ainda não foram concluídas.

O ambiente de serviços diferenciados é composto por roteadores com sistema operacional TROPIX [T01], sendo os seus módulos desenvolvidos em linguagem ANSI-C. O TROPIX é um sistema operacional multi-usuário e multitarefa, de filosofia UNIX, desenvolvido no Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro (NCE/UFRJ). Foi inicialmente concebido durante os anos de 1982 a 1986 (na época com o nome PLURIX) para o computador PEGASUS, baseado nos processadores MOTOROLA 68010/20, também construído no NCE. Em 1994 foi iniciado o transporte para a linha INTEL de processadores (386, 486, Pentium), e desde 1996 o TROPIX já está operacional em PCs, sendo utilizado em diversos computadores. A fim de implementar o ambiente de serviços diferenciados, foram necessárias algumas alterações e acréscimos no núcleo do TROPIX, bem como a criação de alguns novos aplicativos.

Além disso, um protótipo do sistema de reserva de recursos foi desenvolvido utilizando o ambiente de desenvolvimento Delphi 4.0 com tabelas em Paradox e sistema operacional Windows 98. Este protótipo permite a realização de requisições de reserva para uma determinado canal (S,G) *multicast* e a configuração dos roteadores TROPIX para tal reserva.

3.3.1- Ambiente de Serviços Diferenciados

Esta seção apresenta os aspectos relevantes da implementação do ambiente de serviços diferenciados. Para a construção deste ambiente tornou-se necessária a implementação de novas políticas de escalonamento de pacotes, classificadores e condicionadores de tráfego no TROPIX.

Três módulos foram desenvolvidos: módulo classificador, módulo condicionador e módulo escalonador. A seguir serão apresentadas as alterações feitas no núcleo do TROPIX e os detalhes relevantes da implementação de cada um destes módulos. A seção 2.1 apresenta as estruturas de dados usados pelos TROPIX. Na seção 2.2 é apresentada a implementação do módulo escalonador. Na seção 2.3 é mostrada a implementação do módulo classificador de pacotes. Na seção 2.4 é apresentada a implementação do módulo condicionador de tráfego. Na seção 2.5 é apresentada a forma de configuração dos roteadores, bem como os aplicativos desenvolvido para fazer tais configurações.

3.3.1.1 - Estruturas de Dados do Núcleo do TROPIX para Rede

Esta seção apresenta as estruturas de dados do núcleo do TROPIX que armazenam as informações dos pacotes de dados, que chegam ou são enviados para a rede.

- **Estrutura de um Pacote de Dados no TROPIX**

Todos os pacotes de dados que chegam da rede, através de um dispositivo de rede, ou que devem ser enviados para a rede, como saída gerada por um aplicativo, precisam residir em áreas de memória do computador. O gerenciamento da alocação de novas áreas de memória, bem como a forma de agregação dos dados nos pacotes, enquanto eles trafegam entre as diversas camadas da pilha de protocolos, são fundamentais para a eficiência da transmissão e recepção de dados.

Duas soluções são comumente utilizadas nos Sistemas Operacionais. A primeira delas utiliza pequenos blocos de memória com tamanhos variáveis, conhecidos como

mbufs [MBK96]. O conteúdo de um pacote é representado por um encadeamento de *mbufs*. Durante a passagem do pacote pelas camadas, as novas informações referentes aos cabeçalhos de cada nível de protocolo são acrescidas, agregando novos blocos à lista. A vantagem desta abordagem é o melhor uso da memória, visto que a quantidade de memória alocada é equivalente ao tamanho do pacote, evitando desperdícios. A desvantagem está no fato de que o conteúdo do pacote não está contíguo na memória, devendo ser sequencializado antes de ser enviado ao dispositivo físico de rede (placa Ethernet, por exemplo).

O TROPIX utiliza uma outra abordagem, que consiste na alocação de grandes blocos de memória (*buffers*) para armazenar os pacotes. Nesta solução, o tamanho do bloco é igual ao tamanho máximo que um pacote pode ter, tipicamente 1500 octetos para cada datagrama IP. A grande vantagem desta abordagem é que o conteúdo do pacote reside em uma área contígua de memória, o que evita cópias no momento da transmissão pelo dispositivo físico. A desvantagem é a alocação da maior quantidade de memória possível, independentemente do tamanho do pacote.

A estrutura de dados que representa um pacote no TROPIX é chamada ITBLOCK, dividindo-se em três áreas: uma de controle, outra de cabeçalho dos protocolos e outra de dados. A Figura 3.8 ilustra um ITBLOCK.

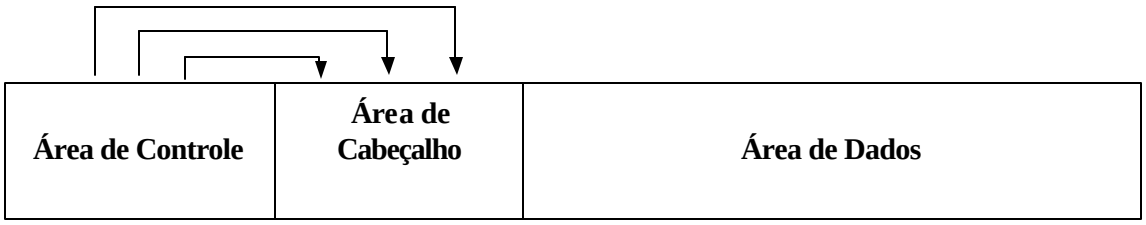


Figura 3.8 – Estrutura de um ITBLOCK

A área de controle é composta por variáveis de controle dos protocolos, ponteiros para os cabeçalhos dos protocolos dentro da área de cabeçalho e ponteiros para as filas de ITBLOCKs. Além destas informações, existem alguns octetos reservados para necessidades de implementações futuras. A área de cabeçalho contém as informações dos cabeçalhos da pilha de protocolos responsável pela condução da

mensagem que está sendo carregada na área de dados. A área de dados armazena o conteúdo útil dos pacotes.

- **Estruturas Utilizadas pelos Dispositivos de Rede**

O TROPIX possui uma estrutura de dados para cada dispositivo de rede, contendo todas as informações de controle necessárias para o controle do dispositivo e o gerenciamento do envio e recebimento de pacotes. É através destas estruturas que há a interação da pilha de protocolos do núcleo do TROPIX com o dispositivo de rede. Estas estruturas são formadas por diversos itens de informação, como por exemplo, ponteiros para as filas dos pacotes que devem ser enviados, disciplina da fila usada para o envio dos pacotes, etc. Atualmente, o TROPIX reconhece dois tipos de placa de rede: as placas de rede compatíveis com o modelo NE2000 de 10 Mbps e as placa RealTek de 100 Mbps.

3.3.1.2- Módulo Escalonador

Em nossa implementação, o módulo escalonador é o elemento responsável por garantir que os pacotes tenham o devido PHB em cada nó de um domínio. O modelo Diffserv não define os mecanismos para a implementação dos escalonadores, mas apenas as características esperadas dos PHBs. O mecanismo utilizado nesta implementação do escalonador foram as disciplinas de escalonamento *Priority Queuing* (PQ), *Deficit Roudin Robin* (DRR) e *Start-Time Fair Queuing* (STFQ). Além da criação destas novas disciplinas de escalonamento no núcleo do TROPIX, houve a reformulação da disciplina de filas FIFO que já estava implementada.

- **Reestruturação das Chamada às Funções de Enfileiramento e Desenfileiramento**

Quando um pacote vai ser enviado através de um determinado dispositivo de rede, há uma chamada à função de escrita do *driver* deste dispositivo. Esta função, por sua vez, coloca o pacote na fila de envio do *driver*. Quando o dispositivo está realmente

pronto para fazer o envio, é chamada uma função do *driver* que retira o pacote da fila do *driver*, acionando o dispositivo físico que transmitirá o pacote.

Com a disciplina de filas FIFO era a única disciplina implementada no TROPIX. Os procedimentos de enfileiramento e desinfileiramento estavam embutidos no código das rotinas de envio e transmissão de pacote. A fim de utilizar diferentes tipos de disciplinas de escalonamento, houve a necessidade de modificar os procedimentos de enfileiramento e desinfileiramento dos pacotes na fila do *driver*. Cada disciplina de fila tem as suas próprias funções de enfileiramento e desinfileiramento.

A Figura 3.9 mostra o diagrama de blocos da estrutura lógica de enfileiramento e desinfileiramento de pacotes implementada na camada de interface do Núcleo do TROPIX.

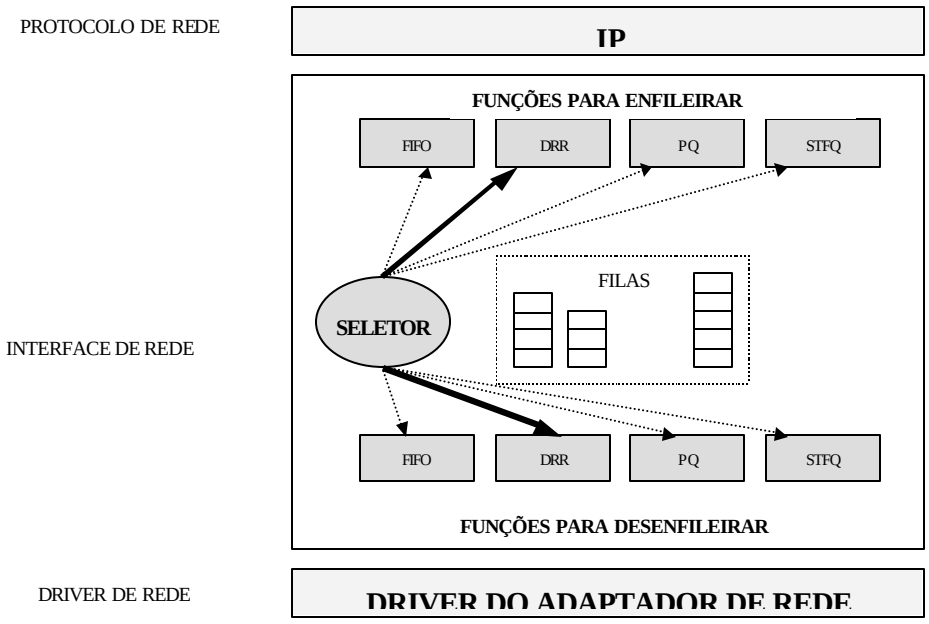


Figura 3.9 - Diagrama da Estrutura Lógica de Enfileiramento e Desenfileiramento de Pacotes

Uma importante característica da implementação do módulo escalonador é a possibilidade da troca dinâmica das disciplinas de escalonamento (com o roteador em funcionamento). A implementação também possibilita o uso de diferentes disciplinas de

escalonamento simultaneamente (cada placa de rede do roteador pode ter uma disciplina de fila diferente das demais placas).

