UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

PROJETO DE DISSERTAÇÃO DE MESTRADO

TÍTULO

Uma Arquitetura para Tráfegos multicast no Modelo de Serviços Diferenciados

ALUNO: Jefferson Manhães de Azevedo
ORIENTADORES: Luci Pirmez, Luiz Fernando Rust da Costa Carmo e Oswaldo Vernet de Souza Pires

RESUMO

Este projeto de tese visa a implementação do tráfego *multicast* no Modelo de Serviços Diferenciados, em um ambiente operacional baseado no TROPIX, que será útil para a criação de uma infra-estrutura de Internet em projetos de educação à distância e tele-medicina. Através desta arquitetura, diferentes níveis de QoS podem ser oferecidos às aplicações multimídia com tráfego *multicast*, como aplicativos de vídeo/áudio conferências e de recuperação de documentos multimídia. A fim de permitir tal tráfego, é necessária a reserva de recursos nos roteadores, que é realizada por intermédio de Agentes e Servidores Alocadores de Largura de Banda. Além disso, políticas de escalonamento e de descarte de pacotes são implementadas nos roteadores objetivando o tratamento diferenciado dos fluxos de dados. O tráfego *multicast* será implementado através de dois tipos de protocolos: um interdomínio e outro intradomínio. O protocolo intradomínio é o BGMP (Border Gateway Multicast Protocol) e o intradomínio é o DVMRP. O Sistema Operacional dos roteadores é o TROPIX, sendo os seus módulos desenvolvidos em linguagem ANSI-C. Já os módulos dos Agentes e Servidores Alocadores de Largura de Banda foram desenvolvidos na Linguagem DELPHI-4, utilizando tabelas em PARADOX.

Além disso, analisamos a implementação de um gerador de tráfego que visa obter medidas mais precisas do atraso sofrido pelos pacotes em seu percurso por uma rede de computadores. Usualmente, os geradores de tráfego apresentam indesejável variabilidade nessas medidas, quando as taxas de transmissão são altas e os tamanhos de pacotes são pequenos. O gerador de tráfego implementado possui uma versão para geração de tamanhos de pacotes constante e outra para tamanhos de pacotes aleatórios, o que melhor espelha o tráfego em uma rede real.

Keywords: Differentiated Services, Multicast, Quality of Service and UNIX Routers.

ÍNDICE

RESUMO 1
ÍNDICE 2
INTRODUÇÃO 5
Capítulo 1 7
QUALIDADE DE SERVIÇO NA INTERNET 7
1- INTRODUÇÃO 7
2- MODELO DE SERVIÇOS INTEGRADOS 7
2.1- Serviço de Tempo-Real 8
2.1.1- Aplicações de Tempo-Real 8
2.1.2- Aplicações Elásticas 9
2.2- Compartilhamento de Recursos 10
2.3- Serviços Adicionais 10
2.4- Mecanismos de Controle de Tráfego 10
2.5- Mecanismo do Protocolo de Configuração 11
3- PROTOCOLO DE CONFIGURAÇÃO DE RESERVA (RSVP) 12
3.1- Especificação de Fluxos (*Flowspecs*) e Especificação de Filtros (*Filter Specs*) 13
3.1.1- Estilos de Reserva 13
3.1.2- Iniciação pelo Receptor 13
3.1.3- *Soft State* 14
4- AGREGAÇÃO DE ESTADO NO RSVP 14
4.1- Módulos dos Roteadores Intserv 14
4.2- Redução da Quantidade de Estados no Intserv 15
4.2.1- Estado de Escalonamento 15
4.2.2- Estado de Classificação 15
4.2.3- Estado do Protocolo de Configuração 15
4.3- Arquitetura de Agregação Básica 16
4.3.1- Estado do Protocolo de Configuração *Multicast* 17
4.3.2- Policiamento no Interior 17
4.4- Extensões e Operação do RSVP 17
5- MODELO DE SERVIÇOS DIFERENCIADOS (DIFFSERV) 18
5.1- Conceitos Fundamentais do Diffserv 19
5.2- Classificação, Condicionamento e Escalonamento de Tráfego 20
5.2.1- Classificadores 20
5.2.2- Condicionador de Tráfego 21
5.2.3- Escalonador de Tráfego 22
5.3- Serviços Oferecidos pelo Diffserv 22
5.3.1- Classe de Precedência 22
5.3.2- Grupo de PHB AF (*Assured Forwarding*) 23
5.3.3- Grupo de PHB EF (*Expedited Forwarding*) 24
6- TRÁFEGO *MULTICAST* 24
6.1- IP *Multicast* 24
6.1.1- Endereço IP *Multicast* 25
6.1.2- Mapeamento de Endereço IP *Multicast* no *Ethernet* 25
6.1.3- *Internet Group Management Protocol* (IGMP) 25
6.1.4- Árvores de Distribuição *Multicast* 26
6.2- Roteamento *Multicast* 27
6.2.1- *Reverse Path Forwarding* (RPF) 27
6.2.2- DVMRP 27
6.2.3- Extensão *Multicast* para o OSPF (MOSPF) 28
6.2.4- *Protocol Independent Multicast* (PIM) 29
6.2.5- *Multiprotocol Border Gateway Protocol* (MBGP) 30
6.2.6- *Border Gateway Multicast Protocol* (BGMP) 30

6.2.7- *Multicast Source Discovery Protocol* (MSDP) 31
6.2.8- PIM-*Source Specific Multicast* (PIM-SSM) 32
6.3- Tráfego *Multicast* no Diffserv 33
6.3.1- Considerações sobre o Tráfego *Multicast* Apresentadas na RFC 2475 34
6.3.2- Proposta de Bless e Wehrle 34
7- POLÍTICAS DE ESCALONAMENTO 35
7.1- Lei da Conservação 35
7.2- Requisitos para as Disciplinas de Escalonamento 35
7.2.1- Facilidade de Implementação 35
7.2.2- Justiça 36
7.2.3- Proteção 36
7.2.4- Limites de Desempenho 36
7.2.5- Facilidade e Eficiência no Controle de Admissão 36
7.3- Projetos de Disciplinas de Escalonamento 36
7.3.1- Níveis de Prioridade 37
7.3.2- Disciplinas de Trabalho Conservativo versus Trabalho não Conservativo 37
7.4- Disciplinas de Enfileiramento 37
7.4.1- Compartilhamento de Processador Genérico (GPS) 37
7.4.2- *Weighted Round-Robin* (WRR) 38
7.4.3- *Deficit Round-Robin* (DRR) 38
7.4.4- *Weighted Fair Queuing* (WFQ) 38
7.4.5- Variantes da WFQ 40
7.4.6- *Class-Base Queuing* (CBQ) 40
8- POLÍTICAS DE DESCARTE DE PACOTES 41
8.1- Classificação das Políticas de Descarte 41
8.1.1- Nível de Agregação 41
8.1.2- Prioridades de Descarte 41
8.1.3- Descarte Antecipado ou em Sobrecarga 42
8.1.4- Posição de Descarte 42
8.2- RED 43
Capítulo 2 44
ARQUITETURA PARA TRÁFEGOS *MULTICAST* NO DIFFSERV 44
1- INTRODUÇÃO 44
2- DESCRIÇÃO DA ARQUITETURA 44
3- PROTOCOLOS *MULTICAST* 48
4- PROCEDIMENTOS DE RESERVA 49
5- ESTRUTURA FUNCIONAL DO SRR 51
6- ESTRUTURA FUNCIONAL DOS ROTEADORES 53
7- ESQUEMA DA INTEROPERABILIDADE ENTRE OS COMPONENTES DA ARQUITETURA 54
Capítulo 3 55
IMPLEMENTAÇÃO 55
1- INTRODUÇÃO 55
2- IMPLEMENTAÇÃO DO AMBIENTE DE SERVIÇOS DIFERENCIADOS 55
2.1 - Estruturas de Dados do Núcleo do TROPIX para Rede 55
2.1.1- Estrutura de um Pacote de Dados no TROPIX 55
2.1.2- Estruturas Utilizadas pelos Dispositivos de Rede 56
2.2- Módulo Escalonador 56
2.2.1- Reestruturação das Chamada às Funções de Enfileiramento e Desinfileiramento 57
2.2.2- Estruturas de Dados para as Disciplinas de Escalonamento 58
2.2.3- Implementação das Disciplinas de Filas 59
2.3- Classificador dos Pacotes 60
2.4- Condicionador de Tráfego 61
2.5- Configuração do Ambiente 61
2.5.1- Funções de Configuração 61

2.5.2- Aplicativos de Configuração 62
3- IMPLEMENTAÇÃO DO GERADOR DE TRÁFEGO 63
3.1- Ambiente para Geração de Tráfego 63
3.1.1- Módulo Transmissor 64
3.1.2- Módulo Receptor 64
3.1.3- Refletor 64
3.2- A Implementação 64
3.2.1- Protocolo de Transporte de Dados Utilizado pelos Geradores de Tráfego 65
3.2.2- Gerando Pacotes de Tamanhos Variados 65
3.3- Contabilizando o Atraso dos Pacotes 66
3.4- Uso dos Geradores 67
3.5- Resultados 67
3.5.1- Ambiente de Teste 68
3.5.2- Medidas Comparativos 68
4. Sistema de Reserva de Recursos 72
CONCLUSÃO 74
REFERÊNCIAS BIBLIOGRÁFICAS 75

INTRODUÇÃO

Este projeto de dissertação de mestrado tem por objetivo propor uma arquitetura para o uso do tráfego multicast no Modelo de Serviços Diferenciados, em um ambiente operacional baseado no TROPIX, sendo útil na criação de infra-estruturas de Internet para projetos de educação à distância e tele-medicina. Tais projetos desejam fazer uso intensivo da Internet utilizando suas aplicações multimídia, como as de tele-conferência de vídeo/áudio e as de recuperação de documentos multimídia. Mas o tráfego gerado por estas aplicações necessitam de níveis de QoS ("Quality of Service" - qualidade de serviço) diferentes do QoS oferecido pelo protocolo IP, através do seu modelo de serviço de entrega de pacotes pelo melhor-esforço (*best-effort*). A fim de propiciar diferentes níveis de QoS no protocolo IP, que é a base de toda a infra-estrutura da Internet, o IETF propõe o modelo de Serviços Integrados (Intserv) [01] e o modelo de Serviços Diferenciados (Diffserv) [02].

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. Também não oferece garantias quanto ao atraso ou a variação do atraso (*jitter*) desta entrega. Além disso, no modelo *best-effort*, 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 tráfegos dos aplicativos multimídia. Estes aplicativos são muito sensíveis ao atraso e ao *jitter* sofrido por seus pacotes de dados quando transportados em uma rede. Em redes onde a carga de 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 melhorar tal desempenho, no início de 1986, Van Jacobson desenvolveu um mecanismo [03] para evitar congestionamento, utilizado atualmente pelo TCP. Este mecanismo torna o TCP sensível ao congestionamento. Assim que a fonte de dados usando o TCP percebe que está havendo perda de pacotes, ela diminui a sua taxa de transmissão e, aos poucos, torna a elevá-la. É justamente este mecanismo que não permite que a Internet atual entre em um colapso total.

Apesar de importante, este mecanismo do TCP, baseado nos nós de extremidade da rede, não é suficiente para evitar o congestionamento na Internet. É necessário também, como complemento, políticas de escalonamento e descarte de pacotes nos nós internos às redes (roteadores), que também é implementado neste trabalho. Estas políticas são amplamente utilizadas tanto no modelo Intserv quanto no modelo Diffserv. As políticas de escalonamento são responsáveis pela escolha do próximo pacote a ser servido, sendo obtidas através de diferentes disciplinas de filas. Neste projeto foram implementadas três novas disciplinas de filas no TROPIX, além da FIFO já existente: a *Deficit Round Robin* (DRR), a *Priority Queuing* (PQ) e a *Start Time Fair Queuing* (STFQ). Esta última é uma variação a I (WFQ). A políticas de descarte de pacotes são as responsáveis pelo gerenciamento do espaço disponível nas filas. A política de descarte implementada neste projeto foi a *Tail Drop*, comumente implementada nos roteadores convencionais, pois o protocolo UDP utilizado no IP *multicast* não é sensível ao descarte.

Os modelos Intserv e Diffserv não visam substituir o modelo de serviços da Internet, mas acrescentar novos serviços e, consequentemente, diferentes níveis de QoS ao protocolo IP. No modelo Intserv a garantia de QoS é dada a cada fluxo individualmente. 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. Vale ressaltar que existem propostas de agregação de fluxos para o modelo Intserv [04].

Diversas propostas de implementação, com alguns experimentos e testes já em andamento, vem sendo desenvolvidas para o tráfego *unicast* no modelo Diffserv. Por outro lado, não há nenhuma proposta para o tráfego *multicast*. Apenas algumas considerações são feitas em [02], como será visto mais adiante. O intuito deste projeto de dissertação de mestrado é contribuir para as discussões e busca de soluções para este tipo de tráfego no modelo Diffserv.

Na arquitetura proposta, o tratamento diferenciado do tráfego *multicast* é obtido através da reserva de recursos nos roteadores, realizada por intermédio de Agentes e Servidores Alocadores de Largura de Banda. Os protocolos *multicast* utilizados são o BGMP, protocolo interdomínio, e o DVMRP, protocolo intradomínio.

Esta arquitetura baseia-se em propostas encontradas em diversos "Internet-drafts" referentes aos modelos Diffserv e Intserv. Dois deles merecem destaque. O primeiro está relacionado aos “Bandwidth Brokers” (corretores de bandas) [05]. O segundo descreve a arquitetura e o funcionamento de um Servidor de Reserva Antecipada de recursos para o modelo Intserv [06].

Capítulo 1

QUALIDADE DE SERVIÇO NA INTERNET

1- INTRODUÇÃO

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*. O modelo *best-effort* não oferece nenhum tipo de garantia quanto à entrega dos pacotes de dados ao destino. Ele também não oferece garantias 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 (neste trabalho o termo fluxo de dados refere-se ao conjunto de pacotes oriundos de uma mesma aplicação ou usuário). Estas características do bem sucedido protocolo IP simplificam os aspectos referentes à infra-estrutura de transporte de dados na Internet, mas tornam-se extremamente prejudiciais aos fluxos de dados 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 com baixa carga de utilização, o desempenho do protocolo IP no transporte de fluxos de dados multimídia é bastante satisfatório. O que 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 [03] 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 do interior 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 a 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 descreve o modelo Intserv. A segunda faz a descrição do protocolo RSVP ("Resource ReSerVation Protocol"), um protocolo de reserva de recursos utilizado para a configuração dos serviços no modelo Intserv. A terceira seção descreve uma proposta de agregação de fluxos para o modelo Intserv. A quarta seção descreve o modelo Diffserv. A quinta seção apresenta conceitos sobre roteamento *multicast* e algumas propostas para a implementação do 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 tanto pelo modelo Intserv quanto Diffserv.

2- MODELO DE SERVIÇOS INTEGRADOS

Atualmente, a Internet é inadequada ao transporte de fluxos multimídia. Devido ao alto índice de congestionamento nos roteadores, os pacotes sofrem grandes atrasos de enfileiramento, causando um atraso variável na entrega dos pacotes, além de uma alta taxa de descarte. Assim, torna-se necessária uma mudança na infra-estrutura da Internet para que haja algum tipo de controle sobre o atraso dos pacotes fim-a-fim, tanto para tráfegos *unicast* quanto *multicast*.

Um outro limite da infra-estrutura atual da Internet é a de não permitir que os administradores de rede controlem o compartilhamento do tráfego nos enlaces (*links*). Não é possível fazer a divisão de uso da largura de banda do enlace entre os diversos tipos de tráfegos, que são oriundos de diferentes aplicações e usuários. Este controle permite a divisão dos tráfegos em classes administrativas, disponibilizando as bandas não utilizadas.

O modelo Intserv visa atender a estas duas demandas. Ele é proposto pelo IETF em [01], sendo uma expansão do atual modelo de serviços da Internet. A proposta de um novo modelo de serviços está relacionada a criação de um conjunto de serviços essenciais baseados no tempo de entrega dos pacotes aos seus destinos. Este modelo de serviços consiste em um conjunto de compromissos da rede, que se compromete na entrega de algum tipo de serviço, em resposta a uma solicitação de serviço. Este compromisso pode ser categorizado pela entidade para quem ele foi feito, tanto para um fluxo individual, quanto para um grupo de fluxos. Para fluxos individuais, os compromissos de serviço buscam oferecer uma performance razoável a uma aplicação e está relacionado ao QoS dado a este fluxo. Já para entidades coletivas, estes serviços são direcionados ao compartilhamento de recursos, onde há uma agregação de recursos disponibilizados para várias entidades.

O modelo Intserv oferece três tipos de serviço de entrega: *best-effort*, tempo-real e controle do compartilhamento de enlaces. Estes dois últimos são descritos a seguir.

2.1- Serviço de Tempo-Real

A essência dos serviços em tempo-real está na garantia de obtenção de recursos feita previamente através de uma reserva. O usuário deve ser capaz de obter um serviço cuja qualidade seja suficientemente previsível, a fim de que sua aplicação possa operar em um modo aceitável durante todo o tempo de sua execução. Para isso, os roteadores têm que oferecer garantias de QoS aos fluxos de dados.

A implementação da reserva nos roteadores é feita por um mecanismo chamado controle de tráfego (*traffic control*). Este mecanismo inclui um módulo classificador de pacotes (*packet classifier*), um módulo de controle de admissão (*admission control*) e um módulo escalonador de pacotes (*packet scheduler*). O classificador de pacotes determina a classe de serviço a que um determinado pacote pertence. O módulo de controle de admissão é responsável pela admissão de uma nova reserva, baseado nos recursos disponíveis do roteador. O módulo escalonador de pacotes, existente em cada interface de saída dos roteadores, é o responsável por dar o tratamento adequado aos pacotes que são enviados, viabilizando o QoS acordado.

A configuração da reserva é realizada em cada roteador que se encontra no percurso do fluxo de dados na rede. Para isso, é necessário um protocolo para a configuração e manutenção desta reserva. Um dos protocolos usados é o *Resource Reservation Protocol* (RSVP), descrito na seção 3 deste capítulo.

As garantias de QoS no modelo Intserv estão quase que exclusivamente relacionadas ao tempo de entrega dos pacotes. A rede assume compromissos com base no valor máximo e mínimo do atraso desta entrega. Com relação à tolerância ao atraso dos pacotes, as aplicações são classificadas como aplicações de tempo-real ou aplicações elásticas.

As aplicações em tempo-real são aquelas que necessitam dos dados de cada pacote em um determinado instante de tempo. Caso o tempo de chegada dos pacotes ultrapasse um limite máximo pré-estabelecido, eles tornam-se inúteis para a aplicação. Elas podem ser classificadas em aplicações tolerantes ou intolerantes. As aplicações tolerantes não necessitam de uma garantia rígida do atraso máximo fim-a-fim dos pacotes, sendo definido o serviço de tempo-real preditivo. Já as aplicações intolerantes necessitam de uma garantia rígida do atraso máximo fim-a-fim dos pacotes, sendo definido o serviço de tempo-real garantido.