- **Estruturas de Dados para as Disciplinas de Escalonamento**

A implementação das disciplinas de escalonamento utiliza um conjunto de filas. O uso destas filas depende do tipo de disciplina. Por exemplo, a disciplina FIFO só usa uma fila. Já a DRR utiliza todas as filas disponíveis. Como as disciplinas de escalonamento são implementadas em nível de interface de rede, e cada dispositivo de rede pode ter a sua própria disciplina de escalonamento, algumas informações de controle das filas foram acrescidas à estrutura original do *driver* do adaptador de rede.

As informações que são gerais e comuns a todas as filas estão contidas na estrutura *NETQUEUE*. As informações particulares a cada fila encontra-se na estrutura *IDQUEUE*. Somente a implementação da disciplina STFQ precisou acrescentar campos à estrutura de dados ITBLOCK do TROPIX, pois existem informações particulares a cada pacote.

A estrutura *NETQUEUE* constitui-se dos seguintes campos:

- Disciplina corrente (FIFO, DRR, STFQ, PQ);
- endereços das funções de enfileiramento e desenfileiramento;
- número de pacotes pendentes;
- *Round Number* da disciplina STFQ;
- conjunto de NUM_QUEUES filas de pacotes;

Cada uma das filas do conjunto possui a seguinte estrutura (IDQUEUE):

- Contador de bits em *deficit* usado na disciplina DRR;
- Peso relativo da fila;
- Quantidade de bits servida por vez;
- *Conection finish number* usado na disciplina STFQ;
- Ponteiro para o primeiro e o último pacote da fila;

3.3.1.3-Implementação das Disciplinas de Escalonamento

Nesta seção é apresentado um resumo da implementação das disciplinas DRR e STFQ, visto que tanto a disciplina FIFO quanto a *Priority Queuing* são bastante conhecidas.

- **DRR (*Deficit Round Robin*)**

Cada classe de serviço diferenciado é representada por uma fila de pacotes no roteador. Assim que um pacote chega ao roteador, ele é inserido na fila correspondente à sua classe de serviço. Como visto, a disciplina DRR modifica a disciplina WRR a fim de possibilitar a manipulação de filas com tamanhos de pacotes variados, sem a necessidade de conhecer estes tamanhos antecipadamente. Na DRR, cada fila possui um contador (*deficit counter*) inicializado em 0 (zero). O servidor da disciplina de escalonamento DRR visita uma fila de cada vez, servindo uma quantidade de bits (*quantum*). O primeiro pacote da fila visitada é servido caso seu tamanho seja menor ou igual ao *quantum*. Se o pacote é maior, o *quantum* é acrescido ao contador da fila. Se o servidor visitar uma fila e a soma do contador da fila e do *quantum* for maior ou igual ao tamanho do pacote, o pacote será servido e o contador da fila será diminuído de um valor igual ao tamanho do pacote. A disciplina DRR também possui uma versão que utiliza pesos, em que o peso de cada fila é multiplicado pelo valor do *quantum*. Para que seja garantido que pelo menos um pacote seja servidor por vez, o tamanho do *quantum* deve ser igual ao maior tamanho de pacote possível.

A implementação disciplina DRR no TROPIX utiliza um conjunto de filas que são servidas a cada rodada. O número de filas é definido por NUM_QUEUES. O número de bits que são servidos em cada fila a cada rodada é definido pela expressão q_weight * QUANTUM, onde QUANTUM é a quantidade de bits *default*. Três informações específicas a cada fila foram acrescentadas à estrutura IDQUEUE. A primeira foi a *q_def_count*, que contabiliza os bits em déficit de uma fila. A segunda foi a *q_quantum*, que define a quantidade de bits de cada fila que são servidos em cada rodada. A terceira foi a *q_weight*, que indica o peso que uma determinada fila possui. Quanto maior o peso, maior será o número de bits servido por vez.

- **STFQ (*Start-Time Fair Queuing*)**

Uma variação da disciplina *Weighted Fair Queuing* (WFQ) é a disciplina STFQ. Nela, além do cálculo do *finish number*, calculado na WFQ para a indexação dos pacotes que serão servidos, há o cálculo o *start number*, que é o número relativo ao momento da chegada do pacote. Diferentemente da WFQ, os pacotes são servidos na ordem do seu *start number*.

O campo *q_cfnumber* (*conection finish number*) foi acrescentado à estrutura IDQUEUE, pois é necessário para cada fila. Também foi necessário criar dois novos campos na estrutura ITBLOCK, particulares a cada pacote. O primeiro foi o *it_snumber*, que contém o valor do start-number do pacote. O segundo foi o campo *it_fnumber*, que contém o valor do *finish-number* do pacote. Estes campos foram criados na estrutura ITBLOCK, pois cada pacote ao entrar na fila deve possuir estas informações. A função de enfileiramento coloca os pacotes segundo a ordem do momento de chegada do pacote.

Na implementação, apenas a fila zero armazena os pacotes. As demais são utilizadas apenas para o armazenamento das informações de controle.

3.3.1.4- Classificador dos Pacotes

Como visto, os classificadores de pacotes selecionam os pacotes, que entram em um domínio DS, a partir do conteúdo de alguma porção do seu cabeçalho. Eles podem ser do tipo MF (*Multi-Field*) ou BA (*Behavior-Aggregate*). Quando chega um novo pacote no roteador, o classificador verifica se o pacote faz parte de um fluxo ou agregado de fluxos com reserva. Isto ocorre por meio da verificação da existência de um estado descritor de reserva relativo ao fluxo ou agregado ao qual pertence o pacote. Caso haja a reserva, o pacote é enviado para o condicionador de tráfego. Se não, ele é enviado diretamente para o dispositivo de rede.

A fim de receber um tratamento diferenciado, cada fluxo ou agregado de fluxos deve ter um descritor de reserva armazenado no roteador. Este descritor fica ativo na memória principal do roteador e descreve as características do fluxo ou agregado. Há uma estrutura de dados para o descritor de reserva MF e outra para o descritor BA.

Nos roteadores que fazem a classificação MF o descritor possui os seguintes campos:

- Endereço IP do host que está enviando e do host que está recebendo;
- porta da aplicação do host que está enviando e do host que está recebendo;
- tempo entre atualizações (em ms);
- tamanho de incremento do *token*;
- profundidade do balde;
- tamanho do *token*;
- valor do *codepoint* que indicando o tipo de serviço.

Nos roteadores que fazem a classificação BA, o descritor possui a seguinte estrutura de dados:

- Tempo entre atualizações (em ms);
- tamanho de incremento do *token*;
- profundidade do balde;
- tamanho do *token*;
- valor do *codepoint* que indicando o tipo de serviço.

Para cada pacote recebido, o roteador deve verificar se há um descritor de reserva associado ao fluxo ou agregado de fluxo. Sendo este procedimento crítico, optamos por armazenar os descritores em uma tabela de dispersão (*hashing*), cujo o acesso é feito da seguinte função:

```
#define       HASH(x,y)    (((x) + (y)) % STMF_HT_SIZE)
```

A função de dispersão utiliza apenas os endereços IP origem e destino. Nada impede que ela seja estendida para levar outras informações da conexão, como as portas da aplicação. Caso o roteador tenha o classificador BA, a tabela é um vetor que possui tantas entradas quantos forem as classes de serviços diferenciados.

3.3.1.5- Condicionador de Tráfego

O condicionador de tráfego é o elemento responsável pela adequação de um fluxo ou agregado de fluxo de dados, entrando em um domínio DS, a um determinado perfil de tráfego estabelecido no SLA (*Servide Level Agreement*). Geralmente, o condicionador é composto pelos seguintes elementos: medidor, marcador, modelador e descartador. O módulo descartador, responsável pelo policiamento dos pacotes, não foi implementado. O que será feito em um trabalho futuro.

Após a classificação, os pacotes de um fluxo de dados são conduzidos para o condicionador. Um medidor de tráfego é utilizado para medir alguma característica temporal do fluxo ou agregado de fluxo de dados, verificando se ele está dentro ou fora do perfil acordado no SLA. O resultado desta comparação é usado para uma ação de marcação, descarte ou modelagem. Na implementação, só é feita a marcação e a modelagem do tráfego.

O módulo marcador é o responsável pela atribuição de valores de *codepoints* aos pacotes dos fluxos e agregados de fluxos de dados, entrando em um domínio DS, segundo a classe de serviços para eles reservada. Na implementação, a marcação só ocorre nos roteadores MF, pois é por eles que um fluxo entra pela primeira vez em um domínio DS.

O módulo modelador atrasa alguns ou todos os pacotes de um fluxo ou agregado de fluxo, a fim de ajustá-los a um determinado perfil. Para isso, foi utilizado o algoritmo de balde de *tokens* (*Token Bucket*). Toda vez que um pacote chega a este módulo, é verificado se há um *token* disponível. Se houver, o pacote é passado para o adaptador de rede. Caso não haja, o pacote é armazenado até a chegada de um novo *token*. Na implementação, cada *token* representa uma quantidade de bits, definida na estrutura do descritor de reserva.

Um descritor de reserva é criado quando há uma requisição de reserva para um fluxo ou agregado de fluxos. Então, é definido o tamanho de *token*, que é ajustado conforme a taxa máxima de transmissão do fluxo ou agregado, para o qual o descritor foi criado, e a taxa de chegada dos *tokens*. O tempo entre as chegadas dos *tokens* é convertido em número de ticks do relógio de tempo real do sistema. Na criação do descritor, a função de atualização do "balde de *tokens*" é programada para se repetir

com este período fixo de "*ticks*". A fim de minimizar a sobrecarga de processamento das constantes atualizações do "balde de *tokens*", os descritores que não tiverem pacotes em suas filas de espera terão sua atualização desativada. Esta só será reativada quando um pacote referente ao descritor chegar ao roteador.

A função de atualização é a responsável por todos os procedimentos do algoritmo do "balde de *tokens"*, como a chegada de um *token*, a verificação do transbordamento do balde e a transmissão dos pacotes que estão esperando para serem transmitidos (os pacotes que estavam fora do padrão de transmissão).

3.3.1.6- Configuração do Ambiente

Nesta seção serão apresentados os detalhes da implementação dos elementos de configuração dos roteadores, sendo apresentadas as novas funções, os aplicativos e o protocolo de configuração.