As aplicações elásticas sempre esperam pela chegada de todos os pacotes. Elas são menos exigentes quanto ao atraso fim-a-fim dos pacotes. Algumas classes de serviços *best-effort* podem ser definidas para atenderem as exigências destas aplicações.

A seguir são descritas em detalhes as aplicações de tempo-real e as elásticas, bem como os tipos de serviços que melhor se adequam a cada uma delas.

2.1.1- Aplicações de Tempo-Real

As aplicações de tempo-real mais usuais são as aplicações de *playback*. Nestas aplicações, a fonte empacota o sinal a ser transmitido e envia-o pela rede. Inevitavelmente, a rede introduz variações no tempo de entrega destes pacotes. O receptor, para manter a fidelidade na reprodução, armazena os pacotes que chegam ("buferização") para uma posterior reprodução. A reprodução do sinal ocorre com um atraso fixo deslocado no tempo com relação ao instante que o pacote foi transmitido na origem. Este atraso fixo é chamado de *playback point*. Qualquer pacote que chegar antes do seu *playback point* pode ser usado para a reprodução do sinal. Por outro lado, todos os pacotes que chegarem após o seu *playback point* são descartados.

A aplicação deve fazer a escolha antecipada do valor do deslocamento de atraso máximo permitido para a reprodução de um sinal. Este valor pode ser fornecido pela rede, por meio de um comprometimento de serviço quantitativo, estabelecendo um limite máximo de atraso. Mas, pode ser escolhido através da observação dos atrasos obtidos pelos pacotes já recebidos. O deslocamento de atraso esperado pela aplicação pode variar durante uma reprodução.

O desempenho das aplicações de *playback* é medido com relação a duas dimensões: a latência e a fidelidade. Algumas aplicações de *playback* são mais sensíveis à latência, principalmente aquelas que são interativas, como a telefonia. Outras, como as aplicações de reprodução de vídeo, são menos sensíveis. Com relação à fidelidade, as aplicações podem ser classificadas como aplicações tolerantes ou intolerantes. As aplicações intolerantes requerem uma absoluta fidelidade na reprodução, da mesma forma que as aplicações de emulação de circuitos. Já as aplicações tolerantes permitem alguma perda de fidelidade, como a maior parte das aplicações de áudio e vídeo.

O atraso da chegada dos pacotes pode afetar o desempenho das aplicações de *playback* de duas maneiras. A primeira refere-se ao valor do deslocamento de atraso, que é definido antecipadamente por meio de uma previsão do atraso dos pacotes, determinando a latência das aplicações. A segunda refere-se ao atraso de cada pacote, que pode exceder o deslocamento máximo de atraso previsto, diminuindo a fidelidade da reprodução. Assim, as aplicações podem tanto modificar o seu deslocamento de atraso a fim de fazer a reprodução dos pacotes atrasados (introduzindo uma distorção na reprodução), quanto optar pelo descarte dos pacotes atrasados (criando um sinal incompleto).

As aplicações intolerantes devem usar um deslocamento de atraso fixo, cujo valor deve ser maior que o atraso máximo permitido aos pacotes, evitando, com isso, a possibilidade de descarte de pacotes atrasados. Uma aplicação só poderá ajustar apropriadamente o atraso máximo para os pacotes, caso haja uma garantia do limite máximo de atraso para cada pacote. O serviço que oferece esta garantia é denominado serviço garantido (*guarantee services*).

As aplicações tolerantes permitem pacotes atrasados. Em vez de usar um valor fixo único para o deslocamento do atraso, elas podem tentar reduzir sua latência variando seu deslocamento de atraso em resposta aos atrasos recentemente obtidos pelos pacotes na rede. Estas aplicações são chamadas de aplicações adaptativas. Para este tipo de aplicação é proposto um modelo de serviço chamado "serviço preditivo" (*predictive service*). Esse serviço tenta assegurar um limite de atraso razoavelmente confiável.

A fim de oferecer um limite de atraso ao fluxo de dados de uma aplicação, é necessária a caracterização da natureza do seu tráfego e controle de admissão para acomodar os novos fluxos de dados. A admissão de um novo fluxo não pode prejudicar os limites já garantidos aos fluxos de dados de outras aplicações. As aplicações de *playback* com "taxa-adaptativa" podem se ajustar às condições oferecidas pela rede, não somente modificando o seu *playback point*, mas também modificando o seu próprio padrão de tráfego. Nestas aplicações, a caracterização do tráfego utilizada nos comprometimentos de serviço não é imutável, permitindo que a rede notifique-as quanto à necessidade de mudança nas características de seus tráfegos. Muitas aplicações de áudio e vídeo podem ajustar seus esquemas de codificação e, com isso, podem alterar o processo de geração de dados, dependo dos serviços disponibilizados pela rede. A alteração do esquema de codificação é uma negociação entre a fidelidade do esquema de codificação e a largura de banda necessária para o fluxo.

2.1.2- Aplicações Elásticas

Ao contrário das aplicações de tempo-real, as aplicações elásticas sempre esperam pela chegada de todos os seus pacotes de dados, mesmo aqueles que estão atrasados. Isto não significa que elas não sejam sensíveis ao atraso (quando há um aumento no atraso dos pacotes o desempenho da aplicação

diminui). A diferença com relação às aplicações de tempo-real está em seu funcionamento. As aplicações elásticas fazem uso imediato dos pacotes de dados assim que eles chegam, sem "buferizá-los" para uma reprodução posterior. Elas sempre optam por esperar todos os pacotes, ao invés de prosseguir sem eles. Assim, elas não precisam fazer uma caracterização do serviço antecipadamente. O desempenho das aplicações elásticas depende da média dos atrasos dos pacotes. Alguns bons exemplos de aplicações elásticas são: rajadas interativas (*telnet*), transferência de carga interativa (*FTP*), transferência de carga assíncrona (*e-mail*).

O modelo de serviços de entrega para aplicações elásticas é baseado no serviço *best-effort*. Várias classes de serviços *best-effort* são propostas de acordo com a sensibilidade ao atraso dos diversos tipos de aplicação. Desta forma, as aplicações de rajada interativa terão menor atraso do que as de carga interativa, e estas, por sua vez, terão menor atraso do que aquelas de carga assíncrona. Neste modelo não há controle de admissão.

2.2- Compartilhamento de Recursos

O serviço de tempo-real está baseado em compromissos assumidos para cada fluxo de dados com relação ao atraso máximo de seus pacotes. No serviço de compartilhamento de recursos, o valor de interesse passa a ser o agregado de banda nos enlaces, denominado "enlace compartilhado" (*link-shared*). O compartilhamento de enlace está relacionado com a questão de como compartilhar a banda de um enlace entre várias entidades de acordo com algum conjunto de especificações de compartilhamento.

O compartilhamento do enlace pode ocorrer de diversas maneiras. Por exemplo, várias empresas podem adquirir um enlace de dados e usá-lo conjuntamente. Neste caso, deve haver um controle de compartilhamento quando o enlace estiver sobrecarregado, bem como quando o enlace estiver em baixa carga. Um outro exemplo de compartilhamento pode ocorrer quando um enlace é compartilhado por diferentes famílias de protocolos. Neste caso, deve haver formas de controlar o compartilhamento de recursos entre famílias de protocolos, pois cada família de protocolo possui diferentes métodos de detecção e resposta aos congestionamentos. Isto vai impedir que um dos protocolos sobrecarregue o enlace, excluindo os demais. Além destes, uma outra situação de compartilhamento ocorre quando um enlace é compartilhado por vários serviços. Neste caso, frações da banda do enlace podem ser alocadas para determinadas classes de serviços.

O controle de admissão é fundamental no compartilhamento do enlace a fim de que sejam assegurados e satisfeitos os compromissos acordados.

2.3- Serviços Adicionais

Até agora todos os pacotes de um mesmo fluxo de dados tiveram um mesmo tratamento, como se todos eles tivessem um mesmo valor para a aplicação de reprodução. O que não corresponde à realidade de muitas aplicações. Em aplicações que utilizam compressão de áudio e vídeo, alguns pacotes dentro de um fluxo podem ser mais importantes que outros. Para atender as aplicações com estas características é proposto um serviço adicional ao Modelo Intserv, denominado serviço de pacotes *preemptable*. Neste serviço, alguns pacotes dentro de um fluxo de dados podem ser marcados como *preemptables*. Quando a rede estiver em perigo, o que poderá causar um não cumprimento dos compromissos assumidos, ela pode descartar alguns pacotes com esta marcação.

Apesar de poder sofrer descarte, os pacotes *preemptables* devem passar pelo controle de admissão da rede. Mas, é possível definir uma classe de pacotes que não necessita passar pelo controle de admissão. Estes pacotes são chamados pacotes "sacrificáveis". Eles são um caso extremo dos pacotes *preemptables*. A expectativa é que muitos destes pacotes sofram o descarte, portanto, não há necessidade de eles passarem pelo controle de admissão.

2.4- Mecanismos de Controle de Tráfego

O controle de tráfego é essencial para a implementação da reserva de recursos nos roteadores. Normalmente, o tratamento dado pelos roteadores aos pacotes restringe-se à escolha do melhor caminho de envio e do descarte dos pacotes. Sendo que este último ocorre quando não há *buffers* disponíveis no

roteador. Para a obtenção de QoS, algumas ações adicionais precisam ser tomadas pelos roteadores, como uma política de escalonamento e descarte de pacotes.

No modelo Intserv, o elemento responsável por estas ações adicionais é o mecanismo de controle de tráfego. Ele é composto por três módulos. O primeiro deles é o escalonador de pacotes. Este módulo implementa políticas de escalonamento e descarte de pacotes a fim de obter o QoS desejado. A política de escalonamento tem como objetivo básico a reordenação da fila de saída dos pacotes. Os pacotes com reserva têm prioridade de envio em relação aos demais. A política de escalonamento é implementada por disciplinas de enfileiramento. Um exemplo é a disciplina de enfileiramento com prioridades (*priority queuing*), em que os pacotes nas filas com maior prioridade sempre são enviados primeiro. O módulo escalonador pode utilizar outras políticas de escalonamento, utilizando diferentes disciplinas de enfileiramento, tais como: *round-robin*, *deficit round-robin* (DRR), *Weighted Fair Queueing* (WFQ) e suas variantes. Nelas, é possível criar classes de pacotes que terão acesso a uma determinada quantidade da banda passante disponível no enlace. O Capítulo 3 deste trabalho faz uma descrição destas disciplinas de filas. Já a política de policiamento do módulo escalonador implementa as políticas de descarte de pacotes. A política de descarte é tão importante quanto à política de escalonamento de pacotes. Ela decide quais pacotes serão descartados ou colocados em uma outra fila em caso de congestionamento. Por exemplo, em transmissões que utilizam o protocolo TCP, a perda de pacotes é uma sinalização de congestionamento em algum ponto da rede. Esta sinalização faz com que a fonte de dados TCP reduza a sua taxa de transmissão. Isto significa que escolher um pacote para o descarte é o mesmo que escolher uma fonte de dados TCP para minimizar a sua taxa de transmissão. Além disso, em uma fila, o descarte de um pacote reduz o *delay* de todos os outros pacotes que o sucedem na fila (que estão atrás). Assim, o descarte de um pacote pode contribuir para o sucesso dos demais pacotes. No modelo Intserv, a política de descarte de pacotes está relacionada à obtenção de um QoS, mas é importante que ela esteja em consonância com a política de escalonamento.

O terceiro módulo é o classificador de pacotes responsável pela identificação dos pacotes de dados de um fluxo. Nos roteadores atuais os pacotes são identificados pelo IP destino. Esta é uma classificação simples e suficiente para as funcionalidades que ele desempenha (a de enviar pacotes pelo melhor caminho). Mas, para a implementação das políticas de escalonamento e descarte, visando oferecer QoS aos fluxos, há a necessidade de uma classificação mais precisa de cada fluxo. Por isso, outros campos do cabeçalho IP são necessários. Os campos de endereço fonte, endereço destino, número do protocolo e número da porta são campos que podem ser utilizados para este fim, pois por intermédio deles é possível identificar um determinado fluxo inequivocamente (uma conexão fim-a-fim). A implementação do classificador é complexa e possui um *overhead* de processamento. Uma abordagem que visa diminuir este *overhead* é a de especificar um campo de identificação do fluxo (*flow-id*) no cabeçalho do pacote da camada Internet. Ele pode ser armazenado na memória *cache* dos roteadores e utilizado como atalho para as classificações dos pacotes.

O último módulo que compõe o mecanismo de controle de fluxo é o Controle de Admissão, fundamental para a oferta de serviços em tempo-real. Os serviços em tempo-real dependem da configuração de um estado descritor de reserva nos roteadores e no estabelecimento de compromissos para certas classes de pacotes. Para assegurar que estes compromissos sejam satisfeitos é necessário que os recursos sejam explicitamente solicitados, a fim de que a solicitação possa ser recusada (caso os recursos não estejam disponíveis) ou aceita (quando há recursos disponíveis). A decisão sobre a aceitação ou recusa de um novo fluxo, com base na disponibilidade dos recursos disponíveis, é chamada de controle de admissão. Para isso, os roteadores deverão ser capazes de medir o uso corrente dos recursos.

2.5- Mecanismo do Protocolo de Configuração

Além dos módulos descritos no item anterior, os roteadores precisam de um módulo a mais. Este módulo implementa o mecanismo do protocolo de configuração que é o responsável pela configuração dos demais módulos. A Figura 2.1 mostra um típico roteador que implementa as funcionalidades do Modelo de Serviços Integrados.

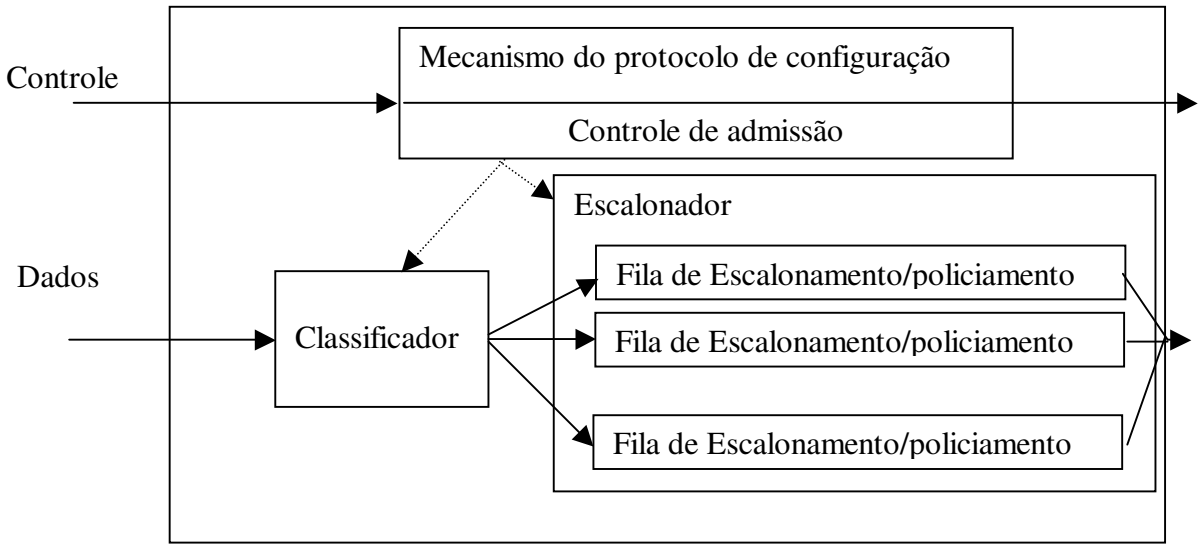


Figura 2.1- Módulos de um Roteador Intserv

O mecanismo do protocolo de configuração tem por responsabilidade a configuração dos estados dos classificadores e das filas de escalonamento no roteador, recebendo do protocolo de configuração requisições que incluem o QoS e uma entrada de classificação de pacote. Ao receber uma mensagem, o mecanismo do protocolo de configuração efetua o controle de admissão para determinar se há recursos suficientes disponíveis para a nova reserva. Se os recursos estão disponíveis, uma nova fila de escalonamento é configurada no escalonador e uma entrada de classificação é instalada na tabela de classificação.

3- PROTOCOLO DE CONFIGURAÇÃO DE RESERVA (RSVP)

O modelo de serviços Intserv prevê o uso de um protocolo de reserva de recursos para configurar e manter a reserva de recursos nos roteadores para os diversos fluxos de dados. Geralmente, o protocolo utilizado para este fim é o *Resource ReSerVation Protocol* (RSVP).

O protocolo RSVP é utilizado para fazer solicitações dinâmicas de QoS à rede. Ele foi projetado para trabalhar em ambientes *multicast* e permitir a reserva de recursos para diferentes tipos de serviço. Com o RSVP é possível que as reservas sejam compartilhadas por diversos receptores nos ramos da árvore de distribuição *multicast*, bem como novos transmissores sejam adicionados ou excluídos desta árvore.

Um *host*, utilizando o RSVP, é capaz de fazer a configuração de reserva para os fluxos de dados de suas aplicações. As solicitações de reserva são enviadas pela rede a fim de fazer a configuração da reserva em cada roteador por onde os fluxos irão passar. Todo o caminho, da origem até o destino, é percorrido. Além disso, o RSVP é encarregado pela manutenção das configurações de reserva nos roteadores durante todo o tempo que durar a conexão.

O protocolo RSVP é um protocolo de nível de transporte e usa o protocolo IP para o transporte de suas informações. Assim, ele não é o responsável pela entrega dos pacotes de uma aplicação, mas opera de forma similar a um protocolo de controle, como o IGMP. O RSVP também não é um protocolo de roteamento, mas é projetado para trabalhar junto aos atuais e futuros protocolos de roteamento, tanto *unicast* quanto *multicast*. Os protocolos de roteamento são responsáveis pela definição dos caminhos por onde os pacotes devem ser enviados, enquanto o RSVP tem como única preocupação a garantia do QoS destes pacotes através dos caminhos escolhidos.

A fim de fazer a reserva, o RSVP precisa de que alguns módulos sejam acrescidos aos roteadores, como o controle de admissão e o controle de política (*policy control*). O módulo de controle de admissão é responsável pela admissão da reserva de um novo fluxo a partir da disponibilidade de recursos do roteador. Já o módulo de controle de política determina se o usuário que está solicitando o recurso possui permissão para tal. Em caso afirmativo, o RSVP configura os parâmetros tanto do classificador de pacotes

(*packet classifier*) quanto do escalonador de pacotes (*packet scheduler*) do mecanismo de controle de tráfego descrito anteriormente.