- **Funções de Configuração**

Visando uma maior eficiência, as funções que implementam o ambiente de serviços diferenciados estão localizadas no núcleo do sistema operacional. Para disponibilizá-las, foram especificadas novas chamadas à função *ioctl*.

FUNÇÃO	DESCRIÇÃO
NET_GET_DISCIPLINE	Obtém a disciplina de fila corrente de um adaptador de rede.
NET_SET_DISCIPLINE	Modifica a disciplina de fila de um adaptador de rede.
I_STMF_INSERT	Cria um estado descritor de reserva *Multi-Field*.
I_STMF_DELETE	Exclui um estado descritor de reserva *Multi-Field*.
I_STBA_INSERT	Cria um estado descritor de reserva *Behavior Aggregate*.
I_ROUTER_TYPE	Exclui um estado descritor de reserva *Behavior Aggregate*.

Tabela 3.1- Novas Funções IOCTL

A mudança de disciplina pode ser solicitada por um aplicativo do superusuário através de um *ioctl*. A disciplina só é efetivamente trocada quando as filas se esvaziarem. A configuração dos roteadores, abrangendo a inserção e remoção dos

estados descritores de reserva, bem como a mudança do tipo de roteador, é também realizada através de *ioctl*. As novas funções podem ser vistas na Tabela 3.1.

- **Aplicativos de Configuração**

Um servidor de configurações foi desenvolvido como aplicativo de usuário para fazer as diversas chamadas à *ioctl*. Ele é executado durante a fase de *boot* do roteador TROPIX e é chamado *sconfig.c*.

Código do Procedimento	Procedimento	Descrição
RR	Requisição de Reserva	Faz a requisição de uma reserva, passando como parâmetros todas as informações necessárias para criar um estado descritor de reserva, como: endereço IP origem, endereço IP destino, porta do aplicativo origem, porta do aplicativo destino, identificador do tipo de serviço (*codepoint*) e taxa de transmissão.
LD	Lista os dispositivos de rede	Obtém os nomes dos dispositivos de rede que estão ativos no roteador.
CD	Troca de Disciplina	Faz a troca da disciplina de um determinado dispositivo de rede. Os seguintes parâmetros são necessários: nome do dispositivo e número da disciplina de escalonamento.
CR	Troca o Tipo de Roteador	Faz a troca do tipo de roteador, passando como parâmetro o número que identifica o novo tipo de roteador. Os tipos de roteadores são: roteador de classificação *multi-field*, roteador de classificação *behavior-aggregate* e roteadores internos (sem classificação).
SC	Configuração do *codepoint*	Cria um novo codepoint e o configura, informando o valor do novo *codepoint*, a taxa de transmissão máxima permitida e a quantidade máxima de bits de rajada (profundidade do balde de *tokens*).

Tabela 3.2- Códigos das Ações Usadas no Protocolo de Configuração dos Roteadores.

A sintaxe para a sua execução é:

sconfig *porta tipo-de-roteador*

A porta indica em qual porta o aplicativo é acessado. E tipo-de-roteador define se a configuração será para um roteador do tipo multi-field ou behavior-aggregate.

Tabela 3.2 apresenta as diversas opções de configuração dos roteadores. Os códigos apresentados na tabela podem ser utilizados tanto para o servidor MF quanto para o servidor BA, exceto o código "SC", que é utilizado apenas para o servidor BA.

Também foi desenvolvido um aplicativo cliente para interagir com o servidor de configurações dos roteadores. Este aplicativo foi desenvolvido no ambiente de programação Delphi 4.0, utilizando sistema operacional Windows 98, criando uma interface gráfica para a configuração dos roteadores TROPIX.

Um protocolo foi desenvolvido especificamente para a configuração dos roteadores. Ele é composto basicamente por um campo que indica o código do procedimento pretendido no servidor de configurações, seguido por outros campos com as informações necessárias para a configuração do roteador.

3.3.2- Sistema de Reserva de Recursos

Um protótipo do sistema de reservas de recursos da arquitetura proposta neste trabalho foi implementado. Os SRR (Servidores de Reserva de Recursos) foram desenvolvidos utilizando o ambiente de desenvolvimento Delphi 4.0 com tabelas em Paradox e sistema operacional Windows 98. Ele é composto por três aplicativos que implementam as suas funcionalidades. Esta abordagem permitiu uma divisão lógica do SRR, facilitando o desenvolvimento paralelo dos aplicativos. Além disso, a divisão das funcionalidades em aplicativos distintos propicia um futuro desenvolvimento *multi-thread* do SRR.

O primeiro aplicativo que compõe o SRR é o cadastro de configurações que permite o cadastramento das configurações necessárias ao funcionamento do SRR, tais como: usuários que têm permissão para reserva (receptores ou SRRs), tipos de serviços Diffserv, quantidade de recursos disponíveis para cada usuário em cada serviço Diffserv, endereços de SRRs adjacentes e topologia da rede. Neste último são cadastrados os roteadores, os seus adaptadores de rede e a taxa de transmissão de cada enlace a ele interconectado.

O segundo aplicativo é o responsável por implementar o módulo de autenticação, o módulo de controle de admissão e o módulo de descoberta de rotas. Este aplicativo está sempre ativo esperando uma solicitação de perfil ou reserva.

Código do Procedimento	Procedimento	Descrição
PR	Pedido de Reserva	Faz o pedido de reserva, indicando: endereço IP da fonte *multicast*, endereço IP do grupo *multicast*, identificador do tipo de serviço (*codepoint*), taxa de transmissão, data de inicio da sessão, hora inicial, data final, hora do final da sessão, endereço IP do host requisitante, porta da aplicação requisitante.
RR	Requisição de Reserva	Faz a requisição de uma reserva, passando como parâmetros todas as informações necessárias para criar um estado descritor de reserva, como: endereço IP origem, endereço IP destino, porta do aplicativo origem, porta do aplicativo destino, identificador do tipo de serviço (*codepoint*) e taxa de transmissão.
CR	Confirmação de Reserva	Faz a confirmação da reserva para uma determinada fonte/grupo.
NR	Negação de Reserva	Indica que foi negada a reserva para uma determinada fonte/grupo.
PP	Pedido de Perfil	Faz o pedido de perfil de um tráfego *multicast* de uma fonte/grupo.
CP	Confirmação de Perfil	Contém todas as informações do perfil solicitado, como: endereço IP da fonte *multicast*, endereço IP do grupo *multicast*, identificador do tipo de serviço (*codepoint*), taxa de transmissão, data de inicio da sessão, hora inicial, data final, hora do final da sessão, endereço IP do *host* requisitante, porta da aplicação requisitante.
NP	Negação de Perfil	Indica que a requisição do perfil foi negada. Provavelmente não há um perfil para a fonte/grupo.

Tabela 3.3- Códigos das Ações Usados no Protocolo de Comunicação dos SRRs

O terceiro aplicativo implementa o módulo configurador. Este aplicativo está constantemente verificando a tabela de reservas, checando se há reservas para configurar. Caso haja, este módulo faz a configuração dos roteadores de acordo com as informações armazenadas na tabela de reservas. Ele é o responsável tanto pelas reservas imediatas quanto pelas reservas antecipadas.

Para permitir que os receptores possam solicitar o perfil e a reserva de uma sessão *multicast* aos SRRs, foi necessário o desenvolvimento de uma outra aplicação hospedada nos receptores, denominada Agente para Reserva.

Além disso, foi necessário desenvolver um protocolo de comunicação entre o agente de reserva e os SRRs, e dos SRRs entre si. Este protocolo é composto por um campo que define a ação e outro com as informações necessárias para que a ação seja executada. A Tabela 3.3 apresenta as diversas ações especificadas para o protocolo.

O protocolo utilizado pelos SRRs para configurar os roteadores é o protocolo descrito na seção 3.3.1.6.

CAPÍTULO 4

Validação e Resultados

4.1- Introdução

Este capítulo apresenta o ambiente utilizado para a validação dos módulos que implementam os serviços diferenciados nos roteadores, as ferramentas utilizadas nos testes e os resultados obtidos.

4.2- Ambiente de Testes

O ambiente onde foram realizados os experimentos é composto por sete microcomputadores interligados diretamente por meio de uma placa de rede *Realtek* de 100 Mbps. A configuração dos computadores pode ser vista a seguir:

- Microcomputador PENTIUM III 550 MHz com 64 MB de memória RAM.

Três deles funcionavam como roteadores e os demais funcionavam como *hosts*, criando 6 sub-redes. A topologia da rede formada pelos sete computadores pode ser vista na Figura 4.1.

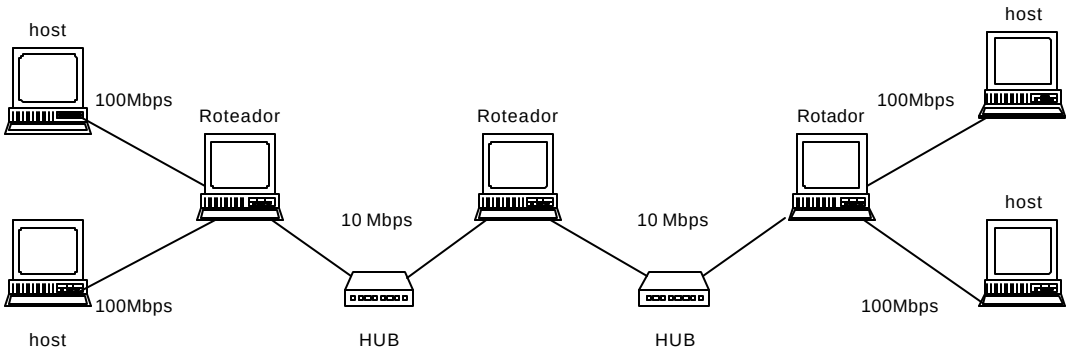


Figura 4.1- Topologia da Rede de Testes

Como pode ser visto, os enlaces que ligam os *hosts* aos roteadores são de 100 Mbps e os enlaces que ligam os roteadores são de 10 Mbps. Os enlaces de 10 Mbps criam uma região de congestionamento nos roteadores, permitindo verificar se eles conseguem dar uma tratamento diferenciado aos fluxos com reserva.

4.3- Ferramentas de Geração de Tráfego

Os geradores de tráfego são ferramentas utilizadas para a avaliação do desempenho de uma rede. Eles podem ser utilizados para medir diversas características de tráfego de dados, como a perda, o atraso e a variação do atraso dos pacotes de dados. Existem diversos geradores de tráfego disponíveis, entre eles o Netperf [N01] e o TGM [LA00].

Usualmente, os geradores de tráfego possuem uma baixa precisão nas medidas do atraso dos pacotes na rede para altas taxas de transmissão e tamanhos de pacotes pequenos, apresentando uma grande variabilidade entre as medidas obtidas. Esta variabilidade está relacionada à sobrecarga do sistema operacional no processamento dos pacotes e na contabilização destes atrasos. Nestas condições, há uma grande perda de tempo dos pacotes no contexto da aplicação geradora de tráfego e do sistema operacional, devido a enfileiramentos e escalonamentos. Isto causa uma dificuldade na interpretação das medidas, pois torna-se difícil distinguir entre o atraso do pacote na rede e o atraso introduzido pelo ambiente gerador de tráfego. Esta seção analisa a implementação de um novo gerador de tráfego que visa minimizar tais problemas e foi desenvolvido para validar os módulos que implementam os serviços diferenciados no TROPIX.

4.3.1- Ambiente para Geração de Tráfego

A estrutura escolhida para geração de tráfego envolve dois aplicativos, que são executados em computadores diferentes: o gerador de pacotes e o refletor. Os pacotes gerados são marcados com o horário de saída de cada um deles. Ao receber o pacote de volta do refletor, é realizada a contabilização do atraso sofrido por cada pacote no seu percurso pela rede.

Dois tipos de geradores foram implementados: o primeiro gera pacotes com tamanho constante; o segundo, com tamanho variado. Este último foi criado a fim de que o tráfego da rede de teste se assemelhasse ao tráfego da Internet.

O aplicativo gerador é composto por dois módulos, responsáveis pela transmissão dos pacotes gerados e pela recepção dos pacotes refletidos. Estes módulos foram implementados em *threads* distintas, devido à relativa independência de execução entre eles.

O módulo transmissor do aplicativo gerador é o responsável pela geração dos pacotes a uma taxa constante. Os pacotes são direcionados para o computador que está executando o aplicativo refletor.

O módulo receptor do aplicativo gerador é o responsável pela recepção dos pacotes que retornam do refletor e pela contabilização do atraso sofrido pelos pacotes na rede. Todas as estatísticas são geradas neste módulo.

O aplicativo refletor é um elemento importante para o funcionamento dos geradores, apesar de sua simplicidade. O refletor recebe os pacotes e os retorna, marcando-os com os horários de chegada e saída. Estas informações são importantes para o cálculo do tempo gasto pelo pacote na rede, como será visto na próxima seção.

4.3.2- A Implementação do Gerador

Os aplicativos gerador e refletor foram implementados em linguagem ANSI-C e são executados no TROPIX.

- **Protocolo de Transporte de Dados utilizado pelos Geradores de Tráfego**

Usualmente, os geradores de tráfego utilizam o protocolo de transporte UDP, que não é orientado a conexão. Optamos por criar um protocolo específico, utilizando para isso o modo-de-acesso-interno, denominado RAW, que permite o envio e a recepção de datagramas IP com protocolos de transporte definidos pelo usuário. O protocolo utilizado pelos geradores é identificado pelo número 23. O núcleo do TROPIX foi alterado para tratar de forma diferenciada os pacotes deste protocolo.

Os pacotes de dados gerados são compostos pelos seguintes campos: horário de saída do pacote no gerador, horário de chegada dos pacotes no gerador, horário de

chegada no refletor, horário de saída do refletor, número de seqüência do pacote e caracteres de preenchimento. Os campos de horário são formados por dois sub-campos que contêm, respectivamente, o número de segundos e microsegundos com relação ao horário de referência do relógio dos sistemas UNIX.

- **Contabilizando o Atraso dos Pacotes**

Como visto, o aplicativo gerador de tráfego é implementado em duas *threads*: uma transmissora e outra receptora de dados. Esta abordagem é geralmente usada em aplicativos como o "*ping*" e outros aplicativos geradores de pacotes. O problema ocorre quando as taxas de transmissão são altas. Como o tempo entre transmissões de pacotes está na ordem de poucos microsegundos e há uma disputa entre as duas *threads* pelo processador, o tempo de escalonamento torna-se crítico, o que é extremamente prejudicial na hora de contabilizar a chegada dos pacotes. Muitas vezes, a *thread* receptora não é escalonada a tempo, pois a transmissora está enviando um pacote. Isto cria uma defasagem no cálculo do atraso do pacote. Além da disputa entre as duas *threads*, ambas concorrem com alguns *daemons* do sistema operacional, como os relacionados à Internet e ao sincronismo do *cache* de disco.

Sendo o TROPIX um Sistema Operacional desenvolvido na UFRJ, sobre o qual temos completo controle e conhecimento, utilizamos uma outra alternativa para solucionar este problema: postergar a amostragem de tempo até o instante em que o pacote é efetivamente transmitido ou recebido. Em outras palavras, a amostragem do tempo é feita no próprio *driver* da placa de rede, o que permite desprezar completamente o atraso que o pacote sofreu devido a procedimentos inerentes ao sistema operacional. Assim, o horário de saída do pacote só é amostrado e copiado para a área de dados do pacote no momento em que este é realmente transmitido pelo dispositivo físico de rede. Evidentemente, este procedimento só é realizado para os pacotes com número do protocolo de transporte igual a 23.

Quando o pacote chega ao computador de origem, retornando do refletor, o *driver* do dispositivo de rede é imediatamente ativado (rotina de interrupção), fazendo a amostragem do horário de chegada com extrema precisão. Neste caso, o tempo obtido não é posto diretamente na área de dados do pacote, mas em um campo da estrutura de controle do pacote. Isto evita que o *driver* faça acesso à área de dados dos pacotes

dentro de uma rotina de interrupção. Somente em um nível superior, onde a identificação do protocolo usado na geração do tráfego é realizada, o tempo registrado pelo *driver* é transferido para a área de dados do pacote.

Mesmo com estas modificações, percebemos ainda alguma variabilidade na marcação do tempo gasto pelo pacote em seu percurso pela rede e concluímos que era devida ao atraso causado pelo aplicativo refletor. Em algumas situações, como altas taxas de transmissão, havia uma perda de tempo aleatória no refletor, também devido ao seu escalonamento pelo S.O. A solução foi acrescentar mais duas amostragens de tempo, feitas no computador refletor de forma análoga às amostragens feitas no gerador: assim que um pacote chega ao refletor, o tempo é amostrado no *driver* da interface de rede; o mesmo ocorre no momento da saída do pacote, após sua "reflexão". Desta maneira, a *thread* receptora do gerador calcula o tempo gasto pelo pacote no percurso na rede da seguinte maneira:

atraso = (tempo_chegada_gerador – tempo_saída_gerador) –
(tempo_saída_refletor – tempo_chegada_refletor)

Assim, foi possível fazer medidas bastante precisas do atraso da rede, pois os tempos de processamento interno dos pacotes não influenciam nas medidas. Vale observar que os relógios dos computadores onde executam o gerador e o refletor não precisam estar sincronizados.

4.3.3- Resultados do Desempenho da Ferramenta de Geração de Tráfego

Esta seção apresenta os resultados comparativos de diversos experimentos entre o gerador de tráfego constante com amostragem do tempo no *driver* (*gtc_drv*) e com amostragem do tempo na aplicação (*gtc_usr*). Além disso, os resultados são confrontados com uma outra ferramenta de geração de tráfego chamada TGM [LA00], desenvolvida no LAND/UFRJ. Esta ferramenta foi implementada para diversos sistemas operacionais e protocolos. Nós utilizamos a versão do TGM para o sistema operacional LINUX, que utiliza o protocolo UDP sobre *Ethernet*.

4.3.3.1- Ambiente de Teste

O ambiente onde foram realizados os experimentos é composto por dois microcomputadores interligados diretamente por meio de uma placa de rede *Realtek* de 100 Mbps funcionando no modo *Full Duplex*. O modo *Full Duplex* evita as colisões e disputas pelo meio de comunicação inerentes ao algoritmo CSMA/CD usado pelo protocolo *Ethernet*, tornando as medidas mais exatas. A configuração dos computadores pode ser vista a seguir:

- Microcomputador PENTIUM II 350 MHz com 128 MB de memória RAM.
- Microcomputador PENTIUM MMX 250 MHz com 96 MB de memória RAM.

Os aplicativos de geração de tráfego foram executados no microcomputador PENTIUM II, enquanto os aplicativos refletores foram executados no microcomputador PENTIUM MMX. A Figura 4.2 apresenta o ambiente de teste.

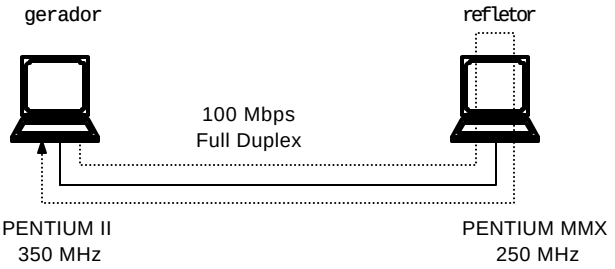


Figura 4.2- Ambiente de Teste

A seguir são apresentados os tipos de experimentos e seus resultados.

4.3.3.2- Medidas Comparativos

Foram realizados dois experimentos a fim de mostrar as vantagens do gerador de tráfego *gtc_drv*, que faz a amostragem do tempo no *driver*, sobre os geradores de tráfego *gtc_usr* e TGM, que fazem amostragem no escopo da aplicação. O primeiro experimento gera pacotes com tamanho fixo de 1450 octetos, fazendo variar a taxa de transmissão. O segundo experimento fixa em 10 Mbps a taxa de transmissão, fazendo variar os tamanhos dos pacotes gerados. Em cada experimento, foram feitas dez rodadas de medições, com 5000 pacotes em cada uma delas.

- **Primeiro Experimento**

O primeiro experimento objetiva aferir a influência de diferentes taxas de transmissão sobre os atrasos medidos. As taxas variaram de 1 Mbps até 35 Mbps, que é o limite máximo de taxa de transmissão para o ambiente proposto, dadas as características dos equipamentos utilizados.

No gráfico da Figura 4.3, apresentamos os valores médios obtidos nas dez rodadas de medição, para cada aplicativo.