O RSVP é um protocolo de requisição de recursos orientados ao receptor. Isto é, visando acomodar membros de grupos dinâmicos e requisições heterogêneas de receptores, o RSVP faz com que os receptores sejam os responsáveis pela requisição do QoS. As solicitações de recursos fazem o caminho reverso ao caminho feito pelo fluxo de dados da fonte geradora de dados. Além disso, a solicitação de reserva não precisa fazer todo o caminho reverso dos dados, mas apenas percorrer o caminho até alcançar um ramo da árvore de distribuição *multicast* que já possua uma reserva de recursos igual ou superior a requisitada. Desta maneira, cada receptor pode fazer uma solicitação de reserva diferenciada e de acordo com suas necessidades e capacidade de recepção.

3.1- Especificação de Fluxos (*Flowspecs*) e Especificação de Filtros (*Filter Specs*)

Em geral, uma solicitação de reserva do RSVP especifica a quantidade de recursos a serem reservados para todos os pacotes ou para um grupo de pacotes de um determinado grupo *multicast*. A quantidade de recursos é especificada pelo *flowspec*, enquanto o subconjunto de pacotes que receberão estes recursos é especificado pelo *filter spec*. Assumindo como bem sucedido o controle de admissão, o *flowspec* será usado para parametrizar uma classe de recursos no escalonador de pacotes e o *filter spec* será instanciado no classificador de pacotes, a fim de mapear os pacotes dentro de uma classe apropriada.

O protocolo RSVP fornece muitas facilidades para criar e manter os estados de reservas distribuídas através da malha de caminhos de entrega *multicast*. O RSVP trata as informações de *flowspecs* e *filter spec* como dados binários opacos (sem significado para o RSVP). Ele é o responsável por conduzir estas informações até os mecanismos de controle de tráfego dos roteadores, a fim de que estes façam a sua devida interpretação.

3.1.1- Estilos de Reserva

O RSVP oferece diferentes "estilos" de reserva. Estes estilos determinam as diversas maneiras pelas quais as solicitações de recursos de múltiplos receptores são agregadas nos roteadores, permitindo uma maior eficiência na reserva de recursos. Atualmente existem três estilos de reserva: *wildcard*, *fixed-filter* e *shared-filter*.

Uma reserva *wildcard* cria uma única reserva que é compartilhada por todas as fontes de um grupo *multicast*. Um estilo de reserva *fixed-filter* cria uma reserva diferenciada para cada fonte de dados de um grupo *multicast*. Já o estilo de reserva *shared-filter* cria uma única reserva que é compartilhada por um conjunto de fontes de dados de um grupo *multicast*. O estilo *shared-filter* diferencia do estilo *wildcard*, pois permite que o receptor especifique quais fontes que irão compartilhar uma determinada reserva.

3.1.2- Iniciação pelo Receptor

Uma outra característica importante do RSVP é que as reservas de recursos são sempre iniciadas pelo receptor. Isto traz grande flexibilidade ao modelo, permitindo ambientes com receptores heterogêneos (com necessidade de QoS diferente). Cada receptor solicita uma reserva mais adequada as suas necessidades. Quaisquer diferenças entre reservas de diferentes receptores são agregadas dentro da rede pelo RSVP. A inicialização pelo receptor é também consistente com o IP *multicast*, visto que um grupo *multicast* é criado implicitamente pelos receptores que a ele se unem.

Para uma maior simplificação e economia, um protocolo de configuração deve suportar somente iniciação em uma direção. Como mostrado acima, o RSVP usa a inicialização de reservas pelo receptor. Desta maneira, um receptor recebe o *flowspec* do transmissor (mensagem *path*) por um mecanismo de alto nível e, então, gera seu *flowspec* desejado (mensagem Resv). Este *flowspec* é enviado em direção ao transmissor percorrendo o caminho reverso das informações vindas do transmissor. A reserva vai sendo feita em cada roteador por onde o fluxo de dados passa. Não há necessidade de o *flowspec* fazer uma reserva até a fonte transmissora de dados, mas apenas até o ramo da árvore de distribuição *multicast* que já possua uma reserva que satisfaça o QoS requerido pelo receptor.

3.1.3- *Soft State*

O RSVP usa uma abordagem de criação e manutenção dos estados descritores de reserva chamada *soft states*. Esta abordagem considera o estado descritor de reserva como uma informação temporária armazenada em um cache nos roteadores e periodicamente atualizadas pelos *hosts* receptores (através das mensagens Resv). Os roteadores desativam os estados descritores de reserva que não estão em uso através da verificação do tempo de validade (tempo de expiração) destas informações. Caso a rota seja modificada, as mensagens de atualização automaticamente configuram os estados necessários ao longo da nova rota. Esta abordagem além de ser bastante simples, oferece grande robustez ao protocolo.

4- AGREGAÇÃO DE ESTADO NO RSVP

A fim de que o Modelo de Serviços Integrados [01] seja implementado na Internet, os roteadores devem ter recursos adicionais, tanto para o armazenamento de estados descritores de reserva, quanto para o processamento de pacotes. Cada grupo, que pode ser um destino *unicast* ou *multicast*, tem seu próprio estado descritor armazenado em todos os roteadores por onde seus fluxos passam. Devido ao grande número de fluxos de dados que trafegam simultaneamente na Internet, inúmeros estados descritores serão armazenados em seus roteadores mais internos, como aqueles pertencentes aos grandes *backbones*. Como conseqüência, haverá um aumento no uso de recursos de armazenamento e processamento destes roteadores. Nesta seção é descrita uma proposta de agregação de estados descritores visando à diminuição do número destes estados no modelo Intserv [04].

Segundo Berson [04], quatro estados descritores são necessários a cada grupo (*unicast* ou *multicast*). O primeiro é o estado de escalonamento que está relacionado ao tratamento do envio do pacote. O segundo é o estado de classificação do pacote que está relacionado à identificação do pacote e a sua condução ao devido tratamento que ele deve receber no roteador. O terceiro é o estado de configuração do protocolo RSVP, relacionado aos estados levados pelas suas mensagens PATH e RESV. Por fim, o quarto estado está relacionado ao protocolo de roteamento *multicast*, não abordado em [04].

O armazenamento de estados por grupo no roteador possui dois efeitos. O primeiro é o maior consumo de memória para o armazenamento de cada estado. O segundo é a necessidade de mais recursos de CPU para processá-los. Geralmente, dois tipos de memória são utilizados para o armazenamento dos estados: uma mais rápida e cara, e outra mais lenta e barata. Na memória mais rápida (*cache*) são armazenados os estados de classificação, escalonamento e de roteamento *multicast*. Na memória mais lenta são armazenados os estados da configuração do protocolo. A diminuição dos estados armazenados na memória rápida traz uma grande redução dos custos dos equipamentos e, portanto, é uma das metas do trabalho de Berson.

Para Berson, a vantagem da manutenção de um estado para cada grupo e, conseqüentemente, um controle de admissão para cada uma delas, é a garantia do isolamento dos recursos necessários para o devido tratamento das sessões. Em um esquema de agregação de estados este isolamento se perde, pois os recursos são compartilhados pelo agregado de fluxo de dados das sessões pertencentes a uma classe de serviços. Por outro lado, a agregação dos fluxos possui uma grande vantagem, como será visto a seguir, que é a possibilidade de minimizar a quantidade de estados necessários nos roteadores da rede.

Alguns termos serão definidos a fim de que a proposta de agregação de estados descrita a seguir seja melhor compreendida. O primeiro deles é o que se refere a uma Região de Agregação, que é uma região bem definida, composta por um ou mais Sistemas Autônomos implementando mecanismos para suportar a agregação de estados de fluxos. O segundo é o conceito de roteador de ingresso, que é o nó por onde entram os tráfegos na Região de Agregação. O terceiro refere-se ao roteador de egresso, o nó por onde os tráfegos deixam a região. O último é o conceito de roteador interior, que são todos os demais roteadores da região. A escolha da forma de agregação e os mecanismos usados para tal são particulares a cada região, podendo variar entre regiões vizinhas.

4.1- Módulos dos Roteadores Intserv

Como visto na seção que descreve o modelo Intserv, os roteadores que implementam os serviços deste modelo possuem cinco elementos funcionais: o escalonador, o policiador, o classificador, o controle de admissão e o mecanismo do protocolo de configuração. O módulo escalonador é responsável pelo tratamento dado aos fluxos de dados, visando atingir a qualidade de serviço (QoS) de envio desejado. Ele implementa algoritmos de escalonamento que decide a fila da qual o próximo pacote a ser transmitido será retirado. O módulo policiador implementa o algoritmo de descarte, que decide quais pacotes serão descartados ou colocados em uma outra fila em caso de congestionamento. A qualidade de serviço é obtida por meio destes dois módulos. Já o módulo classificador é o responsável pelo envio de um pacote para uma fila apropriada, de acordo com a identificação deste pacote, que geralmente é um conjunto de campos do seu cabeçalho. Existe uma tabela que relaciona a identificação do pacote com a fila apropriada ao seu tratamento. Por último, o mecanismo do protocolo de configuração tem por responsabilidade a configuração dos estados dos classificadores e das filas de escalonamento no roteador, recebendo do protocolo de configuração requisições que incluem o QoS e uma entrada de classificação de pacote.

4.2- Redução da Quantidade de Estados no Intserv

A seguir é apresentado um resumo de como os estados são utilizados no modelo Intserv e as conseqüências de sua agregação.

4.2.1- Estado de Escalonamento

O estado de escalonamento tipicamente toma a forma de classes de tráfegos, designando diferentes prioridades, compartilhamento de enlace e políticas de policiamento, com o intuito de oferecer um determinado nível de serviço. No modelo Intserv cada fluxo possui a sua própria classe de tráfego, pois tem reservado exclusivamente para ele um conjunto de recursos. Para que se possa reduzir o número de classes é necessário que os recursos sejam reservados e compartilhados por um agregado de fluxos.

4.2.2- Estado de Classificação

O estado de classificação é utilizado para designar as classes de tráfego às quais os pacotes pertencem. Cada pacote que chega a um nó é designado para uma determinada classe de tráfego de acordo com o estado estabelecido pelo protocolo de configuração. Se uma classificação dos pacotes for feita nos nós de ingresso, atribuindo cada um dos pacotes a um pequeno conjunto de classes de tráfego pré-definidas, e estas forem codificadas no cabeçalho de cada pacote, o número de estados de classificação nos nós interiores será bastante reduzido.

4.2.3- Estado do Protocolo de Configuração

O estado do protocolo de configuração é utilizado para o controle de admissão, policiamento, reclassificação e junção de mensagens de reserva. Os dois primeiros estão relacionados tanto aos tráfegos *unicast* quanto ao *multicast*. Já os dois últimos estão relacionados apenas aos tráfegos *multicast*.

No controle de admissão o estado do protocolo de configuração contém a quantidade de recursos reservados para cada fluxo admitido no roteador. Na falta deste estado, o controle de admissão pode ser feito com base na mensagem de controle de requisição e na carga atual do roteador.

No policiamento de tráfego, o estado do protocolo de configuração efetua o policiamento em uma rede Intserv nos pontos de junção e separação dos fluxos de dados. No caso de junção de dois fluxos (F1 e F2) em um terceiro (F3), há a necessidade de um policiamento neste último, como mostrado na Figura 2.2.

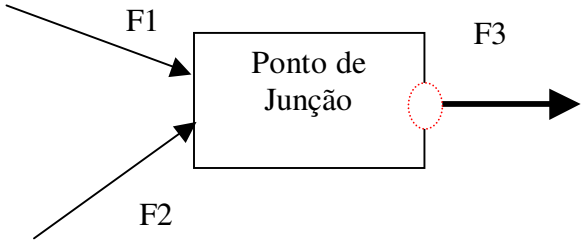


Figura 2.2- Junção de Dois Fluxos

No caso de separação, onde o fluxo de dados pode ser enviado para diferentes interfaces de saída, há a necessidade de um policiamento, visto que diferentes interfaces de saídas podem ter diferentes reservas. A Figura 2.3 mostra a separação de fluxos.

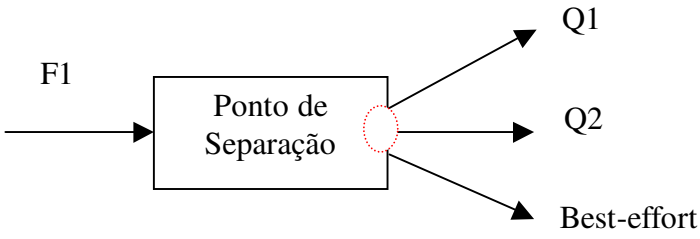


Figura 2.3- Separação de Fluxos

O estado do protocolo de configuração para o tráfego *multicast* é utilizado para permitir que nos pontos de separação, onde as reservas das interfaces de saídas são diferentes, seja feita a reclassificação do pacote para um tratamento *best-effort*. O estado do protocolo de configuração é também usado para fazer a junção das mensagens de controle de reserva dos nós de egresso para os nós de ingresso. Só as mensagens de reserva que possuem valor maior ou igual à reserva atual são passadas a diante.

4.3- Arquitetura de Agregação Básica

Berson propõe um esquema de agregação de estados no Intserv para tráfegos *unicast* com uma quantidade constante de estados de escalonador em todos os roteadores, uma quantidade constante de estados de classificação nos roteadores interiores e nenhum estado do protocolo de configuração nos roteadores internos. Para o tráfego *multicast* é proposto um estado de escalonador limitado, uma grande redução do estado de classificação e estados completos do protocolo de configuração nos roteadores.

Também é proposto um número fixo de classes de serviços que são definidas e configuradas em todos os roteadores em uma região de agregação. Cada uma delas é submetida a um controle de admissão na entrada da região. O policiamento de tráfego e a reclassificação serão feitos na entrada, tanto para tráfegos *unicast* quanto *multicast*, sendo que para este último também o policiamento será efetuado nos pontos de separação. E, uma vez que há um número fixo de classes de tráfego, o estado de escalonamento em todos os roteadores será constante. Todos os pacotes de dados são classificados e marcados na entrada da região de agregação com um identificador da classe de tráfego a qual pertence.

Um estado completo do protocolo de configuração será armazenado nos roteadores dos limites da região de agregação e, como visto, nos roteadores interiores para o tráfego *multicast*. Quando uma mensagem de reserva (RESV) chega ao nó de entrada da região (vinda do nó de saída), e esta reserva passa pelo controle de admissão, então os pacotes do fluxo associados à nova reserva serão designados para uma das classes de serviços de tráfego definidas na região.

O controle de admissão será baseado em medições e, por não haver um estado do protocolo de configuração nos nós interiores, o controle de admissão será feito no nó de entrada. A decisão se baseará na medida do congestionamento dentro da região. Assim, cada nó dentro da região, ao receber uma

mensagem de controle de admissão, tomará uma decisão de congestionamento local. Esta decisão será tomada com base em alguns limites. A nova reserva não será admitida caso a carga do nó somada a carga solicitada ultrapasse este limite.

A fim de implementar a decisão de congestionamento local, cada nó deverá manter uma estimativa de sua carga para cada classe de tráfego. Caso algum nó não seja capaz de admitir a reserva solicitada, a mensagem de admissão de reserva é marcada devidamente pelo nó e a mensagem é passada a frente. Quando a mensagem chegar ao nó de egresso, o nó verifica se a mensagem foi marcada. Caso a mensagem esteja marcada, o nó de egresso envia uma mensagem para o nó de ingresso que fará a rejeição da reserva e o envio da mensagem de erro de reserva. Caso contrário, quando todos os nós forem capazes de admitir a reserva solicitada, uma mensagem de aceitação de reserva será enviada.

4.3.1- Estado do Protocolo de Configuração *Multicast*

Uma vez que os tráfegos com múltiplos destinos estão sendo classificados e marcados nos roteadores de entrada, um pacote *multicast* marcado receberá um serviço reservado em qualquer lugar da região, incluindo os ramos para os quais não foram feitas as reservas. Além disso, os tráfegos *multicast* possuem um complicador que são as suas características de heterogeneidade e dinamicidade. A primeira refere-se à possibilidade de diferentes ramos de uma árvore de distribuição de dados *multicast* ter diferentes níveis de QoS. O que pode fazer com que os fluxos que trafegam pelos ramos sem reserva recebam um tratamento como se eles possuíssem uma reserva. A segunda está relacionada à flexibilidade de membros de um grupo *multicast* poder se conectar à árvore ou dela sair a qualquer momento. Um novo receptor pode se juntar à árvore como receptor de tráfego *best-effort*, mas receber fluxos reservados, podendo acarretar congestionamentos.

Para solucionar tais problemas, há a necessidade da manutenção do estado do protocolo de configuração em cada nó para as sessões *multicast*. Também é necessária uma entrada de classificação nos pontos de separação, onde há interfaces de saídas heterogêneas, objetivando remarcar o identificador dos pacotes para um tratamento *best-effort* nos ramos sem reserva.

4.3.2- Policiamento no Interior

Por não haver o armazenamento do estado do protocolo de configuração para os tráfegos *multicast*, pode haver problemas nos nós de junção, visto que o tráfego de saída do nó pode estar não-conforme. Isto acontece quando chegam fluxos de diferentes fontes no nó. Estes fluxos estarão conformes na sua chegada, mas devido ao compartilhamento (*wildcard* e *shared*) poderão estar não-conformes na saída. Nada pode garantir que haja recursos disponíveis entre este nó e o nó de egresso. Vale ressaltar que o policiamento ocorrerá somente no próximo nó de ingresso.

4.4- Extensões e Operação do RSVP

Duas novas mensagens de controle são propostas para o RSVP, bem como uma nova forma de operação para o controle de admissão. As duas novas mensagens são o ADREQ (ADmission REQuest) e ADREP (Admission REPly). A nova forma de operação para o controle de admissão pode ser visto na Figura 2.4.

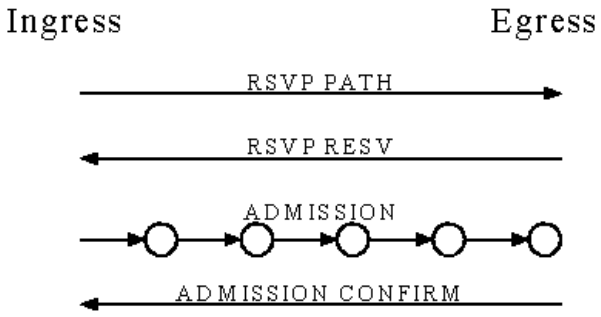


Figura 2.4- Novas Mensagens do RSVP

Com esta nova forma de operação, a responsabilidade do controle de admissão é do nó de ingresso da região de agregação. Quando ele recebe uma mensagem Resv, envia uma mensagem ADREQ. Esta mensagem, que contém a classe de serviço solicitada, passa por todos os nós por onde veio a mensagem Resv até atingir o nó de egresso da região. Cada nó, ao recebê-la, verifica a sua carga e, caso não possa suportar a nova reserva, faz uma marcação na mensagem. Caso possa aceitar a nova reserva, o nó apenas encaminha a mensagem ADREQ para o próximo nó, sem alterá-lo. Quando a mensagem de admissão chegar ao nó de egresso da região de agregação, este verifica se a mensagem está marcada. Em caso afirmativo, há uma indicação de uma rejeição por algum nó no caminho. Então, o nó de egresso envia uma mensagem de ADREP para o nó de ingresso, com uma flag indicando que a reserva foi rejeitada. Caso contrário, esta mensagem é enviada com a flag indicando que a reserva foi possível. A partir do recebimento da mensagem ADREP é que o nó de ingresso retornará ao solicitador a confirmação ou a rejeição da reserva.

5- 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). No modelo Intserv, o protocolo de reserva RSVP (*Resource ReSerVation Protocol*) 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 o *overhead* de processamento nos roteadores que estão no interior da rede, principalmente em redes WAN, 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 WANs.

Uma nova abordagem na reserva de recursos para a Internet é oferecida pelo modelo de serviços Diferenciados (Diffserv) [07]. 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 (WAN), como a Internet. Uma desvantagem é a impossibilidade de garantias precisas no atendimento aos requisitos dos fluxos de dados. Em [06], é descrita uma proposta de agregação de estados de reserva para o protocolo RSVP, tornando o modelo Intserv apropriado às redes WAN.

A seguir serão apresentados alguns conceitos que facilitarão a compreensão do modelo Diffserv.

5.1- 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*). Geralmente, um domínio DS possui uma rede provedora de serviços diferenciados e várias redes clientes. Um conjunto de domínios DS interligados forma uma região DS. A Figura 1 mostra uma região DS.

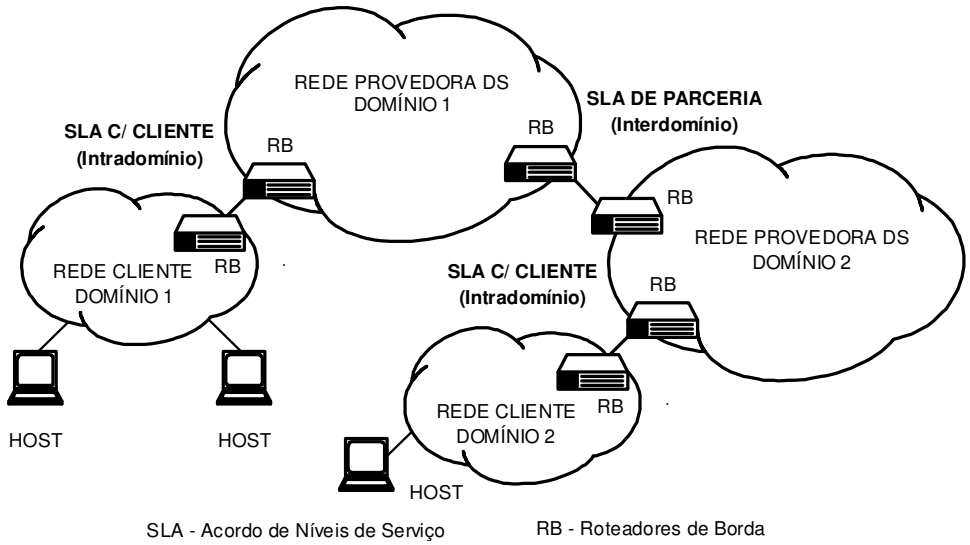


Figura 1 – Região DS

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" [07], que substitui as definições existentes do octeto do Tipo de Serviço (*Type of Service*) do IPv4 [08] e o *Traffic Class Octet* do IPv6 [09]. O campo *"DS Filed"* vazio ou com o valor zero caracteriza o serviço de entrega *best-effort*.

O comportamento que os pacotes de dados têm em um nó de um Domínio DS, devido ao tratamento recebido nele, é chamado *Per-Hop-Behavior* (PHB). O conceito de comportamento difere-se do conceito de serviço, pois o PHB é o comportamento que o agregado de fluxo irá ter em cada nó de um domínio DS, enquanto o serviço é o comportamento fim-a-fim no 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 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 se um pacote está dentro ou fora do perfil, 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 de extremidades do domínio. 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 a de tornar os nós do interior do domínio o mais simples possível (sem políticas de condicionamento), passando toda a sua complexidade para os nós das extremidades.

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

5.2- 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.

5.2.1- 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, um menor *overhead* 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.

5.2.2- 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.

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.

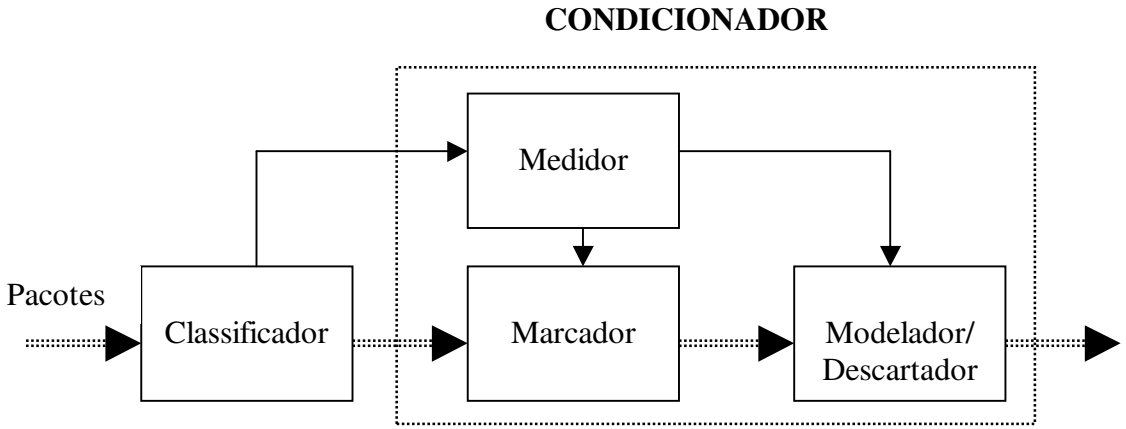


Figura 2.2 – Diagrama de blocos de um Classificador de Pacote e do Condicionador de Tráfego

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 *codepoint*. Visando garantir que somente os fluxos ou agregados de fluxos com reservas recebam uma marcação 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.

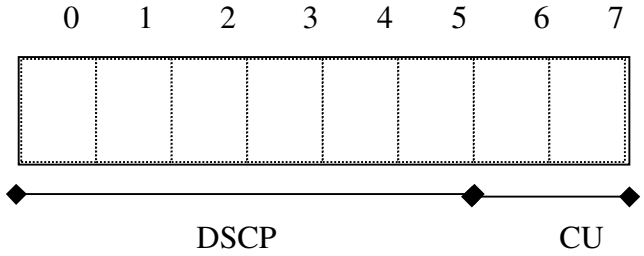
5.2.3- Escalonador de Tráfego

O escalonador é o elemento responsável por 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 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, j*itter*). O modelo Diffserv não define os mecanismos para a implementação dos escalonadores, mas apenas as características esperadas dos PHBs. Uma variedade de mecanismos de implementação pode ser adequada na implementação dos escalonadores. Os capítulos 3 e 4 descrevem uma série destes mecanismos.

5.3- Serviços Oferecidos pelo Diffserv

Um PHB é selecionado em um nó de um domínio DS pelo *codepoint* de um pacote recebido. Como descrito em [08], o *codepoint* é definido em um campo do cabeçalho IP, denominado "DS Field". Ele é composto pelos primeiros seis bits do "DS field", 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 tem 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 intradomínio.

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.

5.3.1- Classe de Precedência

Em [08] 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*) [09]. Atualmente, o campo de precedência do IP possui um amplo uso, definindo tratamentos especiais que são dados aos pacotes por ele identificados [10]. Este uso será preservado. O

que não será mantido é a compatibilidade com os bits "DTR" definidos na [09]. Na Figura 3 é mostrada a codificação dos bits "IP *precedence*" no TOS do IPv4:

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

Figura 2.5- Codificação dos Bits "IP *Precedence*" no TOS do Ipv4

A classe de precedência é uma composição dos três bits indicado na tabela da Figura 2.5 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' [10]. 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*".

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

O grupo de PHB AF [11] permite a entrega de pacotes IP em quatro classes distintas, onde cada classe pode ter até três níveis diferentes de prioridade de descarte. A Figura 2.6 apresentada uma tabela dos *codepoints* referentes às classes AF.

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

Figura 2.6- Tabela dos "Codepoints" Referentes as Classes AF

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

- 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.

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

O grupo de PHB EF [12] 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 [05]. 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 [02]. O *codepoint* recomendado para o PHB EF é "101110".

6- 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. Mas, enviar pacotes *broadcast* para todas as redes usa uma quantidade significante de recursos de rede, quando há um pequeno grupo de computadores que necessitam receber os pacotes da aplicação multiponto. A terceira forma de comunicação multiponto é a comunicação *multicast*. Nela, a aplicação envia apenas uma cópia de cada pacote para o endereços 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 dos grupo multiponto, havendo uma otimização do uso dos recursos de rede.

6.1- 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, podendo estar localizado em qualquer lugar na Internet. Os computadores interessados em receber tráfego de um determinado grupo devem fazer parte deste grupo, fazendo uma requisição explícita para se juntar ao grupo. Esta requisição é feita por meio do *Internet Group Management Protocol* (IGMP) [IGMP].

6.1.1- Endereço IP *Multicast*

O endereço multicast especifica um grupo de computadores que se juntaram ao grupo e desejam receber o tráfego enviado para este grupo. O Internet Assigned Numbers Authority (IANA) controla a atribuição de endereços *multicast*, destinando os endereços IP Classe D para o endereçamento *multicast*.

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 de endereços entre 224.0.1.0 até 238.255.255.255 é chamada endereços de escopo global [IANA]. Estes endereços podem ser usados para troca de dados *multicast* entre organizações e através da Internet. Alguns destes endereços foram reservados para algumas aplicações, como, por exemplo, o endereço 224.0.1.1 é reservado para o *Network Time Protocol* (NTP).

A faixa de endereços entre 239.0.0.0 até 239.255.255.255 é chamada endereços de escopo local ou endereços de escopo administrativo [RFC 2365], 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 [RFC 2770] propõe que a faixa de endereço 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.

6.1.2- 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 *multicas*, sobrando 23 bits para identificar um endereçco 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 mulicast* esta associado a 32 endereços IP *multicast*.

6.1.3- *Internet Group Management Protocol* (IGMP)

O IGMP faz parte da pilha de protocolos TCP/IP. Ele deve ser implementado em todos os *hosts* que desejam receber tráfego *multicast*, sendo 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 queries as sub-redes.

O funcionamento básico do IGMP pode ser resumido da seguinte forma:

- 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 a enviar mensagens IGMP *queries* periódicas 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 a seu dispositivo de rede para tal.

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

Usando *queries* e *reports*, um roteador *multicast* mantêm 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 [RFC 1112] 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. O juntar-se a um grupo *multicast* é bastante rápido na versão 1 do IGMP, mas o deixar o grupo pode ser bastante vagaroso. Isto, 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 do *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 [RFC 2236] não difere muito da versão 1. A principal diferença é que um *host* pode solicitar a sua saída de um determinado grupo multicast, através do envio da mensagem *Leave Group* ao roteador multicast de sua sub-rede. Ao receber tal mensagem, o roteador envia a 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 [IGMPv3], 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 [PIM-SSM]. Tais protocolos de roteamento *multicast* são utilizados na arquitetura proposta neste trabalho e serão discutidos a seguir.

6.1.4- 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 forma uma *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 (*shortest 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 e então o tráfego é 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*. Tanto a SPT quanto as Árvores Compartilhadas são livres de *loop*.

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, parando 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 inicia o repasse do tráfego novamente.

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 uma menor quantidade de memória nos roteadores. A desvantagem é que em certas circunstâncias os caminhos entre as fontes de dados e os receptores podem não ser os menores caminhos, introduzindo uma maior latência na entrega dos pacotes. A localização do RP na rede é fundamental para o desempenho da entrega de dados em um grupo.

6.2- 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 como ele encaminha 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* que é representado por um endereço de grupo *multicast*. O roteador *multicast* deve determinar qual a direção é de "subida do fluxo" (*upstream*) e qual direção é de "descida do fluxo" (*downstream*). Se houver múltiplos caminhos de descida, o roteador replicará o pacote recebido da fonte, repassando cada replica do pacote por estes caminhos de descida. O conceito de repasse de tráfego *multicast* baseado na fonte, ao invés dos receptores, é chamado de Repasse pelo Caminho Reverso (*Reverse Path Forwarding*).

6.2.1- *Reverse Path Forwarding* (RPF)

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 *upstream* quanto no sentido *downstream*. Um roteador somente repassa um pacote *multicast* a diante se ele o recebeu pela interface *upstream*. 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:

a) 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;
b) Se o pacote chegou pela interface do caminho de retorno a fonte, a verificação RFC é bem-sucedida e o pacote é repassado adiante;
c) Se a verificação RPF foi mal-sucedida, o pacote é descartado.

6.2.2- DVMRP

O protocolo de roteamento *multicast* DVMRP [RFC 1075] é um protocolo intradomínio. Ele usa a técnica Encaminhamento pelo Caminho Reverso (*Reverse Path Forwarding*). Quando um roteador recebe um pacote *multicast*, ele o repassa para todas as interfaces exceto por 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*). Como resultado, 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, visto que ele necessita de inundações constantes. Esta limitação é exacerbada, pois as primeiras versões do DVMRP não implementavam a "poda" dos ramos da árvore de distribuição.

O DVMRP tem sido usado no MBONE (*backbone multicast* de pesquisa na Internet), interligando redes institucionais através de túneis entre roteadores que tem habilitado o DVMRP. O MBONE é usado amplamente pela comunidade de pesquisa para a transmissão de diversas conferências.

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

O protocolo de roteamento multicast MOSPF é um protocolo interdomínio. Ele foi definido como uma extensão do protocolo de roteamento *unicast* OSPF (*Open Shortest Path First*) [19], que é um protocolo de estado de enlace (*Link-state*). 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 3.3 apresenta uma rede com roteamento OSPF dividida em áreas com os respectivos SRRs.

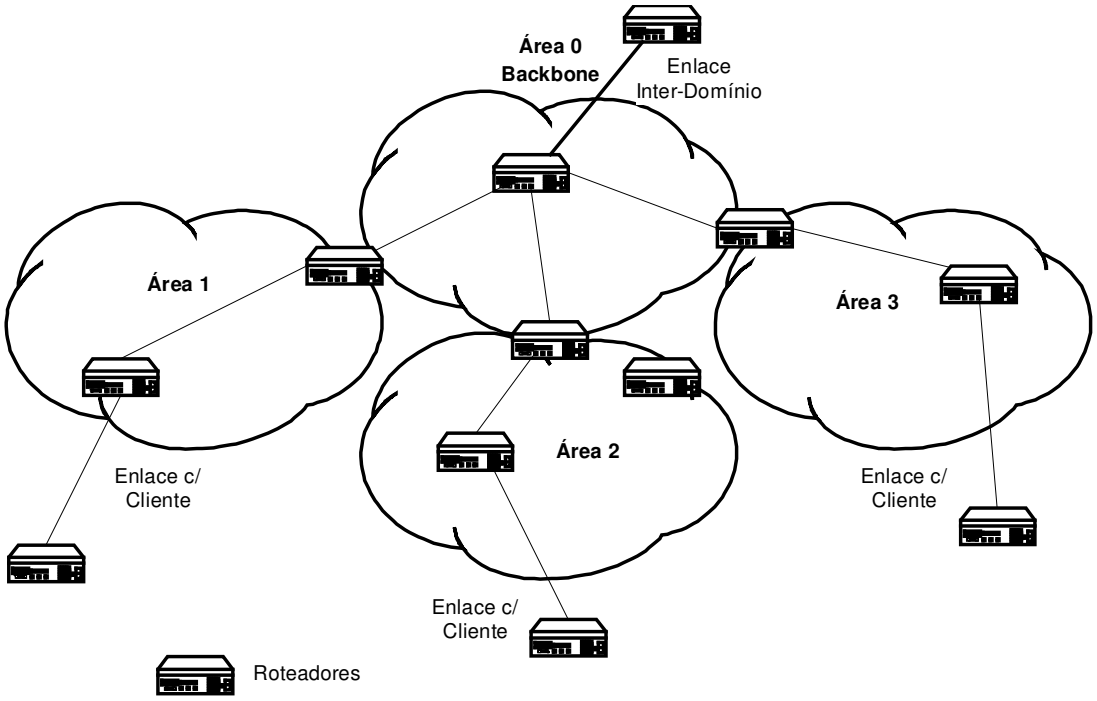


Figura xx- Rede OSPF Dividida em Áreas

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 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*) [20], 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 tem 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 quando houver uma mudança de estado de um enlace ou quando o tempo do *cache* se expira.

O MOSPF é mais adequado para ambientes que tem 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*.

6.2.4- *Protocol Independent Multicast* (PIM)

O protocolo de roteamento *multicast* PIM é um protocolo interdomínio. Ele 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 [OSPF] e BGP [BGP], e também com o roteamento estático. Ele usa a tabela de roteamento *unicast* para executar as funções de verificação de Envio pelo Caminho Reverso (RPF). O PIM não envia nem recebe atualizações de roteamento *multicast* entre roteadores como os demais protocolos, suportando dois tipos diferentes de padrões de distribuição de tráfego multiponto: denso e esparso.

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

O PIM-DM 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 de descida do fluxo (*downsstream*), enviam mensagens de "poda" de retorno, na direção de subida do tráfego (*upstream*), para não receber tráfego desnecessário. Este processo se repete a cada três minutos.

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:

a) As fontes e os receptores estão próximos uns dos outros;
b) há poucas fontes e muitos receptores;
c) o volume de tráfego *multicast* é intenso;
d) o tráfego *multicast* é constante.

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

O PIM-SM [RFC 2362] usa um modelo to tipo "*pull*" para a entrega de tráfego *multicast*. Somente segmentos de rede que tem 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 trêfego pode permanecer na árvore compartilhada ou se 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, parando de enviar o tráfego pela árvore compartilhada.

O modo esparço é mais adequado quando:

a) Há poucos receptores em um grupo;
b) as fontes de dados e os receptores são separados por enlaces WAN;
c) 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.

6.2.5- *Multiprotocol Border Gateway Protocol* (MBGP)

O protocolo de roteamento *multicast* MBGP [RFC 2283] é um protocolo interdomínio, sendo uma extensão do protocolo de roteamento *unicast* BGP [BGP]. Ele oferece um método para provedores distinguirem quais rotas prefixadas serão usadas para a execução da verificação RPF. A verificação RPF é o mecanismo fundamental usado pelos roteadores para determinar os caminhos por onde os ramos da árvore de distribuição *multicast* se estenderão e usados para a entrega do tráfego *multicast* aos receptores do grupo.