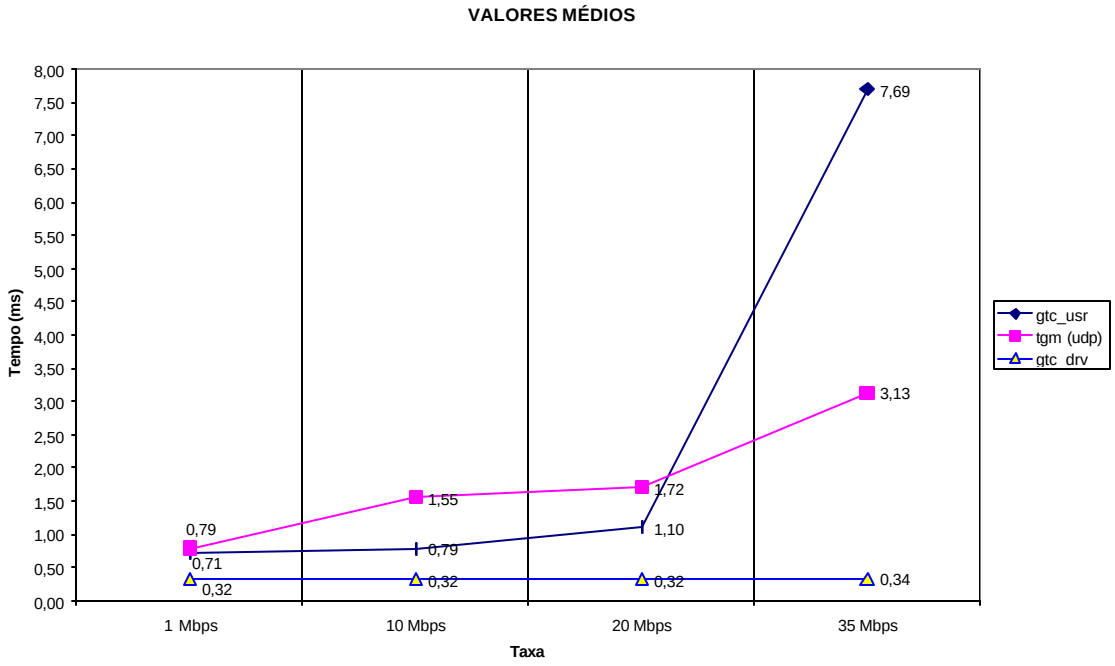


Figura 4.3 – Gráfico da Média dos Tempos de *Round Trip* para Diferentes Taxas de Transmissão.

Os resultados mostram que o gtc_drv mantém um tempo médio constante para qualquer taxa de transmissão, o que não ocorre com os demais geradores. Os geradores que fazem amostragem na aplicação começam a perder a sua precisão quando as taxas de transmissão aumentam, devido à sobrecarga do sistema operacional. Além disso, o tempo medido pelo gtc_drv é bem menor, pois não inclui o tempo gasto pelos pacotes no percurso do aplicativo gerador e refletor até o sistema operacional.

Além disso, a sobrecarga do sistema operacional em altas taxas de transmissão cria uma grande variabilidade nas medidas. Nos gráficos seguintes, são apresentados os

valores mínimo, médio e máximo dos atrasos nas dez rodadas de medição, para cada aplicativo.

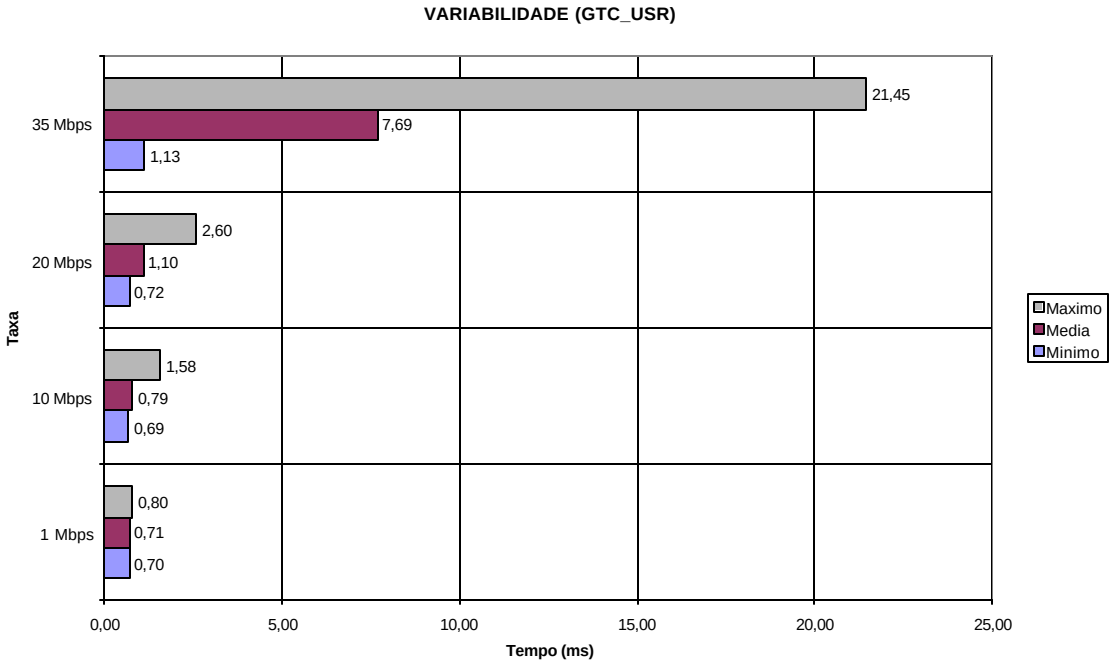


Figura 4.4 – Variabilidade dos Tempos de Round Trip no Aplicativo *GTC_USR* para Diferentes Taxas de Transmissão.

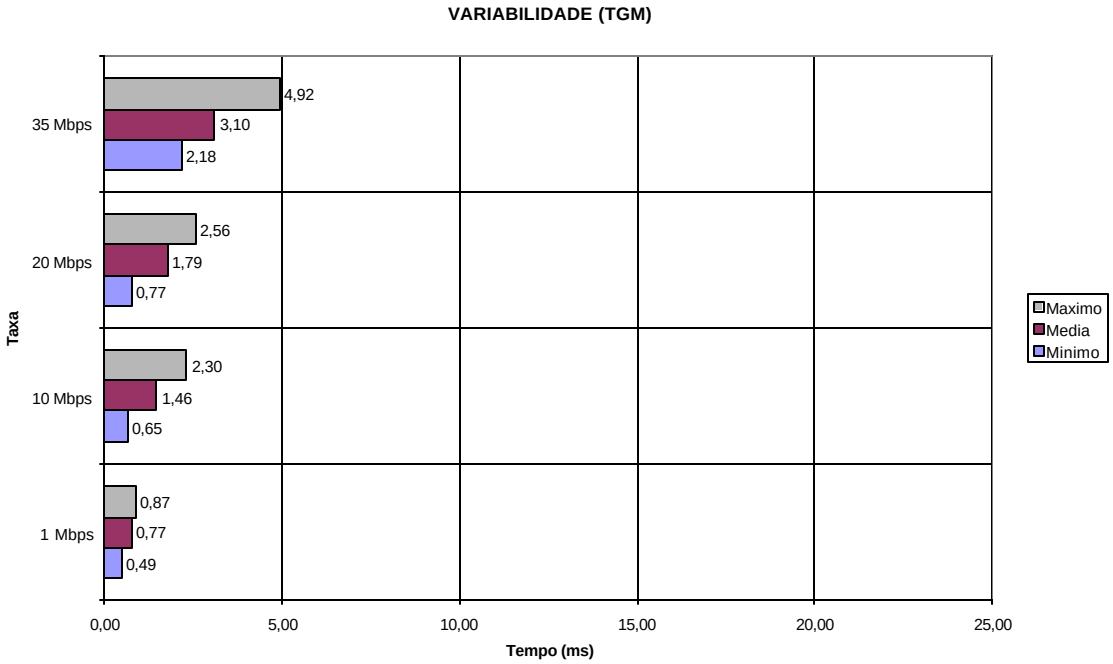


Figura 4.5 – Variabilidade dos Tempos de Round Trip no Aplicativo *TGM* para Diferentes Taxas de Transmissão.

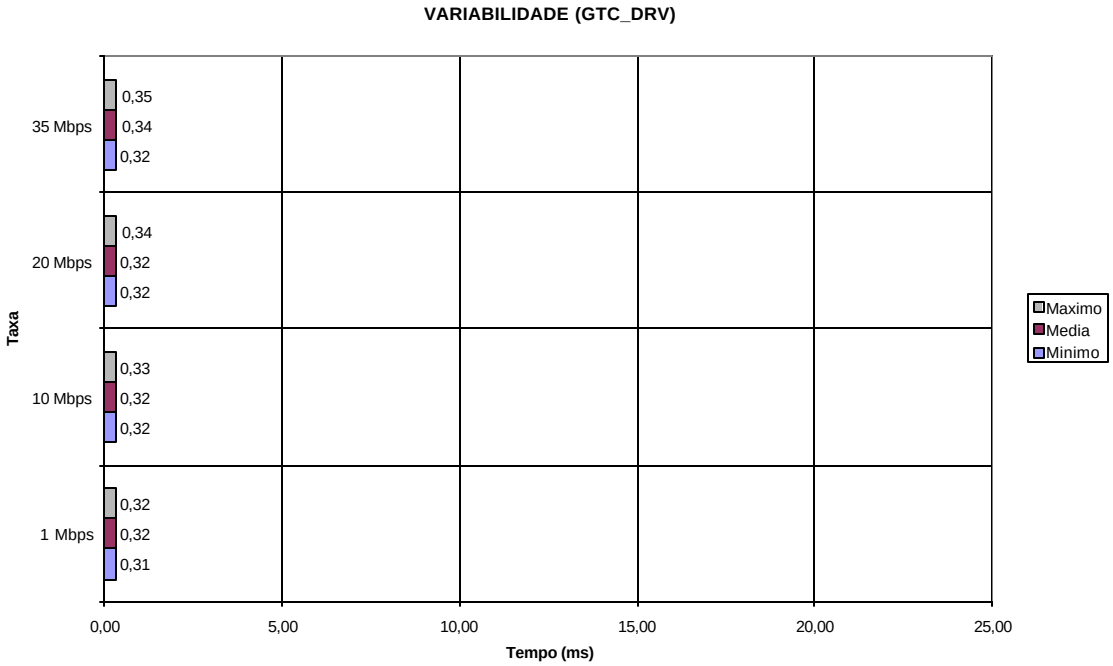


Figura 4.6 – Variabilidade dos Tempos de Round Trip no Aplicativo *GTC_DRV* para Diferentes Taxas de Transmissão.

Os resultados obtidos mostram que, quanto maior é a taxa de transmissão, maior é a variabilidade dos tempos medidos. Pode-se observar que o aplicativo gtc_drv possui uma certa independência da taxa de transmissão, confirmando a maior precisão de suas medidas com relação aos demais aplicativos. A variabilidade dos tempos medidos pelo gtc_drv está na ordem de alguns microsegundos, podendo ser considerada desprezível, enquanto nas demais aplicações esta variabilidade pode chegar a dezenas de milisegundos.