Por ser um extensão do BGP, o MBGP possui todos os mecanismos administrativos que provedores e clientes desejam em seus ambientes de roteamento interdomínio, incluindo todas as ferramentas inter-AS para filtrar e controlar o roteamento (ex., mapeamento de rotas). Portanto, usando o MBGP, qualquer rede utilizando o BGP interno (iBGP) ou o BGP externo (eBGP) pode aplicar as múltiplas políticas de controle usuais no BGP para especificar a política de roteamento para o tráfego *multicast*.

Dois atributos de caminhos, MP_REACH_NLRI e MP_UNREACH_NLRI, foram introduzidos no BGP4+. Estes novas atributos criam uma maneira simples de carregar dois conjuntos de informações de roteamento, um para roteamento *unicast* e outro para roteamento *multicast*. Os roteadores associados para o roteamento *multicast* usam a verificação RPF nas bordas interdomínios.

A principal vantagem do MBGP é que uma inter-rede pode suportar topologias *unicast* e *multicast* não congruentes. Quando as topologias *unicast* e *multicast* são congruentes, o MBGP pode suportar diferentes políticas para cada uma das topologias.

6.2.6- *Border Gateway Multicast Protocol* (BGMP)

O BGMP [BGMP] é um protocolo de roteamento *multicast* interdomínio. Ele constrói árvores compartilhadas de domínios para os grupos *multicast* ativos. Esta árvore é bi-direcional e forma uma hierarquia de domínios. O domínio de maior hierarquia é o domínio raiz e os demais são chamados domínios folhas. A árvore é construída a partir dos domínios folhas em direção ao domínio raiz. Entre um domínio folha e o domínio raiz pode haver outros domínios folhas. Uma característica importante deste protocolo é que estando em um domínio folha e sabendo o endereço *multicast* do grupo, é possível descobrir o próximo domínio folha que está na direção do domínio raiz da árvore de distribuição daquele grupo. O BGMP requer que cada grupo *multicast* interdomínio seja associado a uma única raiz, que no BGMP não é um roteador, mas um domínio inteiro. A Figura xx mostra uma árvore compartilhada de domínios. Nela, um domínio é formado por uma rede provedora de acesso e redes clientes. Um domínio pode ser considerado um AS.

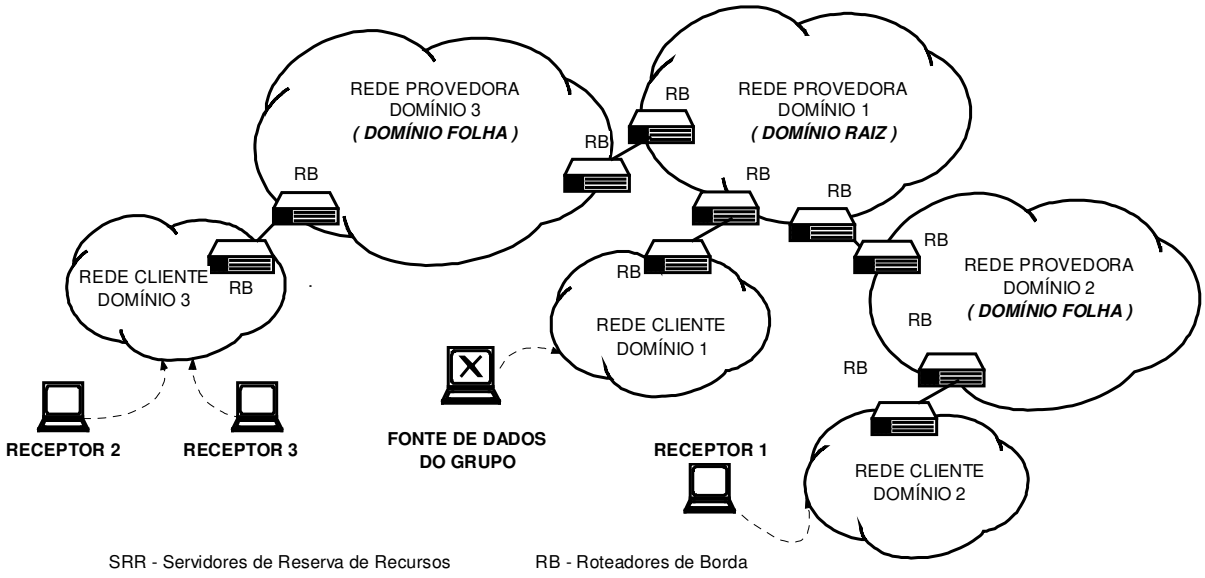


Figura xx – Árvore Compartilhada de Domínios

O BGMP assume que faixas de endereços IP classe D sejam associados aos domínios que farão parte da árvore de domínios *multicast*. Cada domínio torna-se a raiz da árvore compartilhada para todos os grupos cujo endereço está dentro de sua faixa de endereço associado. Um protocolo de alocação de endereços, como o MASC (*Multicast Address-Set Claim*) [13], deve ser usado nos domínios que farão parte da árvore de distribuição.

O MASC [MASC] forma a base para uma alocação de endereços hierárquicos. O domínio executando o MASC forma uma hierarquia baseada na estrutura de uma topologia interdomínio existente. Então, o MASC aloca dinamicamente faixas de endereços para os domínios usando uma abordagem de escuta e reivindicação com detecção de colisão (*listen and claim with collision detection*). Nesta abordagem, domínios "filhos" escutam as faixas de endereço *multicast* selecionados pelos seus "pais", selecionando sub-faixas de endereços a partir das faixas de endereços dos domínios "pais". A seguir, o domínio reivindicador propaga o endereço alocado para os seus domínios "irmãos" (domínios ligados ao mesmo domínio "pai"). O domínio reivindicador espera por um determinado período para detectar se houve qualquer colisão na alocação de endereços, antes de comunicar a faixa de endereços adquirida para o Servidor de Alocação de Endereços *Multicast* (MAAS) e aos outros domínios. Esta comunicação é feita por meio do protocolo *unicast* interdomínio BGP como rotas de grupo.

O BGMP é executado nos roteadores de borda dos domínios. Ele usa as rotas de grupo do BGP para construir a árvore de distribuição compartilhada para os grupos ativos. A árvore é formada pelos roteadores de borda dos domínios que estão ao longo dos caminhos entre os domínios fontes/receptores e o domínio raiz do grupo. Cada árvore compartilhada tem a sua raiz no domínio cujo endereço alocado inclui o endereço do grupo. Assim, o domínio que injeta a faixa de endereço *multicast* no BGP é o domínio raiz para este grupo. Uma vez que o iniciador de um grupo normalmente pega um endereço *multicast* da faixa alocada para seu domínio, a árvore compartilhada tem este domínio como a sua raiz. Protocolos multicast intradomínio, tais como PIM e DVMRP, são usados para o envio de dados entre os membros de um grupo no interior de um domínio e seus roteadores de borda BGMP. Por sua vez, estes roteadores de borda enviam os dados pela árvore compartilhada de domínios para alcançar os membros nos outros domínios.

6.2.7- *Multicast Source Discovery Protocol* (MSDP)

No PIM-SM as fontes e os receptores devem se registrar no Rendezvous Point (RP). Normalmente, o roteador mais próximo das fontes e dos receptores é eleito o RP. Este roteador tem conhecimento sobre todas as fontes e receptores de um determinado grupo. O problema é que um RP não tem como conhecer as fontes localizadas em outros domínios. O MSDP [MSDP] foi desenvolvido para

solucionar este problema. Ele é um mecanismo que conecta domínios PIM-SM, permitindo que diferentes RPs compartilhem informações sobre fontes ativas. Quando os RPs em domínios remotos têm conhecimento sobre as fontes ativas, eles podem passar estas informações para seus receptores locais e o tráfego *multicast* pode ser repassado entre os domínios. Uma importante característica do MSDP é que ele permite que cada domínio mantenha um RP independente dos outros domínios, mas possibilita aos RPs repassarem tráfego entre domínios.

O RP em cada domínio estabelece uma sessão de parceria MSDP usando uma conexão TCP com o RP em outro domínio ou com os roteadores de borda na direção do outro domínio. Quando o RP aprende sobre uma nova fonte *multicast* dentro de seu próprio domínio (por meio de mecanismo normal de registro do PIM-SM), o RP encapsula o primeiro pacote de dados em uma mensagem de Fonte Ativa (Source Active – SA), enviando o AS a todos os parceiros MSDP. O AS é repassado para cada parceiro receptor usando uma verificação RPF modificada, até que ele atinja todos os roteadores MSDP dos Domínios parceiros. Se o parceiro MSDP é um RP, e o RP já possui uma entrada (*,G) para o grupo encapsulado no SA (há um receptor interessado), o RP criará um estado (S,G) para a fonte e irá juntar-se a árvore de caminhos mais curtos para esta fonte. O pacote de dados é desencapsulado e repassado pela árvore compartilhada do RP. Quando o pacote é recebido pelo último roteador antes do receptor, ele pode também se juntar a árvore de caminho mais curto daquela fonte. O RP da fonte periodicamente envia SAs que incluem todas fontes dentro de seu domínio. A Figura xx mostra o tráfego *multicast* entre uma fonte no Domínio A e um receptor no Domínio E.

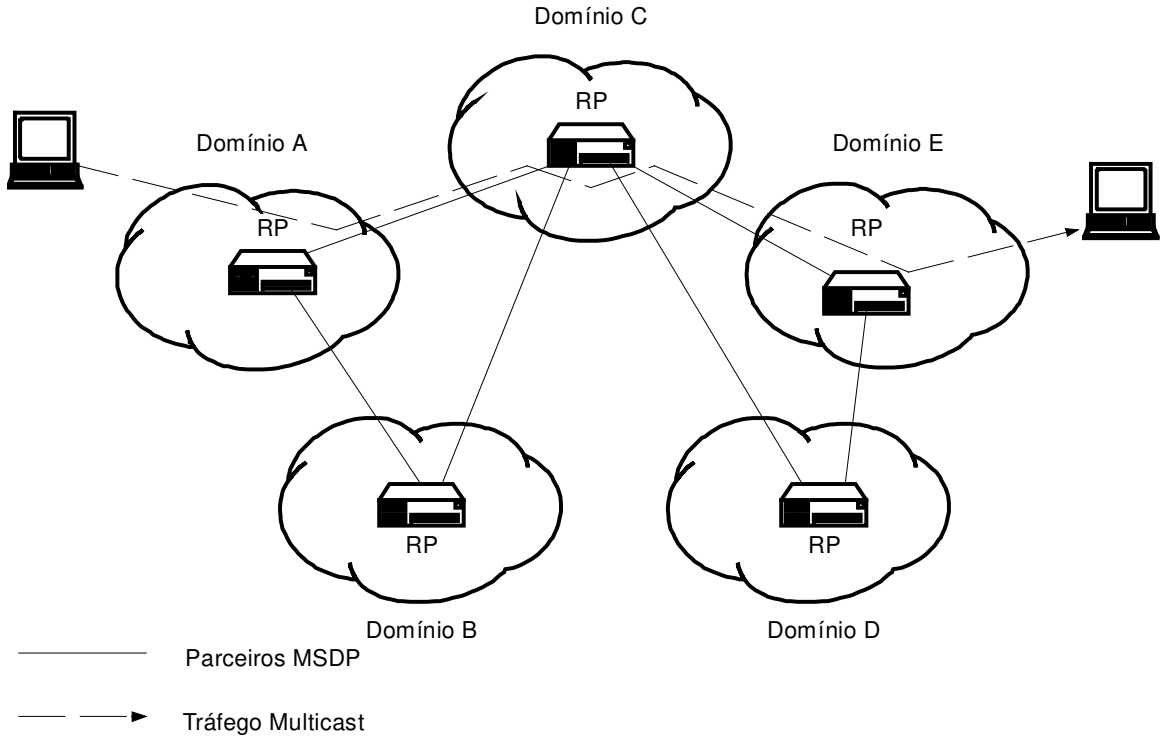


Figura Xx – Troca de Informações entre RPs

O MSDP foi desenvolvido para parceria entre Provedores de Serviços de Internet (ISPs). Os ISPs não querem confiar em um RP mantido por um ISP concorrente para serviços aos seus clientes. O MSDP permite que cada ISP tenha seu próprio RP e ainda repasse e receba tráfego *multicast* na Internet.

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

A característica do Multicast de uma Fonte Específica 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 são baseadas no protocolo PIM-SM e no protocolo MSDP. 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) [RFC 1112]. Por exemplo, com o ISM as redes devem manter o conhecimento sobre quais hosts na rede 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 (MSDP) e nem é necessário usar os mecanismos como a inundação (flood) ou Rendezvous Point. Como o receptor conhece a fonte de dados, a rede pode enviar a solicitação se "join" em direção a fonte. A comunicação com este roteador deve ser feita pelo IGMP versão 3, que permite a identificação do grupo e da fonte da qual o receptor deseja receber o tráfego.

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. Para se tornar membro de um grupo, o host deve usar o protocolo IGMP.

No SSM, a entrega de datagramas IP é baseado em canais (S,G). O tráfego para um canal (S,G) consiste de datagramas com um endereço IP unicast da fonte S e 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 o canal (S,G) a fim de, respectivamente, receber ou não receber o tráfego de uma determinada fonte. 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 services ISM existents 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.

6.3- 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. Na definição do modelo Diffserv [04] são apresentadas duas considerações para a implementação deste tipo de tráfego, de modo a evitar que os recursos reservados para o tráfego *unicast* sejam comprometidos. A primeira é que o tráfego *multicast* deve ter um conjunto de *codepoints* diferenciado do conjunto de *codepoints* destinado ao tráfego *unicast*. A segunda é a de que o tráfego *multicast* deve ter um SLA diferenciado do SLA do tráfego *unicast*.

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.

6.3.1- 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 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*.

6.3.2- Proposta de Bless e Wehrle

Uma importante contribuição para a implementação do tráfego *multicast* no Diffserv foi a proposta feita por Bless e Wehrle [13]. 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, pouco *overhead* de processamento será acrescido 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 a tabela de roteamento *multicast*. Este campo deve conter o valor do *codepoint* para a remarcação do *"DS Filed"* dos pacotes oriundos de uma determinada fonte *multicast*. Inicialmente, todas as estradas da tabela *multicast* devem ter este novo campo contendo o valor do *codepoint* equivalente ao serviço LBE (*Lower than Best-Effort*) [14], 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 Filed"* 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* da qual o receptor pertence. Quando um roteador recebe esta mensagem, ele deve solicitar a uma entidade de gerenciamento (*Bandwith Broker*) a admissão de um novo fluxo. Como os roteadores interiores de um domínio devem ser mais simples (com menos *overhead* 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.

7- POLÍTICAS DE ESCALONAMENTO

Uma rede de computadores possibilita aos usuários o compartilhamento de recursos, tais como: sistemas de arquivos, impressoras, linhas de comunicação, etc. Como os recursos compartilhados são geralmente insuficientes para atenderem simultaneamente a todos os usuários que o solicitam, surge então o problema da disputa (contenção) pelos recursos compartilhados.

Usualmente, as requisições para um determinado recurso são feitas ao servidor daquele recurso. Aquelas requisições que foram feitas e ainda não puderam ser atendidas são postas em uma fila de serviço, onde esperam até o momento de serem servidas (o tempo que uma requisição leva na fila até ser atendida é definido como atraso de enfileiramento - *queuing delay*). A política de escalonamento decide qual é a próxima requisição a ser servida, definindo a ordem de atendimento das requisições. Desta maneira, a política de escalonamento pode ser utilizada para estabelecer diferentes classes de QoS para os fluxos de dados de diferentes conexões ou agregados de fluxos.

As políticas de escalonamento são as responsáveis pelo tratamento diferenciado que os nós de um domínio DS dão aos pacotes de um fluxo ou agregado de fluxos de dados. Estas políticas são implementadas através de disciplinas de enfileiramento, estando localizadas nos módulos de envio dos pacotes à rede. Nesta seção são descritas tanto as características das políticas de escalonamento, quanto alguns exemplos de disciplinas de enfileiramento sugeridas para a implementação do modelo Diffserv. No Diffserv, os recursos compartilhados são normalmente a largura de banda dos enlaces e os *buffers* nos roteadores. Portanto, as políticas de escalonamento discutidas a seguir dão mais ênfase ao compartilhamento destes recursos, o que não inviabiliza a aplicação delas no compartilhamento de outros tipos de recursos.

7.1- Lei da Conservação

Uma disciplina de enfileiramento bem conhecida é a *First-Come-First-Server* (FCFS). Nesta o escalonador transmite os pacotes na ordem de sua chegada, descartando aqueles pacotes que chegarem 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 enfileiramento 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.

7.2- Requisitos para as Disciplinas de Escalonamento

As disciplinas de escalonamento devem satisfazer aos seguintes requisitos:

7.2.1- 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*.

7.2.2- 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. Para transportar uma decisão tomada localmente 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.

7.2.3- 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.

7.2.4- 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.

7.2.5- 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.

7.3- Projetos de Disciplinas de Escalonamento

Existem quatro graus de liberdade no projeto de disciplinas de escalonamento que são descritos a seguir.

7.3.1- Níveis de Prioridade

Nos projetos de escalonadores com prioridade, os fluxos de dados de cada conexão são associados a um nível de prioridade. Desta forma, o escalonador pode oferecer aos pacotes dos fluxos das conexões de maior prioridade um menor atraso de enfileiramento, às custas 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 *starvation*, 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**.**

7.3.2- 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 de descida do fluxo (*downstream*) quanto a variação do delay (*jitter*) experimentado pelos pacotes.

7.4- Disciplinas de Enfileiramento

Nesta seção serão apresentadas algumas disciplinas de enfileiramento que implementam as políticas de escalonamento de pacotes. 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*, *Start-Time Fair Queuing* e a *Class-Base Queuing*.

7.4.1- 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.

7.4.2- *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 com pesos diferentes (utilizam intervalos de tempo de serviços diferenciados), permitindo que o atendimento dos pacotes seja equivalente aos pesos dados a cada fila. Por exemplo, suponha três classes de serviço possuindo pacotes com 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.

Se os pacotes das filas possuem tamanhos diferentes, o que acontece na Internet, o servidor da disciplina de enfileiramento 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, suponha que a média do tamanho de pacotes no exemplo anterior 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.

7.4.3- *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 enfileiramento 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, sendo bastante indicada em implementações em que a justiça de uma fila não seja muito rigorosa e o tamanho dos pacotes seja pequeno.

7.4.4- *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 nehuma 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 ("interated 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 cálculo do *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. O *round number* é o inverso do número de conexões ativas. Por exemplo, caso haja quatro conexões ativas, o *round number* corrente é 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, dividi-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 e10 respectivamente, o valor 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.

7.4.5- 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 mantém boa parte do seu desempenho.

A primeira delas é a *Self-Clocked fair Queuing* (SCFQ). Nela, quando um pacote chega a uma conexão vazia, 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*.

7.4.6- *Class-Base Queuing* (CBQ)

CBQ [15] refere-se a um gerenciamento de recursos baseado em classes hierárquicas [16]. O CBQ especifica uma política desejável em termos de divisão da largura de banda em um determinado enlace em momentos de congestionamento. A Figura 3.1 mostra um exemplo de divisão em classes de um enlace.

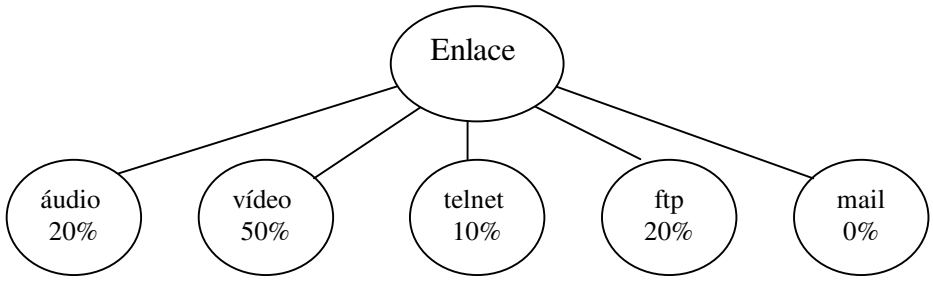


Figura 3.1 – Divisão de um Enlace em Classes

A classe de áudio é uma classe folha e, com relação à classe de enlace, é considerada uma classe interior. As porcentagens referem-se à quantidade relativa de largura de banda que cada classe, em períodos de congestionamentos, deverá ter aproximadamente.

Os objetivos gerais de um compartilhamento de enlace são:

- cada classe interior ou folha deve receber aproximadamente sua largura de banda alocada sobre intervalos de tempos apropriados;
- se todas as classes interiores e folhas, com suficiente demanda, receberem, no mínimo, a largura de banda alocada, a distribuição do excesso de banda não deve ser arbitrário, mas deve ser feito com base em uma política de distribuição razoável.

Portanto, a política de gerenciamento de recursos CBQ não define políticas de escalonamento e de descarte, mas pressupõe o seu uso para alcançar os objetivos acima descritos.

8- 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 necessário uma estratégia de descarte de pacotes. Um pacote só deverá ser descartado em caso de absoluta necessidade. As estratégias de descarte são conhecidas como políticas de descarte. 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.

8.1- Classificação das Políticas de Descarte

As políticas de escalonamento podem ser classificadas de acordo com quatro características:

- nível de agregação;
- escolha de prioridades de descarte;
- descarte antecipado ou em sobrecarga;
- posição de descarte.

A seguir estas características serão descritas.

8.1.1- 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 quando há *buffers* disponíveis, os fluxos alocam a quantidade de *buffers* que precisam.

8.1.2- 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 cuja origem é mais distante já usou uma grande quantidade de recursos da rede e, caso sofra descarte, haverá um maior desperdício de recursos.

8.1.3- 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*. Este fenômeno 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 de descarte que descartem pacotes antes das filas tornarem-se cheias. Com este objetivo é que foram desenvolvidas políticas com descarte antecipado, onde o descarte de pacotes ocorre antes dos *buffers* das filas estarem preenchidos. 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) [17]. 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.

A implementação do descarte antecipado, RED, possui duas características importantes. A primeira é que os pacotes são descartados baseado em uma média exponencial do tamanho da fila, em vez de um tamanho em um dado instante. Desta forma, pequenas rajadas de tráfego são permitidas, havendo apenas o descarte quando as rajadas possuem duração mais longa. A segunda, é que a probabilidade de descarte dos pacotes é uma função linear da média do tamanho de fila. O significado disso é que quanto maior a fila, maior a probabilidade de descarte dos seus pacotes.

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.

8.1.4- Posição de Descarte

O pacote que será descartado não é necessariamente aquele que está no final de uma fila. Ele pode ser o primeiro da fila. Pode estar em uma posição aleatória da fila. Como também uma fila pode 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.

8.2- RED

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 baseado 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.

Capítulo 2

ARQUITETURA PARA TRÁFEGOS *MULTICAST* NO DIFFSERV

1- INTRODUÇÃO

Este capítulo apresenta a arquitetura de rede que tem por objetivo possibilitar o tráfego *multicast* em redes que implementam os serviços diferenciados, fazendo uma verificação antecipada da disponibilidade de recursos e a sua reserva nos enlaces por onde passa o tráfego *multicast*. Ela 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 *Bandwith Brokers* [05], 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 [10] é descrita uma arquitetura para reservas antecipadas para o modelo Intserv/RSVP.

Os roteadores implementam o serviço "*multicast Premium*" utilizando o PHB *Expedited Forwarding* [12]. O serviço *premium* [11] 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 [04], 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 [13]. 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 citada, mas não descrita, 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.

O estudo das arquiteturas apresentadas em [06] e em [05], além da proposta apresentada em [13] foram importantes para a elaboração da nossa arquitetura.

2- 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 a 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.

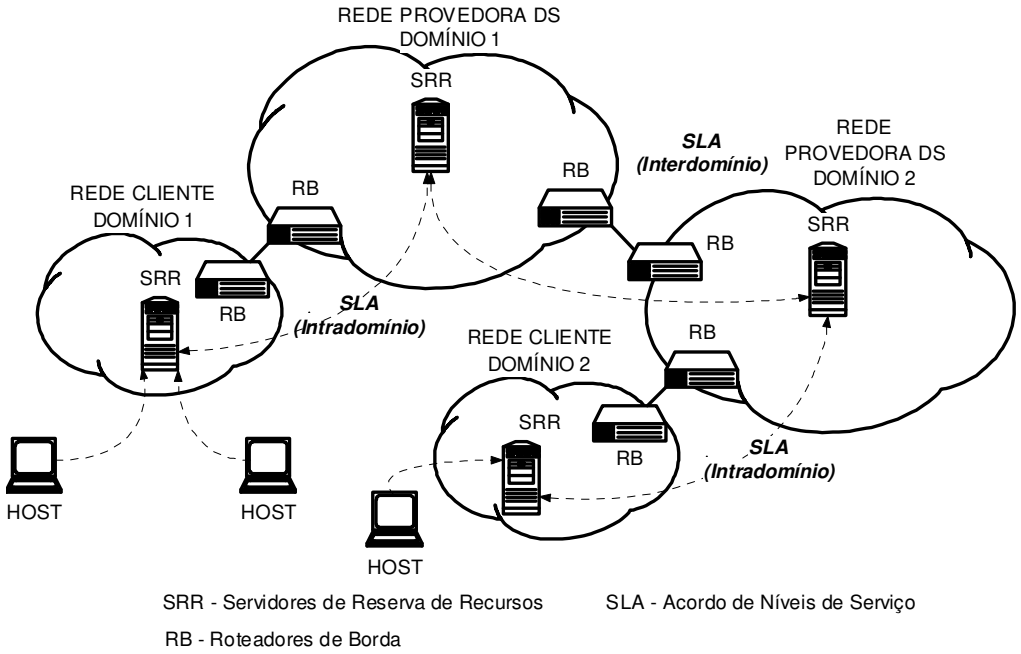


Figura 3.1 – *Comunicação* entre SRRs em um Domínio DS

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 1 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 Filed"* (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.

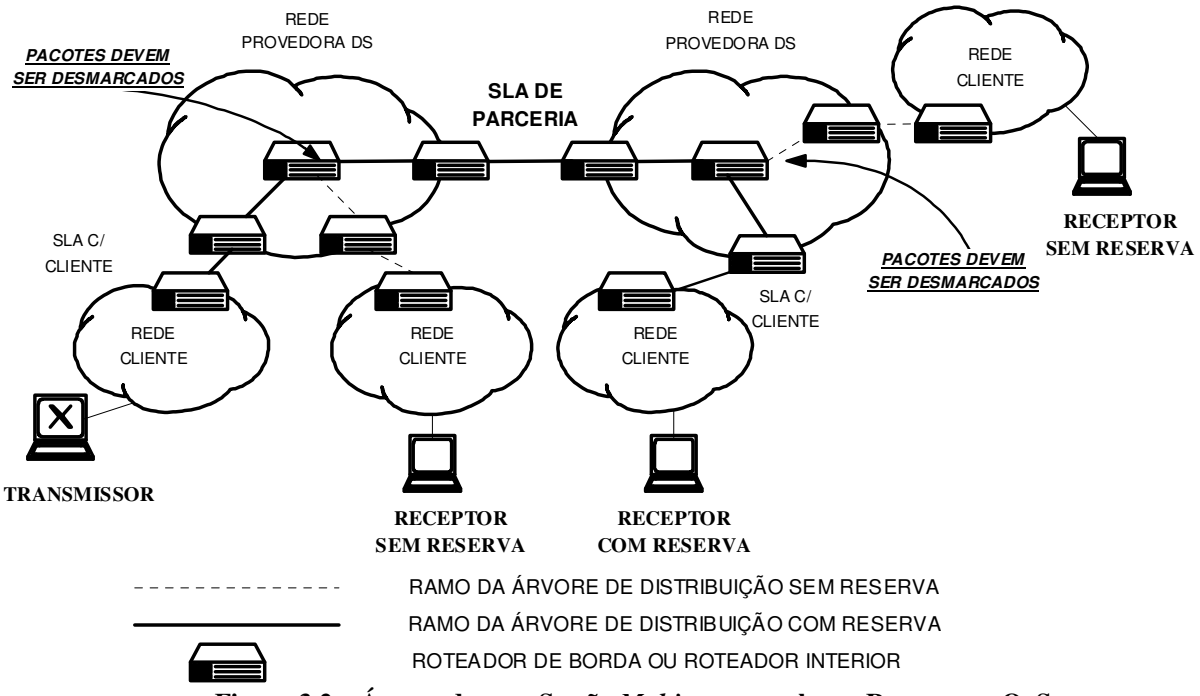


Figura 3.2 – Árvore de uma Sessão *Multicast* com alguns Ramos sem QoS

A configuração especial citada consiste em criar uma *flag* 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 placa 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 placas de rede um determinado pacote *multicast* deve ser enviado. Inicialmente, todas as entradas da tabela possuem a *flag* 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 a *flag* 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 [12].

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 o 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- 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 condução dos pacotes de dados *multicast* no interior de um Domínio DS e o protocolo de roteamento *multicast* interdomínio com a responsabilidade pela condução dos pacotes de dados *multicast* 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 [CIC 2000]. 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 DVMRP foi substituído pelo MOSPF.

A abordagem de usar dois tipos de protocolos multicast distintos parecia ser a mais adequada à Internet, visto que não obrigava aos diversos Domínios DS interligados em uma Região DS 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 grupo e definem de qual fonte de dados eles querem 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. A grande vantagem do uso deste protocolo é que ele pode trabalhar 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*) [20], 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 é descrito na seção sobre o protocolo *multicast* MOSPF no capítulo 1 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 projetada, 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 xx 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.

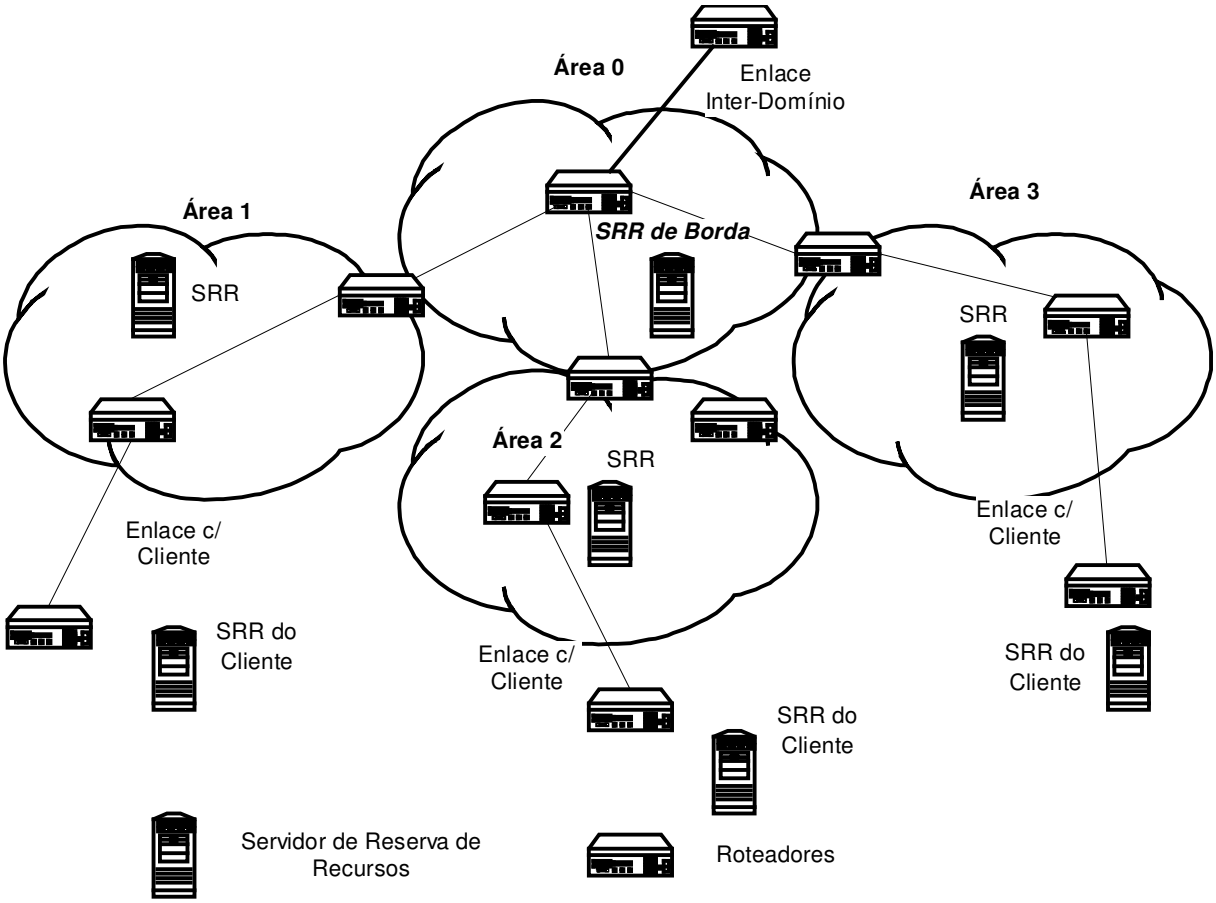


Figura 3.3- Rede OSPF Dividida em Áreas e os SRR de cada Área

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 tem que 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.

4- PROCEDIMENTOS DE RESERVA

O processo de reserva de recursos é iniciado a partir da criação de um perfil com as características de um grupo *multicast*. Este perfil é armazenado no(s) SRR(s) de borda do Domínio DS do transmissor.

Para que os receptores de um grupo possam receber tráfego *multicast* com QoS, eles devem fazer uma requisição de reserva, com base no perfil daquele grupo, 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 *multicast* 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.

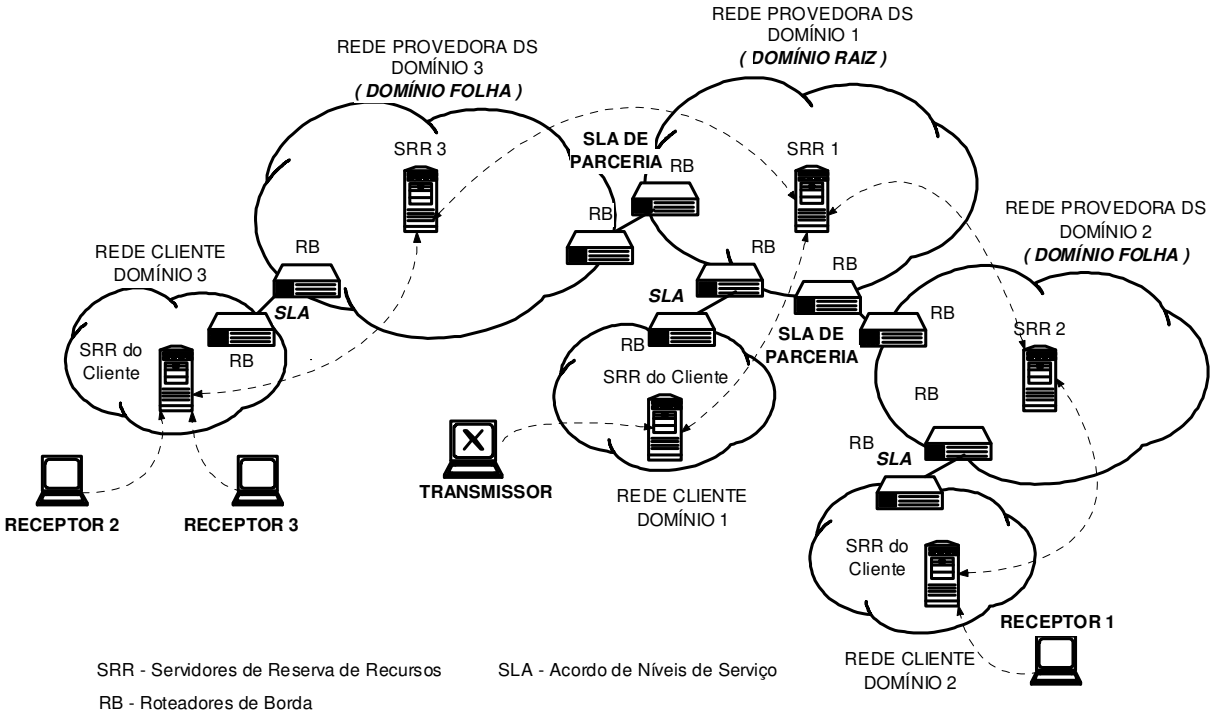


Figura 3.4 - Árvore Multicast Intradomínio e Interdomínio

A Figura 3.4 mostra um esboço de uma árvore de interligação de SRRs de uma fonte/grupo *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.

5- 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.

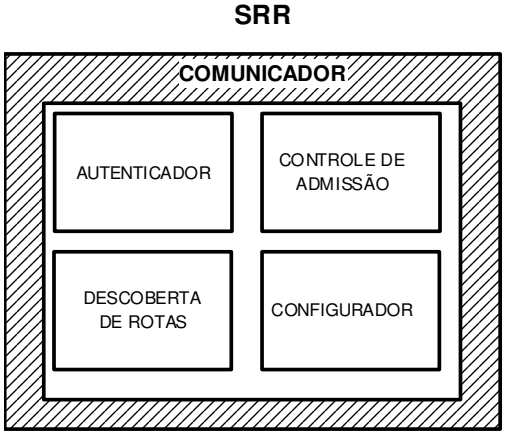


Figura 3.5 - Estrutura Modular de um SRR

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 têm disponíveis as quantidades de recursos solicitadas.

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 as 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 a 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 a 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 no capítulo referente à implementação.

6- 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 2. A estrutura funcional dos roteadores é composta pelos seguintes módulos: classificador, condicionador, escalonador, configurador e comunicador. Os roteadores internos não utilizam o módulo condicionador.