- **Segundo Experimento**

O segundo experimento mantém constante a taxa de transmissão em 10 Mbps e varia o tamanho dos pacotes entre 200 e 1450 octetos, tendo por objetivo verificar a influência dos diferentes tamanhos de pacotes nas medidas de round trip. Foram realizadas 10 rodadas de medidas, com 5000 pacotes em cada uma delas. Os gráficos a seguir apresentam os resultados obtidos.

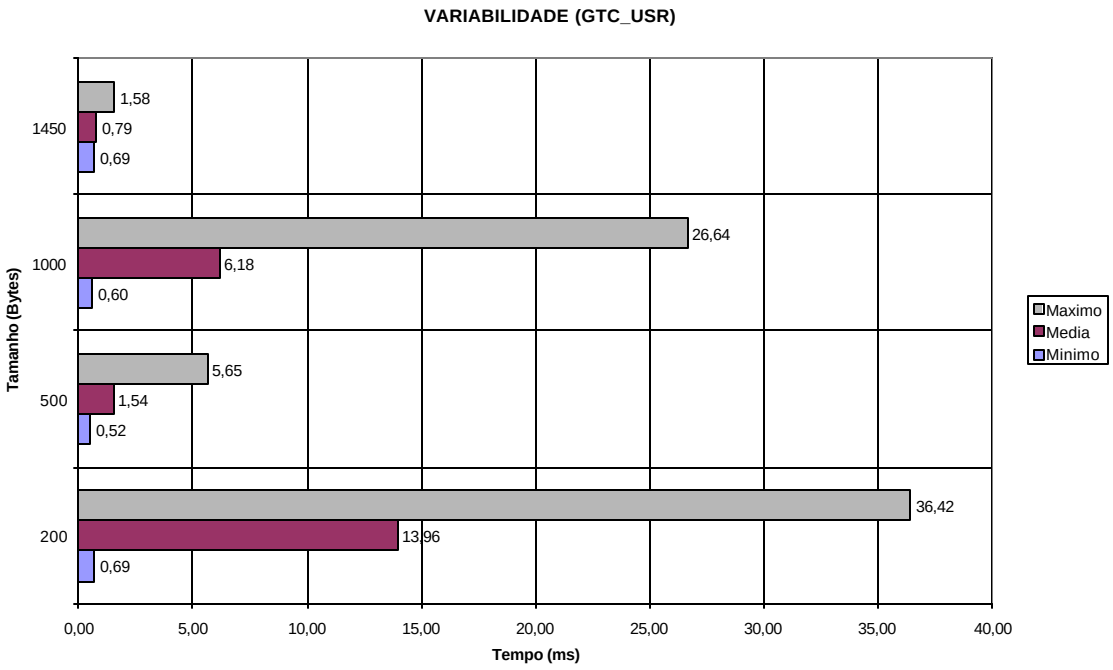


Figura 4.7 – Variabilidade dos Tempos de *Round Trip* no Aplicativo *GTC_USR para Diferentes Tamanhos de Pacotes*

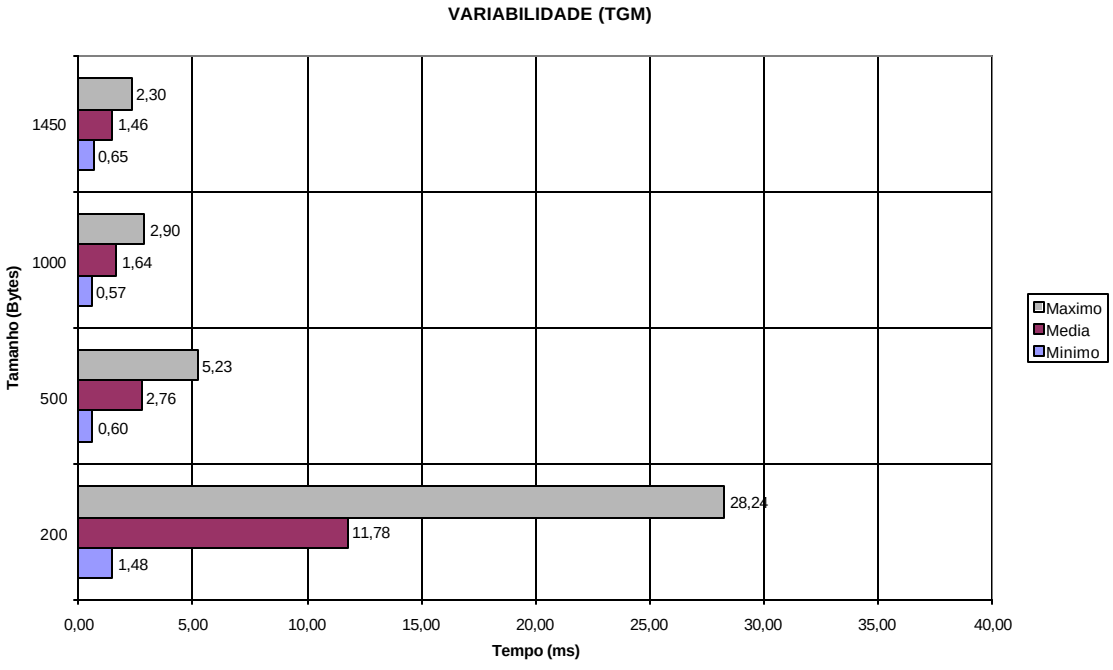


Figura 4.8 – Variabilidade dos Tempos de *Round Trip* no Aplicativo *TGM para Diferentes Tamanhos de Pacotes*

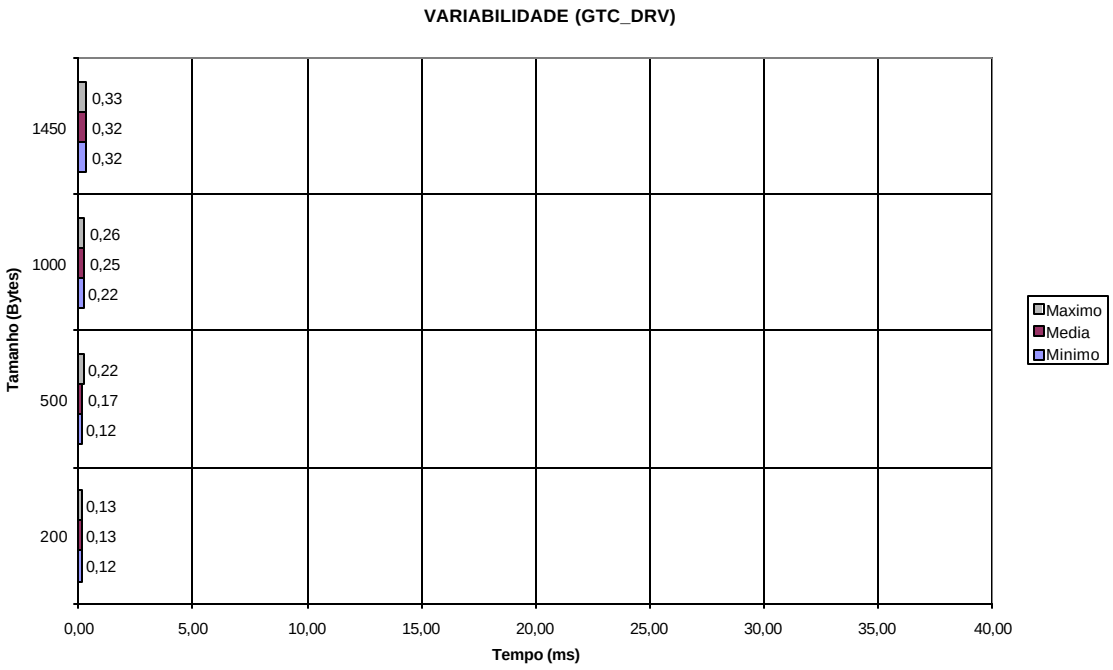


Figura 4.9 – Variabilidade dos Tempos de *Round Trip* no Aplicativo *GTC_DRV para Diferentes Tamanhos de Pacotes*

Os resultados obtidos mostram que, quanto menor o tamanho do pacote, maior a variabilidade da medição dos tempos, o que pode ser atribuído a um maior número solicitações ao S.O. por parte dos aplicativos. Mais uma vez pode-se constatar a independência entre o tamanho dos pacotes e as medidas realizadas pelo aplicativo gtc_drv, mostrando a sua maior precisão e confiabilidade com relação aos demais aplicativos.

4.3.4- Segunda Ferramenta

Apesar de extremamente eficiente e precisa, a ferramenta *gtc_drv* apresenta um inconveniente para os testes de validação do ambiente de serviços diferenciados: a necessidade de tráfego bidirecional para fazer as medidas. Como a reserva de recursos é unidirecional, os pacotes recebem um tratamento diferenciado apenas na direção do refletor. Em todo o caminho de volta, os pacotes são tratados segundo a disciplina FCFS, criando um tráfego competitivo que onera o processamento nos roteadores e obscurecendo assim a interpretação das medidas. Portanto, foram criadas duas novas

ferramentas que fazem as medições gerando tráfego unidirecional: uma geradora e outra receptora.

A ferramenta geradora gera um tráfego semelhante a *gtc_drv*, mas não espera tráfego de retorno. Ela chama-se *gtc_flood* e usa os mesmos parâmetros de geração da ferramenta *gtc_drv*. A ferramenta receptora é executada em um outro computador, semelhante à aplicação refletora. A diferença é que esta nova aplicação não envia de volta os pacotes recebidos, além de realizar ela própria as medidas.

Sempre que um pacote chega ao computador executando a aplicação receptora, uma amostra do tempo é feita no *driver*. Ao chegar o primeiro pacote, os contadores de tempo utilizado na medição são "zerados", tornando o momento da chegada do primeiro pacote a referência das medições realizadas. A fim de saber quando um pacote é o primeiro de uma seqüência, há um teste que compara se a diferença de tempo entre o pacote que acabou de chegar e o pacote anterior é maior que um segundo. Este tempo foi estabelecido, visto que em uma mesma seqüência de transmissão o tempo entre pacotes é no máximo alguns poucos microsegundos. Se o tempo entre pacotes for maior que um segundo, isto indica com certeza que o pacote que acabou de chegar faz parte de outro fluxo. Atualmente, o receptor mede apenas o tempo médio entre pacotes, fazendo amostragens parciais a cada 1000 pacotes.