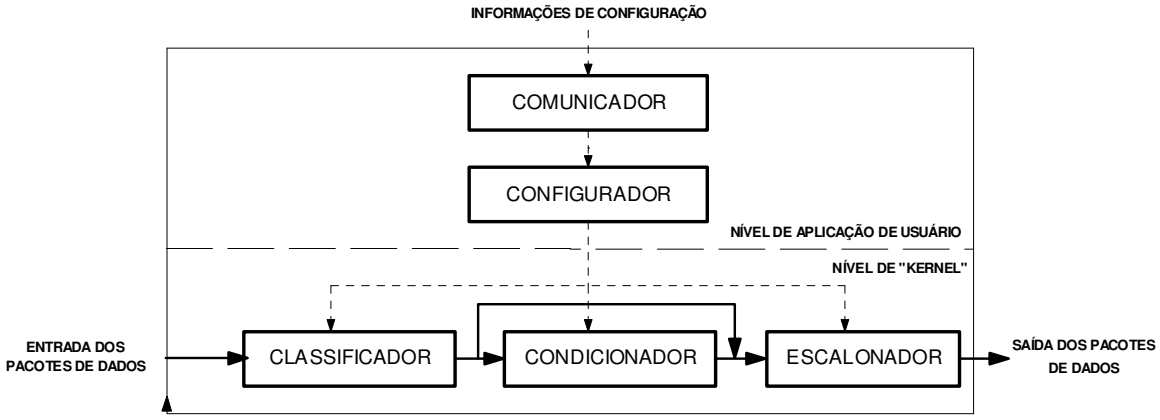


Figura 3.7 – Estrutura Funcional dos Roteadores

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 ("Behaviour-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 IP 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 e modelador. O sub-módulo marcador atribui ou modifica um valor de *codepoit* 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. O módulo condicionador faz o condicionamento tanto dos fluxos individuais quanto dos agregados de fluxos.

O módulo escalonador implementa disciplinas de fila a fim de a 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.

7- ESQUEMA DA INTEROPERABILIDADE ENTRE OS COMPONENTES DA ARQUITETURA

A Figura 3.8 mostra o esquema da interoperabilidade entre os componentes da arquitetura proposta para o tráfego *multicast* com QoS.

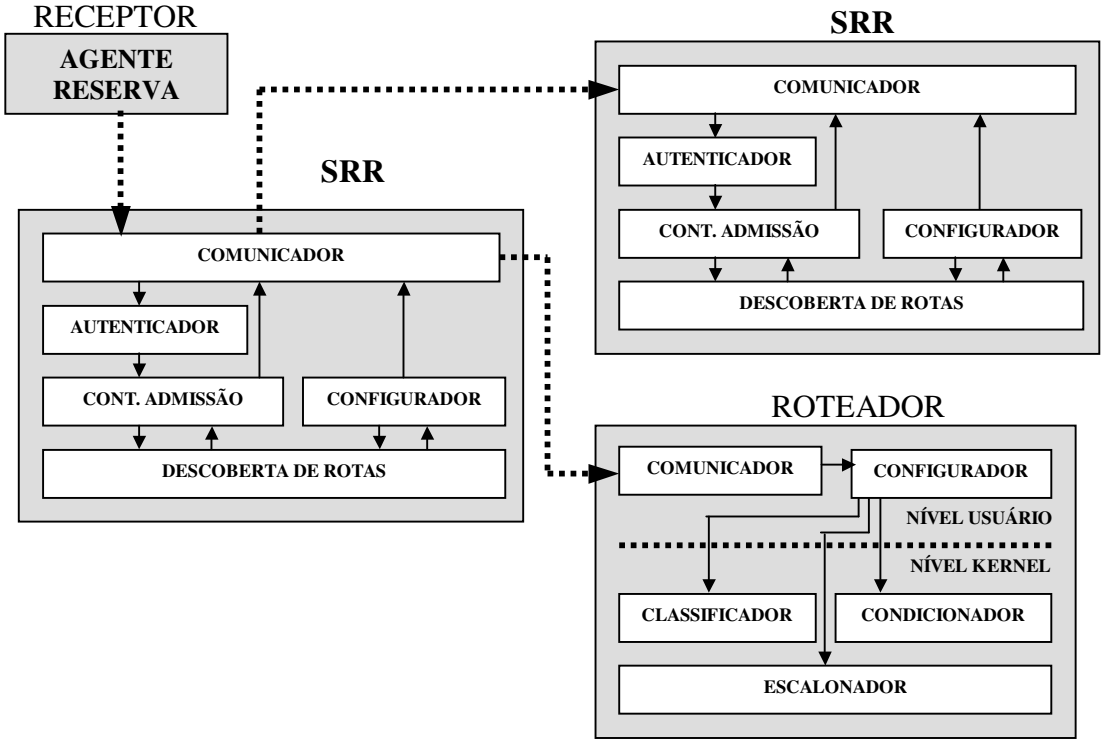


Figura 3.8 – Esquema da Interoperabilidade entre os Componentes da Arquitetura

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. A descrição completa da implementação de cada um destes componentes e os seus respectivos módulos será descrita no capítulo referente à implementação.

Capítulo 3

IMPLEMENTAÇÃO

1- INTRODUÇÃO

Este capítulo analisa a implementação de dos elementos que constituem tanto o ambiente de serviços diferenciados quanto o sistema de reserva de recursos.

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. Tornou-se necessária a criação de novas políticas de escalonamento de pacotes, classificadores e condicionadores de tráfego, bem como acrescentar recursos para tráfego *multicast* no TROPIX. Também foi criada uma ferramenta de geração de tráfego para avaliar o desempenho dos recursos implementados nos roteadores que compõe tal ambiente e um aplicativo de configuração destes roteadores.

Além disso, um protótipo do sistema de reserva de recursos foi desenvolvido. Este protótipo permite a realização de requisições de reserva para uma determinada fonte/grupo *multicast* e a configuração dos roteadores TROPIX para tal reserva.

2- IMPLEMENTAÇÃO DO AMBIENTE DE SERVIÇOS DIFERENCIADOS

Esta seção apresenta a implementação do ambiente de serviços diferenciados, que está baseado em roteadores com sistema operacional TROPIX e desenvolvido 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.

Para a construção deste ambiente foi necessário a criação de novas políticas de escalonamento de pacotes e condicionadores de tráfego, implementadas no núcleo do sistema operacional. 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 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 o aplicativos desenvolvido para fazer tais configurações.

2.1 - 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.

2.1.1- 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* [6]. 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 2 ilustra um ITBLOCK.

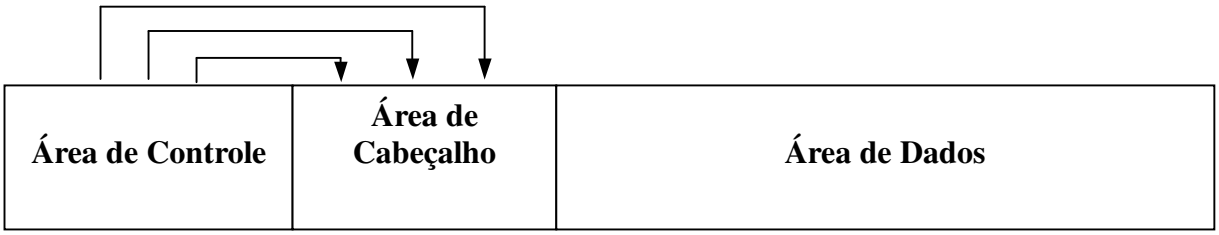


Figura 4.1 – Estrutura de um ITBLOCK

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.

2.1.2- 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. A estrutura de dados para o primeiro tipo de placa é a estrutura EDSTATUS e para o segundo é a estrutura RTL.

2.2- Módulo Escalonador

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 filas "Priority Queuing" (PQ), "Deficit Roudin Robin" (DRR) e "Start-Time Fair Queuing" (STFQ). Além da criação destas novas disciplinas de fila no núcleo do TROPIX, houve a reformulação da disciplina de filas FIFO que já estava implementada.

Todas as funções referentes às disciplinas de filas, inclusive a FIFO, foram postas em um arquivo de cabeçalho chamado *netqueue.c*, sendo as suas definições postas no arquivo de cabeçalho *netqueue.h*.

2.2.1- Reestruturação das Chamada às Funções de Enfileiramento e Desinfileiramento

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, tanto a codificação da rotina de enfileiramento dos pacotes quanto o desinfileiramento estava misturada ao 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. Foram criadas duas funções: uma para enfileirar os pacotes e outra para desinfileirá-los. Cada disciplina de fila tem as suas próprias funções de enfileiramento e desinfileiramento. Por exemplo, as novas funções de enfileiramento e desinfileiramento são respectivamente *enqueue_fifo* e *dequeue_fifo.* Todas estas funções encontram-se no arquivo *netqueue.c*

Como cada dispositivo de rede de um roteador tem uma disciplina de escalonamento de pacotes associada, que pode ser modificada dinamicamente, a chamada às funções de enfileiramento e desinfileiramento é feita utilizando-se o endereço das funções.

A Figura 3 mostra o diagrama de blocos da estrutura lógica de enfileiramento e desinfileiramento de pacotes implementada na camada de interface do Núcleo do TROPIX.

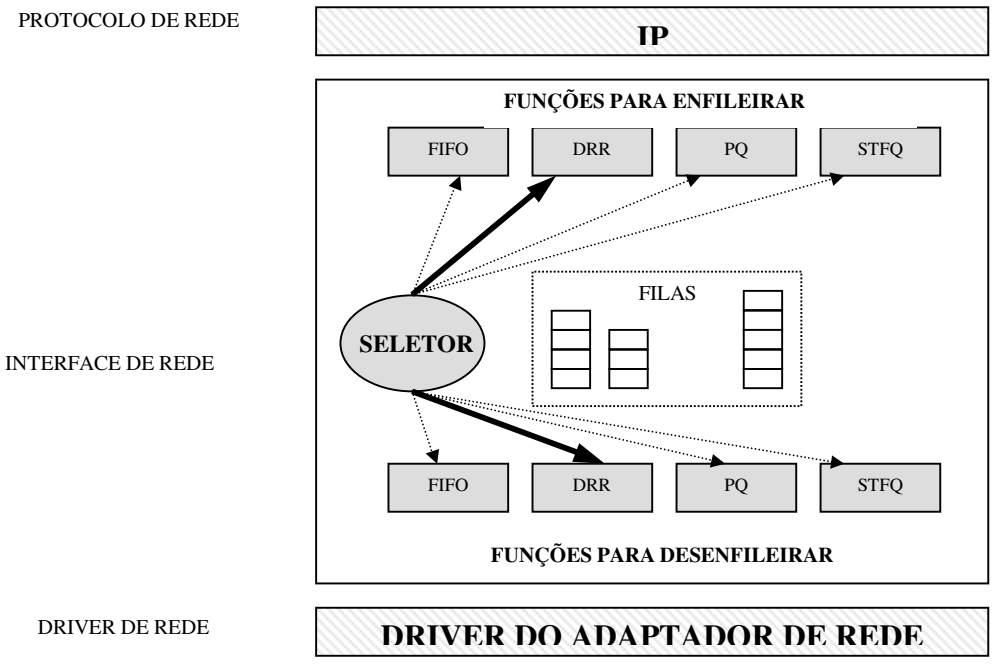


Figura 4.2 - Diagrama da estrutura lógica de enfileiramento e desinfileiramento de pacotes

Uma importante característica da implementação do módulo escalonador é a possibilidade da troca dinâmica das disciplinas de filas (com o roteador em funcionamento). A implementação também possibilita o uso de diferentes disciplinas de filas simultaneamente (cada placa de rede do roteador pode ter uma disciplina de fila diferente das demais placas).

2.2.2- 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 filas 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 seguir é apresentada a estrutura *NETQUEU*:

```
LOCK  q_lock;     /* Semáforo da estrutura */
QDISC discipline; /* Disciplina utilizada nesta unidade */

/* Endereços da função de enfileiramento */
void (*enqueue) (NETQUEUE *, ITBLOCK *);

/* Endereços da função de enfileiramento */
ITBLOCK  *(*dequeue) (NETQUEUE *);

EVENT q_no_packets; /* Indica que as filas estão vazias */
int   q_count;      /* Número de pacotes pendentes */
int   q_rnumber;    /* Round number da disciplina STFQ */

IDQUEUE queue[NUM_QUEUES]; /* Filas de pacotes */
IDQUEUE *cur_queue;        /* Endereço da última fila utilizada  */
```

A definição da estrutura *IDQUEUE* pode ser vista a seguir:

```
int     q_def_count; /* Contador de bits em deficit */
int     q_weight;    /* Peso relativo da fila */
int     q_quantum;   /* Quantidade de bits servida por vez */
int     q_cfnumber;  /* Conection finish number */
ITBLOCK *q_first;    /* Aponta para o primeiro pacote da fila */
ITBLOCK *q_last;     /* Aponta para o último pacote da fila */
```

O tipo *QDISC* é uma estrutura de enumeração que define as disciplinas de escalonamento implementadas no ambiente Diffserv do TROPIX. A sua definição é a que segue:

```
typedef     enum
{
      FIFO,             /* Disciplina FIFO */
      DRR,              /* Disciplina DEFICIT ROUND ROBIN */
      STFQ,             /* Disciplina START TIME FAIR QUEUING */
      PQ,               /* Disciplina PRIORITY QUEUING */

}     QDISC;
```

Algumas outras informações necessárias às disciplinas de filas foram definidas. Estas informações podem ser vista a seguir.

```
#define     NUM_QUEUES  4     /* Numero de filas */
#define     QUANTUM     128   /* Quantidade mínima de bits servido
                                 por vez */
#define     PNETQUEUEOUT      PTTYOUT     /* Situação de erro */
```

Todas estas estruturas e definições foram agrupadas em um arquivo de cabeçalho independente, chamado *netqueue.h*.

Os dois novos campos necessários a disciplina de fila STFQ acrescidos à estrutura de dados ITBLOCK podem ser visto a seguir:

```
int it_snumber;              /* Start number */
int it_fnumber;              /* Finish number */
```

2.2.3- Implementação das Disciplinas de Filas

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 a 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 enfileiramento 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 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. A função de enfileiramento é chamada *enqueue_drr* e a de desinfileiramento é chamada *dequeue_drr*.

- **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 calculo 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*.

A função de enfileiramento chama-se *enqueue_stfq* e a de desinfileiramento chama-se *dequeue_stfq*. 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, chamando a rotina *put_inorder_stfq*, definida em *netqueue.c*.

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

2.3- 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 ("Behaviour-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. Estas estruturas de dados estão definidas no arquivo de cabeçalho *diff_data.h*.

Nos roteadores que fazem a classificação MF o descritor, denominado *STMF_DATA*, possui a seguinte estrutura de dados:

```
IPADDR stmf_snd_addr;     /* IP do host que está enviando */
IPADDR stmf_rcv_addr;     /* IP do host que está recebendo */
int    stmf_snd_port;     /* Port do host que está enviando */
int    stmf_rcv_port;     /* Port do host que está recebendo */
int    stmf_update_time;  /* Tempo (em ms) entre atualizações */
int    stmf_token_size;   /* Tamanho do token de incremento */
int    stmf_bucket_depth; /* Profundidade do balde */
int    stmf_token_quant;  /* Quantidade em bytes de token */
int    stmf_codepoint;    /* Valor do codeponit que indentifica
                             do tipo de servico */
```

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

```
int         stba_update_time;  /* Tempo (em ms) entre atualizações */
int         stba_token_size;   /* Tamanho do token de incremento */
int         stba_bucket_depth; /* Profundidade do balde */
int         stba_token_quant;  /* Quantidade em bytes do token */
int         stba_codepoint;    /* Valor do codeponit que indentifica
                                  o tipo de servico */
```

Sempre que o rotedor é inicializado é criada uma lista de descritores livres, onde sua quantidade é definida pela variável `MAX_STMF` ou `MAX_STBA`, que respectivamente indica o número máximo de descritores MF e BA. Toda vez que uma reserva para um fluxo ou agregado é requisitada, é retirado um descritor da lista de livres e posta em uma tabela. Se for um roteador com classificador MF, o descritor é posto em uma tabela hash, pois pode haver um grande número de estados de reserva (um para cada fluxo com reserva). Nos roteadores BA a tabela é implementada por meio de um vetor, pois há um número fixo e reduzido de estados de reserva (um para cada tipo de serviço).

A função utilizada para acesso à tabela HASH nos roteadores MF é definida a seguir:

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

A função hash é bastante simples em nosso protótipo, pois utiliza apenas os endereços IP origem e destino. Mas há uma versão que pode-se usar a quadrupla: IP origem, IP destino, Porta origem e Porta destino.

Caso o roteador tenha o classificador BA, a tabela é um vetor que possui tantas entradas quantos forem as classes de serviços diferenciados. As funções de gerenciamento dos descritores de reserva estão definidas em *diff_mf.c* e *diff_ba.c*.

2.4- 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 não foi implementado, visto que o ambiente Diffserv descrito neste artigo foi desenvolvido para tráfego IP "multicast". Este tipo de tráfego utiliza o protocolo de transporte UTP, que não é sensível ao descarte.

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 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 baldes de token ("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 pelos campos *stmf_token_size* ou *stba_token_size*.

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". A taxa de chegada dos "tokens" é definida pelos campos *stmf_update_time* e *stba_update_time*. Este tempo 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 pela função *toutset* para se repetir com este período fixo de "ticks" do relógio.

Quando a função *stmf_update* for ativada, ela se reprograma para ser ativada novamente segundo o mesmo número de "ticks". Há uma função equivalente para os descritores BA, chamada *stba_update*. 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).

2.5- 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.

2.5.1- Funções de Configuração

Visando uma maior eficiência, as funções que implementam o ambiente de serviços diferenciados estão localizadas no núcleo do sistema operacional. Para acessá-las, foram especificadas novas chamadas à função 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 deleção 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 a seguir:

FUNÇÃO	DESCRIÇÃO
NET_GET_DISCIPLINE	Obtêm 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 “Behaviour Aggregate”.
I_ROUTER_TYPE	Exclui um estado descritor de reserva “Behaviour Aggregate”.

2.5.2- Aplicativos de Configuração

Um servidor de configurações foi desenvolvido como aplicativo de usuário para fazer as diversas chamadas à IOCTL. Ele é executado no roteador TROPIX e é chamado *sconfig.c*. A sintaxe para a sua execução é:

sconfig [-v] porta tipo-de-roteador

A opção *–v* coloca o configurador no modo “verbose”, que apresenta na tela as informações de configuração que estão sendo executadas. A *porta* indica em que porta o aplicativo é acessado. E *tipo-de-roteador* define se a configuração será para um roteador do tipo “multi-field” ou “behaviour-aggregate”. A seguir é apresentada uma tabela com as diversas opções de configuração dos roteadores:

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.
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 “behaviour-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).
LD	Lista os dispositivos de rede	Lista os dispositivos de rede que estão ativos no roteador.