4.4- Avaliação das Medidas

Os resultados aqui apresentados foram obtidos através do segundo conjunto de ferramentas. Vale ressaltar que a mesma precisão aferida para o primeiro conjunto de ferramentas (*gtc_drv* e o *refletor*) vale para o segundo conjunto (*gtc_flood* e o *receptor*).

A Figura 4.10 mostra o ambiente de teste.

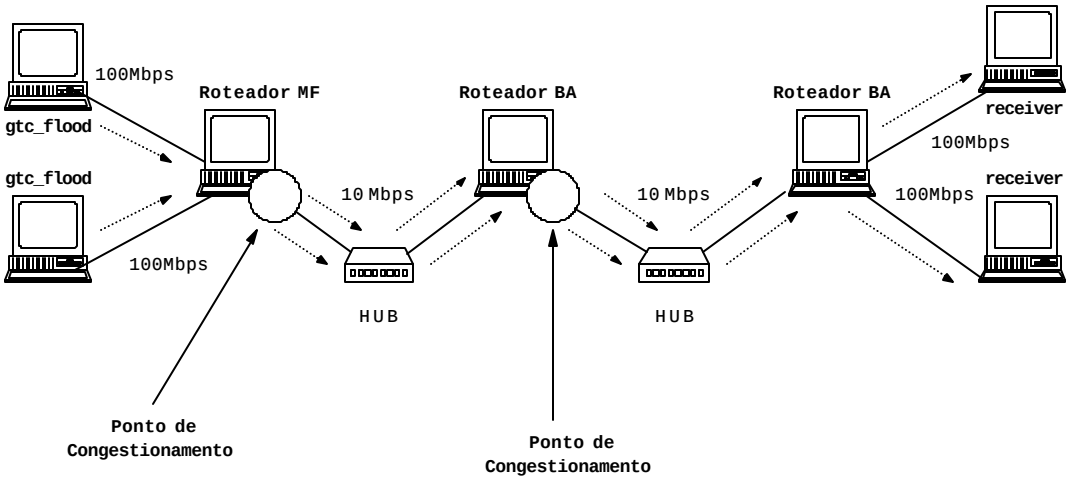


Figura 4.10- Ambiente de Validação

O ambiente de teste é formado por dois computadores gerando tráfego simultâneo por meio dos aplicativos *gtc_flood*. Um fluxo, denominado *fundamental*, será objeto de medição de desempenho e o outro, denominado *perturbador*, ocasiona apenas disputa de recursos nos roteadores. Os dois fluxos possuem taxa de transmissão de 5 Mbps com pacotes de 500 octetos. Há três computadores fazendo o roteamento: o primeiro deles faz o roteamento MF (*Multi-Field*), marcando apenas os pacotes do fluxo a ser medido; os demais roteadores fazem o roteamento BA (*Behavior-Aggregate*). Os dois outros computadores executam o aplicativo receptor, onde são realizadas as medidas.

A Figura 4.10 também mostra os pontos (interfaces de rede) onde ocorrem propositais congestionamentos. Variando a disciplina de escalonamento nestes pontos é possível comprovar se há ou não um tratamento diferenciado dos pacotes marcados. As disciplinas DRR e STFQ usam peso de 200 e um *quantum* de 128 bits. A fim de evitar a sobrecarga de processamento nos roteadores, optamos por desabilitar o módulo condicionador de tráfego, mantendo apenas a classificação e marcação dos pacotes.

4.5- Resultados

Foram realizados dois experimentos, consistindo cada um de 5 seqüências de medidas. No primeiro experimento, apenas o fluxo *fundamental* é utilizado, não havendo "perturbação" na rede; no segundo, ambos os fluxos disputam recursos. Cada

uma das 5 seqüências é composta por 10 rodadas de medidas, com 1000 pacotes de 500 octetos em cada uma. A primeira seqüência usou o roteamento normal do TROPIX com disciplinas FCFS nos pontos de congestionamento. A segunda seqüência usou os roteadores MF e BA, como mostrado na Figura 4.10, ainda com disciplina FCFS nos pontos de congestionamento. As demais seqüências utilizaram roteamentos MF e BA, variando as disciplinas de escalonamento nos pontos de congestionamento. Foram utilizadas respectivamente as disciplinas DRR, STFQ e PQ. O Gráfico da Figura 4.11 apresenta os resultados.

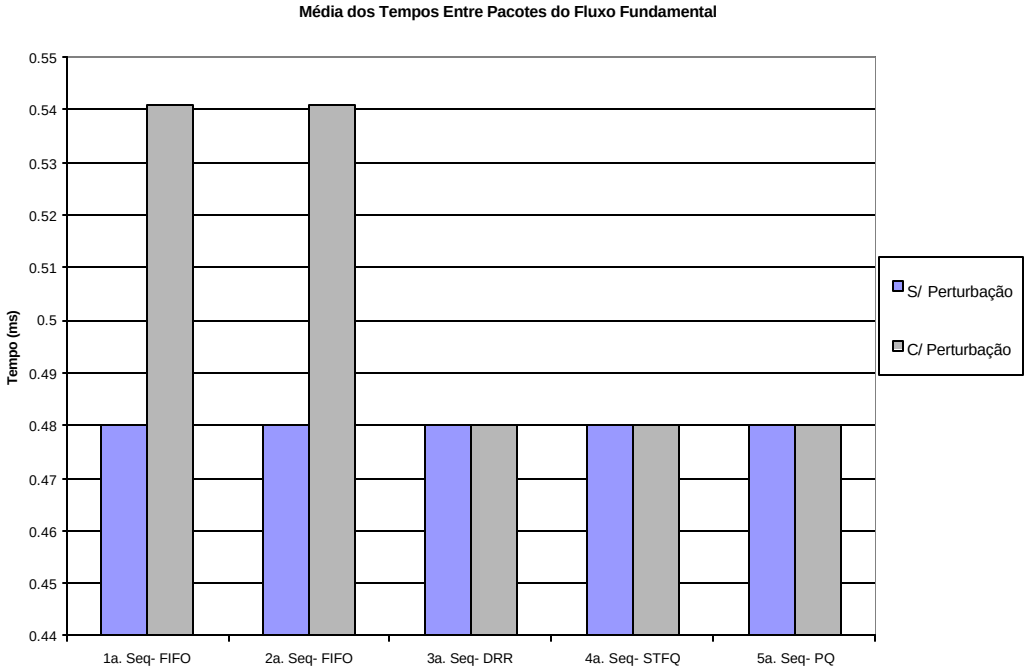


Figura 4.11- Gráfico das Médias dos Intervalos de Tempo entre Pacotes do Fluxo Fundamental

Como pode ser observado, quando as políticas de escalonamento DRR, STFQ e PQ são utilizadas nos pontos de congestionamento, a média do intervalo entre os pacotes mantém-se a mesma quando o fluxo *fundamental* é único ou quando ele disputa recursos com o fluxo *perturbador*. No entanto, quando a disciplina FCFS é utilizada, torna-se irrelevante o tipo de roteador (tipo de classificação). A taxa de geração utilizada (5 + 5 Mbps) para os fluxos tornou desprezível a sobrecarga adicional

decorrente da classificação MF, onde é necessária a verificação da existência da reserva para cada pacote que chega ao roteador, para possível marcação.

As disciplinas de escalonamento apresentaram comportamentos praticamente equivalentes nos experimentos realizados. Isto se deve à escolha do peso e do *quantum* das disciplinas DRR e STFQ: os valores 200 e 128, respectivamente, aproximam o comportamento destas disciplinas ao comportamento da disciplina PQ.

Os experimentos comprovam a eficácia das políticas de escalonamento no tratamento diferenciado dos pacotes do fluxo *fundamental*.

Um conjunto adicional de experimentos foi realizado substituindo os enlaces de 10 Mbps por enlaces de 100 Mbps. Nestes testes, não percebemos nenhum benefício na utilização das políticas implementadas: as taxas de 10 Mbps (5 + 5 Mbps) na entrada do roteador MF não criaram congestionamento no enlace de saída (100 Mbps). Para criar pontos de congestionamento nas saídas dos roteadores, aumentamos as taxas de geração dos aplicativos, até um limite de aproximadamente 30 + 30 Mbps, o máximo tolerado pelos roteadores. Além disso, para as altas taxas, os tamanhos dos pacotes tiveram que ser aumentados para 1450 octetos. Com o aumento das taxas de geração e dos tamanhos dos pacotes, não foi observado qualquer ganho com o uso das políticas, apresentando todas resultados semelhantes. Isto se deve à sobrecarga de processamento nos roteadores.

Tais resultados comprovam que a classificação MF deve ser implementada apenas nos roteadores das extremidades das redes de serviços diferenciados, como os roteadores folha e de borda das redes clientes de um Domínio DS, visto que nestes roteadores as taxas de transmissão são menores do que as taxas de transmissão dos *backbones* da Internet (ISP).

CONCLUSÕES

O presente trabalho apresentou uma proposta para o tráfego *multicast* em redes que oferecem serviços diferenciados, visando melhorar o desempenho das aplicações multimídia e multiponto. Esta arquitetura complementa e apresenta algumas alternativas à proposta existente na literatura [BW00].

Baseada na verificação antecipada da disponibilidade de recursos e a sua reserva nos ramos da árvore de distribuição de dados de uma sessão *multicast*, nossa arquitetura é composta por um sistema de reserva de recursos de rede e de roteadores que implementam serviços diferenciados. Apesar das recomendações propostas na RFC 2475, a arquitetura utiliza o mesmo valor de *codepoint* e SLA (*Service Level Agreement*) para os fluxos *unicast* e *multicast*.

O sistema de reservas é baseado em Servidores de Reserva de Recursos (SRR) que se comunicam a fim de gerenciar a reserva de recursos nos domínios de serviços diferenciados para cada sessão *multicast*. Além disso, os SRRs são os responsáveis pela configuração dos roteadores destes domínios, que implementam o serviço "*multicast Premium*" através do PHB *Expedited Forwarding*.

O protocolo *multicast* proposto para a arquitetura foi o PIM-SSM. Este protocolo, ainda em fase de especificação pelo IETF, cria uma árvore de distribuição baseada na fonte de um grupo *multicast*, ao invés de uma árvore compartilhada. Tal abordagem apresenta-se mais adequada aos propósitos da nossa arquitetura, onde a fonte de dados é conhecida *a priori* pelas aplicações.

A comunicação entre os elementos da arquitetura ocorre por intermédio de protocolos proprietários.

Os módulos classificador, condicionador e escalonador dos roteadores foram codificados em linguagem ANSI-C. Esta implementação contou com a disponibilidade do sistema operacional TROPIX, cujo núcleo vem sendo desenvolvido no NCE/UFRJ. As seguintes disciplinas de escalonamento foram desenvolvidas: *Priority Queueing*, *Deficit Round-Robin* e *Start-Time Fair Queueing*. O condicionamento do tráfego utiliza o algoritmo do "Balde de Fichas" ("*Token Bucket*").

Os módulos implementados dos Servidores de Reserva de Recursos foram desenvolvidos em ambiente Windows 98, utilizando o Delphi versão 4 e tabelas em Paradox.

Para a obtenção de medidas que validassem a eficácia das disciplinas de escalonamento, desenvolvemos no TROPIX aplicativos geradores de tráfego. A abordagem convencional de geração realiza amostragens de tempo no contexto da aplicação geradora; desta maneira, os atrasos decorrentes do processamento dos pacotes pelo sistema operacional e pelo aplicativo influenciam os tempos medidos, prejudicando sobremaneira a interpretação dos resultados. Em nossa ferramenta de geração, os tempos são amostrados no *driver* da interface de rede por onde os pacotes são transmitidos/recebidos, eliminando completamente a parcela de tempo decorrente da interferência do sistema operacional e da aplicação geradora. Os aplicativos implementados podem também ser utilizados como "perturbadores", introduzindo diferentes níveis de tráfego em uma rede sendo estudada. As comparações realizadas permitem constatar a precisão das medidas obtidas pelas ferramentas que desenvolvemos, em experimentos onde taxa de transmissão e tamanhos de pacotes foram exaustivamente variados.

Finalmente, criamos um ambiente de testes onde os experimentos realizados comprovaram a correção e eficácia das implementações realizadas para taxas de transmissão de até 10 Mbps. Para taxas mais elevadas, não foi observado ganho algum com as diferente disciplinas de escalonamento, devido à sobrecarga de processamento nos roteadores. Notamos igualmente uma sobrecarga de processamento quando o módulo de condicionamento estava ativado nos roteadores. Tais resultados comprovam que a classificação MF deve ser implementada apenas nos roteadores das extremidades das redes de serviços diferenciados, como os roteadores folha e de borda das redes clientes de um Domínio DS, visto que nestes roteadores as taxas de transmissão são menores do que as taxas de transmissão dos *backbones* da Internet (ISP).

O presente trabalho tem prosseguimento natural com a implementação dos módulos da arquitetura relativos ao roteamento *multicast* (PIM-SSM e IGMP versão 3), bem como as suas configurações. Como estes protocolos estão em fase de especificações pelo IETF, deixamos como trabalho futuro a implementação deste módulos.

É igualmente indispensável a implementação de procedimentos de segurança na configuração dos roteadores, evitando atender solicitações espúrias. Por fim, recomendamos a utilização de protocolos padronizados para a configuração e comunicação dos elementos da arquitetura, como o RSVP ou similares.

REFERÊNCIAS BIBLIOGRÁFICAS

[APO00] Azevedo, J., Pirmez, L., Vernet, O., et alii. **Quality of Services for Internet Multicast Application Traffic**. Anais do CIC 2000, Las Vegas, Nevada, USA, pp.251 a 254, junho, 2000.

[B95] Baker, F. **Requirements for IP Version 4 Routers.** Disponível na INTERNET via url: http://www.ietf.org. RFC 1812, junho, 1995.

[BBC98] Blake, S., Black, D., Carlson, M., et alii. **An Architecture for Differentiated Services.** Disponível na INTERNET via url: http://www.ietf.org. RFC 2475, Dezembro, 1998.

[BCC98] Braden, B., Clark, D., Crowcroft, J., et alii. **Recommendation on Queue Management and Congestion Avoidance in the Internet.** Disponível na INTERNET via url: http://www.ietf.org. RFC 2309, abril, 1998.

[BCS94] Braden, R., Clark, D., Shenker, S. **Integrated Services in the Internet Architecture: an Overview.** Disponível na INTERNET via url: http://www.ietf.org. RFC 1633, junho, 1994.

[BLB98] Berson, S., Lindell, R., Braden, R. **An Architecture for Advance Reservations in the Internet**. Junho, 1998. Disponível na INTERNET via url: http://www.iam.unibe.ch/~balmer/paperindex.html#RSVP_IETF.

[BV98] Berson, S., Vicent, S. **Aggregation of Internet Integrated Services State.** Internet Draft <draft-berson-rsvp-aggregation-00.txt, Agosto 1998. Disponível na INTERNET via url: http://www.iam.unibe.ch/~balmer/paperindex.html #RSVP_IETF.

[BW00] Bless, R., Wehrle, K. **IP Multicast in Differentiated Services Networks.** draft-bless-diffserv-multicast-01.txt, novembro, 2000. Disponível na INTERNET via url: http://search.ietf.org/internet-drafts/draft-bless-diffserv-multicast-01.txt.

[BW99a] Bless, R., Wehrle, K. **A Lower Than Best-Effort Per-Hop Behavior**. Internet-Draft –draft-bless-diffserv-lbe-phb-00.txt, setembro, 1999. Disponível na INTERNET via url: http://www.iam.unibe.ch/~balmer/paperindex.html #DS_IETF.

[BZB97] Braden, R., Zhang, L., Berson, S., et alii. **Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification**. RFC 2705, setembro, 1997. Disponível na INTERNET via url: http://www.ietf.org.

[CDF01] Cain, B., Deering, S., Fenner, B., et alii. **Internet Group Management Protocol, Version 3** . Internet Draft <draft-ietf-idmr-igmp-v3-07.txt> , março, 2001. Disponível na INTERNET via url: http://www.ietf.org.

[D89] Deering, S. **Host Extensions for IP Multicasting**. RFC 1112, agosto, 1989. Disponível na INTERNET via url: http://www.ietf.org.

[DEF98] Deering, S., Estrin, D., Farinacci, D., et. Alii. **Protocol Independent Multicast Version 2, Dense Mode Specification.** Internet Draft <draft-ietf-pim-v2-dm-00.txt>, agosto, 1998. Disponível na INTERNET via url: http://brutus.snu.ac.kr/~schoi/references/draft-ietf-pim-v2-dm-00.txt

[DH97] Deering, S., Hinden, R.. **Internet Protocol, Version 6 (IPv6) Specification.** RFC 2460, dezembro, 1998. Disponível na INTERNET via url: http://www.ietf.org.

[EFH98] Estrin, D., Farinacci, D, Helmy, A., et alii. **Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification**. RFC 2362, junho, 1998. Disponível na INTERNET via url: http://www.ietf.org.

[F97] Fenner, W. **Internet Group Management Protocol, Version 2** RFC 2236, novembro, 1997. Disponível na INTERNET via url: http://www.ietf.org.

[FRM00] Farinacci, D., Rekhter, Y., Meyer, D., et. alii. **Multicast Source Discovery Protocol (MSDP)**. Internet Draft <draft-ietf-msdp-spec-02.txt>, janeiro, 2000. Disponível na INTERNET via url: http://www.belnet.be/technical/pages/multicast/draft-ietf-msdp-spec-02.txt.

[HBW99] Heinanen, J., Baker, F., Weiss, W., et alii. **Assured Fowarding PHB Group.** RFC 2597, junho, 1999. Disponível na INTERNET via url: http://www.ietf.org.

[HC00] Holbrook, H., Cain, B. **Source-Specific Multicast for IP.** Internet Draft <draft-holbrook-ssm-arch-01.txt>, novembro, 2000. Disponível na INTERNET via url: ftp://ftpeng.cisco.com/ipmulticast/drafts.

[I01] Internet Multicast Addresses. Disponível na INTERNET na url: http://www.isi.edu/in-notes/iana/assignments/multicast-addresses. Consultado em 2001.

[ISI81] Information Sciences Institute. **Internet Protocol.** RFC 791, setembro, 1981. Disponível na INTERNET via url: http://www.ietf.org.

[J94] Moy, J. **Multicast Extesions to OSPF**. RFC 1584, março, 1994.Disponível na INTERNET via url: http://www.ietf.org.

[J88] Jacobson, V.. **Congestion Avoidance an Control.** ACM SIGCOMM'88, agosto, 1988.

[JNP99] Jacobson, V., Nichols, K., Poduri, K. **An Expedited Fowarding PHB.** RFC 2598, junho, 1999. Disponível na INTERNET via url: http://www.ietf.org.

[LA00] Leocádio, M., Aguiar, P. H. **Uma Ferramenta para Geração de Tráfego e Medição em Ambiente de Alto Desempenho.** Anais do 18º SBRC, Belo Horizonte, MG, pp. 321 a 336, maio de 2000.

[LR91] Lougheed, K., Rekhter, Y. **A Border Gateway Protocol 3 (BGP-3)** RFC 1267, outubro, 1991. Disponível na INTERNET via url: http://www.ietf.org.

[K98] KESHAV, S. **An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network**. Addison-Wesley, 1998. cap. 9, p 209-264.

[M98] Moy, J.. **OSPF Version 2**. RFC 2328, abril, 1998. Disponível na INTERNET via url: http://www.ietf.org.

[M98a] Meyer, D. **Administratively Scoped IP Multicast.** RFC 2365, julho, 1998. Disponível na INTERNET via url: http://www.ietf.org.

[MBK96] McKusick, M. K., Bostic, K., Karels, M. J., et alii. **The Design and Implementation of the 4.4 BSD Operating System.** Addison-Wesley, 1996. cap. 11, p 369-374.

[ML00] Meyer, D., Lothberg, P. **GLOP Addressing in 233/8** RFC 2770, fevereiro, 2000. Disponível na INTERNET via url: http://www.ietf.org.

[N01] **Página oficial do Netperf**. Disponível na INTERNET na url: http://www.netperf.org. Consultado em 2001.

[NBB98] Nichols, K., Blake, S., Baker, F., et alii. **Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.** RFC 2474, dezembro, 1998. Disponível na INTERNET via url: http://www.ietf.org.

[NJZ99] Nichols, K., Jacobson, V., Zhang, L. **A Two-bit Differentiated Services Architecture for the Internet**. RFC 2638, julho, 1999. Disponível na INTERNET via url: http://www.ietf.org. Arquivo consultado em 2001.

[T01] Página oficial do TROPIX. Disponível na INTERNET na url: http://www.tropix.nce.ufrj.br/. Consultado em 2001.

[WDP88] Waitzman, D., Deering, S., Partridge, C. **Distance Vector Multicast Routing Protocol.** RFC 1075, novembro, 1988. Disponível na INTERNET via url: http://www.ietf.org.