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 bastante simples foi desenvolvido 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- IMPLEMENTAÇÃO DO GERADOR 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. Além disso, os geradores podem servir como ferramentas auxiliares na geração de pacotes em uma rede a fim de simular um determinado tráfego de dados segundo modelos matemáticos. Existem diversos geradores de tráfego disponíveis, entre eles o Netperf [21] e o TGM [22].

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.

A seção 3.1 deste artigo apresenta o ambiente para a geração de tráfego. Na seção 3.2, são discutidos os detalhes da implementação do gerador de tráfego e as otimizações que permitem uma maior precisão nas medidas. Na seção 3.3 são apresentadas as diversas versões do gerador e suas aplicações. Na seção 3.4, são mostrados alguns resultados comparativos.

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, conforme pode ser visto na Figura 1.

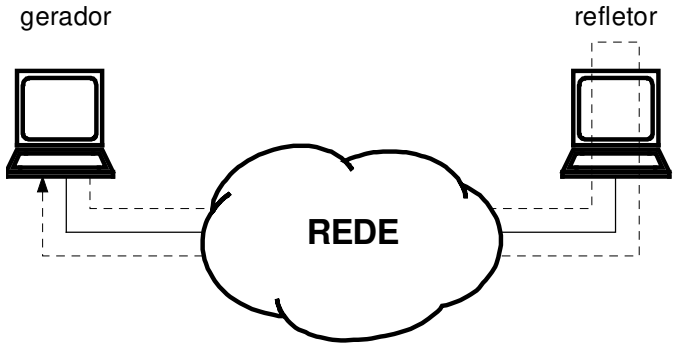


Figura 1 – Ambiente de Geração de Tráfego

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.

3.1.1- Módulo Transmissor

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. Para a geração dos pacotes, são necessárias as seguintes informações: o endereço IP do computador executando o aplicativo refletor, a taxa de transmissão, o tamanho do pacote e o número de pacotes que devem ser gerados. A taxa de transmissão do enlace, por onde os dados serão transmitidos, é pré-definida, podendo ser de 10 Mbps ou 100 Mbps.

Os parâmetros necessários para a geração dos pacotes são calculados com base nestas informações. Primeiro é calculado o número de pacotes que devem ser transmitidos em cada segundo; depois é calculado o tempo de transmissão de cada pacote e o intervalo de tempo entre eles para obter a taxa de transmissão desejada.

Antes da transmissão de cada pacote, é feita a leitura do relógio de tempo real, marcando o horário corrente da transmissão. Com esta informação, é calculado o momento da transmissão do próximo pacote, resultado da soma do horário corrente, do tempo de transmissão de um pacote e do intervalo entre pacotes. O pacote a transmitir é rotulado com o horário da transmissão (horário de saída do gerador). Após o envio, o módulo aguarda o instante da próxima transmissão. Esta espera foi implementada com "espera ocupada", evitando os atrasos decorrentes do reescalonamento da *thread* transmissora.

O algoritmo se encerra uma vez atingido o número de pacotes definido pelo usuário.

3.1.2- Módulo Receptor

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.

Assim que um pacote retorna, o módulo receptor faz a leitura do relógio de tempo real e calcula o atraso sofrido pelo pacote na rede, utilizando uma rotina apropriada. Os detalhes do cálculo do tempo gasto pelo pacote serão visto mais adiante. Quando o limite de pacotes recebidos atinge o seu número máximo, definido pelo usuário, uma função de impressão das estatísticas é executada.

3.1.3- Refletor

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.

3.2- A Implementação

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

3.2.1- 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.

As seguintes informações são necessárias para a transmissão de um pacote: o endereço de destino, o número do protocolo e os dados que devem ser transmitidos. Estas informações são diretamente utilizadas no cabeçalho IP e nenhuma sobrecarga de controle é necessária na área de dados do pacote.

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, conforme veremos na seção 3.4.

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;
- caracteres de preenchimento.

A Figura 3 mostra o formato do pacote IP gerado.

CABEÇALHO IP ÁREA DE DADOS IP

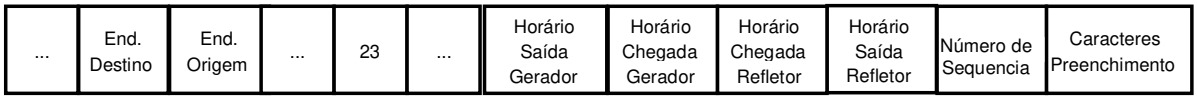


Figura 3 – Formato de um pacote gerado

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. Estes dois sub-campos são do tipo inteiro longo, com quatro octetos cada um.

3.2.2- Gerando Pacotes de Tamanhos Variados

Normalmente, os geradores de tráfego produzem pacotes com tamanhos fixos. Para melhor simular o tráfego na Internet, ou em qualquer rede de dados, criamos uma versão do gerador que permite o uso de tamanhos variados de pacotes.

A geração de pacotes com tamanhos completamente variados cria dificuldades para o cálculo dos parâmetros utilizados no algoritmo de geração dos pacotes, pois eles dependem da definição “a priori” do tamanho do pacote. Por exemplo, o cálculo do número de pacotes que devem ser transmitidos a cada segundo, para manter a taxa de transmissão, depende do tamanho do pacote. O intervalo de tempo entre os pacotes depende, por sua vez, do número de pacotes. Estes cálculos podem ser vistos a seguir:

```
número_pacotes = taxa_transmissao / (8 * tamanho_pacote)

intervalo_entre_pacotes = (1000000 - tempo_transmitindo_pacotes) /
                                    número_pacotes
```

A fim de solucionar esta dificuldade de cálculo do número de pacotes e todos os demais parâmetros, foi utilizado um tamanho médio de pacotes, fixado em 782 octetos, que é a média entre 64 e

1500 (tamanho mínimo e máximo de um pacote Ethernet). O algoritmo de transmissão é similar ao anterior: todo pacote com tamanho x, aleatoriamente gerado entre 64 e 1500, é seguido de um pacote complementar, com tamanho 1564 - x. Este procedimento mantém a média dos tamanhos dos pacotes em 782.

Mantendo o tamanho médio dos pacotes, o intervalo entre os pacotes pode ser calculado com relação a este tamanho e não com relação aos diferentes tamanhos dos pacotes gerados. Esta abordagem é simples, mas permite uma grande variabilidade do tamanho dos pacotes, criando um ambiente de teste mais adequado.

3.3- 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.

A primeira solução que tentamos foi o aumento da prioridade das *threads* do gerador e a diminuição da prioridade dos *daemons* do sistema, além de desativar o procedimento periódico de atualização do cache de disco. Isto amenizou, mas ainda não resolveu o problema, pois permaneceu a disputa entre as *threads* transmissora e receptora. Muitos geradores são executados como aplicações de tempo-real, o que também não soluciona por completo o problema apresentado anteriormente.

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 (ITBLOCK). 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.

3.4- Uso dos Geradores

Diversas versões do aplicativo gerador foram criadas, possibilitando testes comparativos sobre a precisão das medidas, que poderão ser vistos na próxima seção. A seguir é apresentada uma tabela indicando os nomes dos geradores, os seus parâmetros de chamada e a sua utilização.

Gerador de Tráfego de Pacotes de Tamanho Constante	
gtc_drv *end_destino taxa_transmissão número_pacotes tamanho_pacotes*	Gerador de tráfego, módulo transmissor e receptor, fazendo amostragem de tempo no driver.
gtc_flood *end_destino taxa_transmissão tamanho_pacotes*	Gerador de tráfego, módulo transmissor, fazendo transmissão contínua de pacotes sem amostragem de tempo.
gtc_usr *end_destino taxa_transmissão número_pacotes tamanho_pacotes*	Gerador de tráfego, módulo transmissor e receptor, fazendo amostragem de tempo no aplicativo.
Gerador de Tráfego de Pacotes de Tamanho Aleatório	
gta_drv *end_destino taxa_transmissão número_pacotes tamanho_pacotes*	Gerador de tráfego, módulo transmissor e receptor, fazendo amostragem de tempo no driver.
gta_flood *end_destino taxa_transmissão tamanho_pacotes*	Gerador de tráfego, módulo transmissor, fazendo transmissão contínua de pacotes sem amostragem de tempo.
gta_usr *end_destino taxa_transmissão número_pacotes tamanho_pacotes*	Gerador de tráfego, módulo transmissor e receptor, fazendo amostragem de tempo no aplicativo.
Refletor	
refletor *num_protocolo número-pacotes*	Refletor de pacotes. Quando o número de pacotes é igual a zero o refletor é executado continuamente.

Como pode ser visto na tabela, há duas classes de geradores: a primeira é a de geradores de pacotes com tamanhos constantes, chamados de *gtc*; a segunda é a de geradores de pacotes com tamanhos aleatórios, chamados de *gta*.

Os aplicativos *gtc_drv* e *gta_drv* ativam os módulos transmissores e receptores como *threads*. O *gtc_flood* e o *gta_flood* são geradores de tráfego ilimitado, que podem ser utilizados como "pertubadores" da rede, apenas servindo para aumentar o tráfego. O *gtc_usr* e o *gta_usr* possuem as mesmas funcionalidades do *gtc_drv* e *gta_drv*, mas se diferenciam por fazerem a amostragem do tempo no aplicativo, estando sujeito aos problemas de escalonamento do sistema operacional.

3.5- Resultados

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 [22], 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*.

3.5.1- 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 apresenta o ambiente de teste.

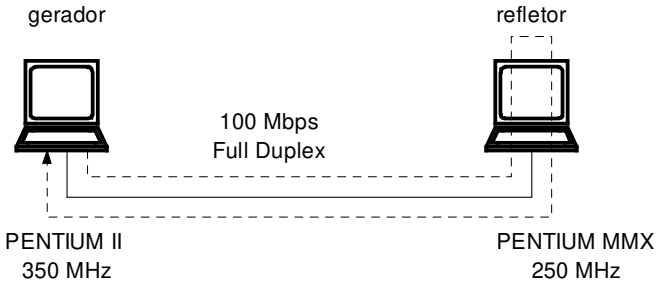


Figura 4- Ambiente de Teste

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

3.5.2- 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.

- **O 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 a seguir, apresentamos os valores médios obtidos nas dez rodadas de medição, para cada aplicativo.

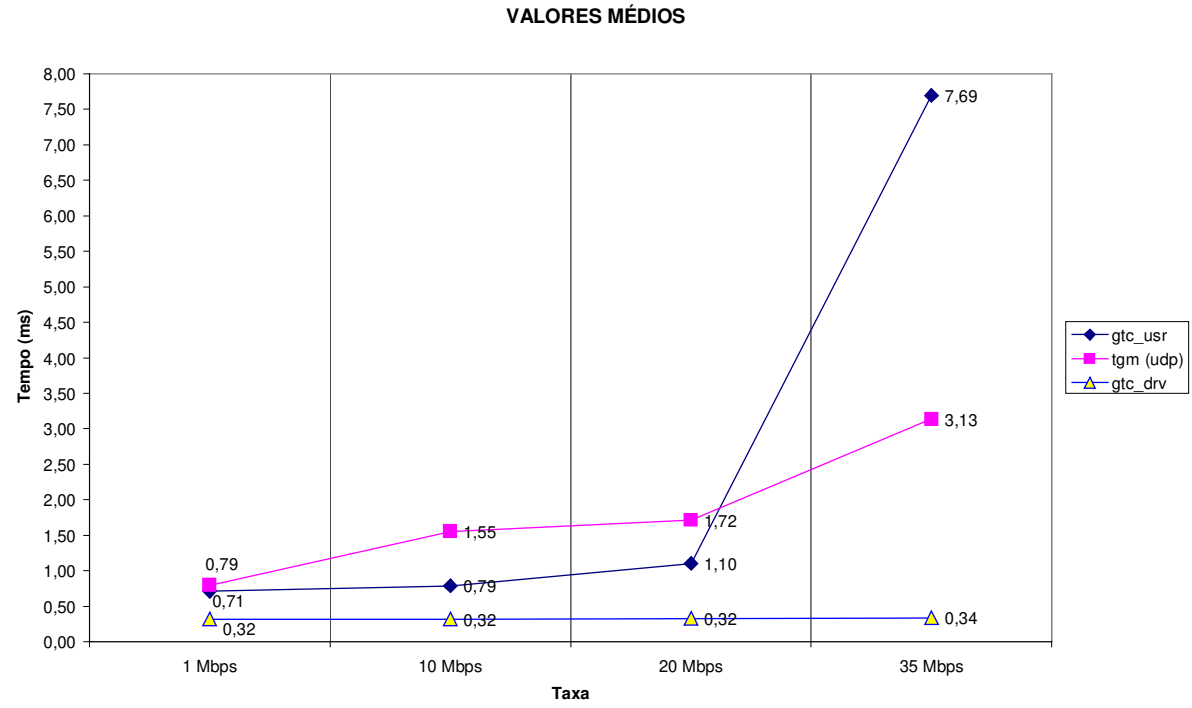


Figura 5 – Gráfico da Média dos Tempos de *Round Trip* para Diferentes Taxas de Transmissão.

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.

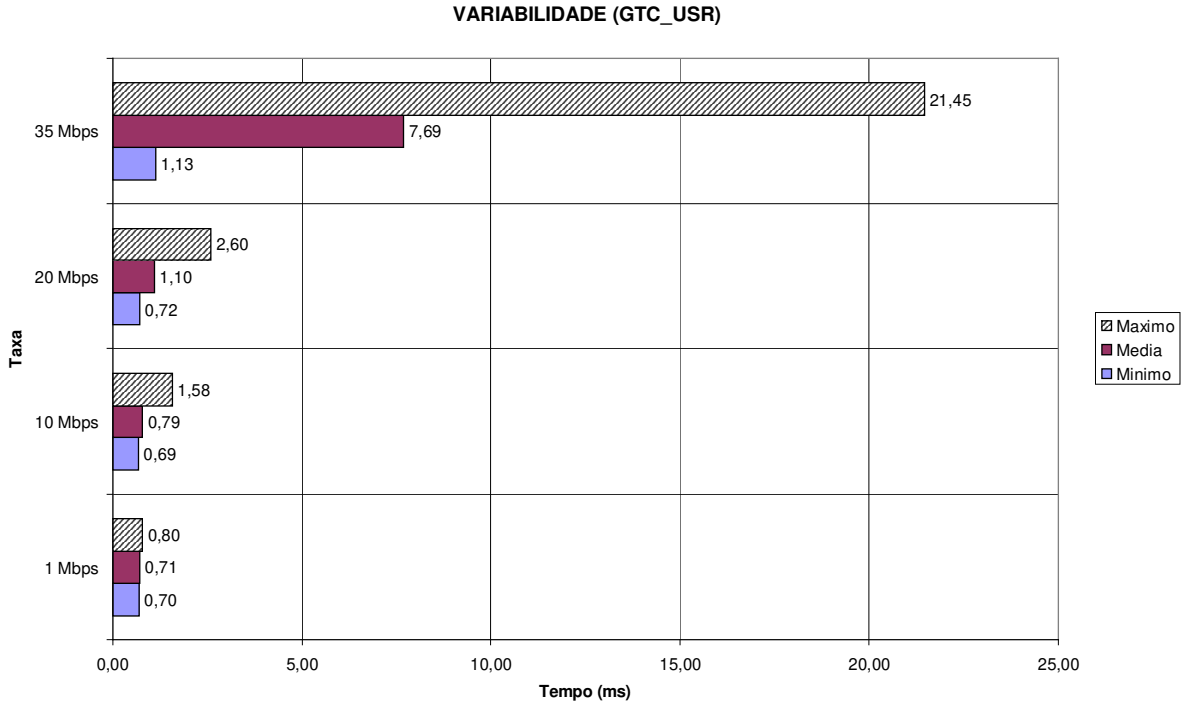


Figura 6 – Variabilidade dos Tempos de Round Trip no Aplicativo *GTC_USR* para Diferentes Taxas de Transmissão.

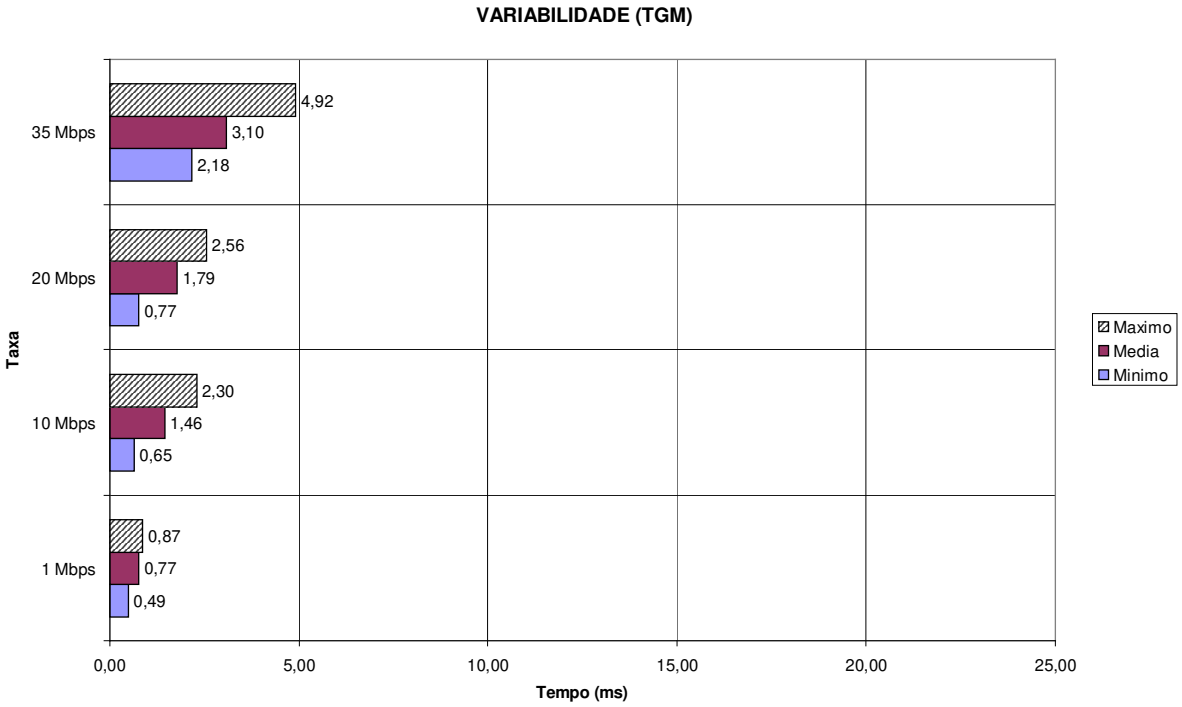


Figura 7 – Variabilidade dos Tempos de Round Trip no Aplicativo *TGM* para Diferentes Taxas de Transmissão.

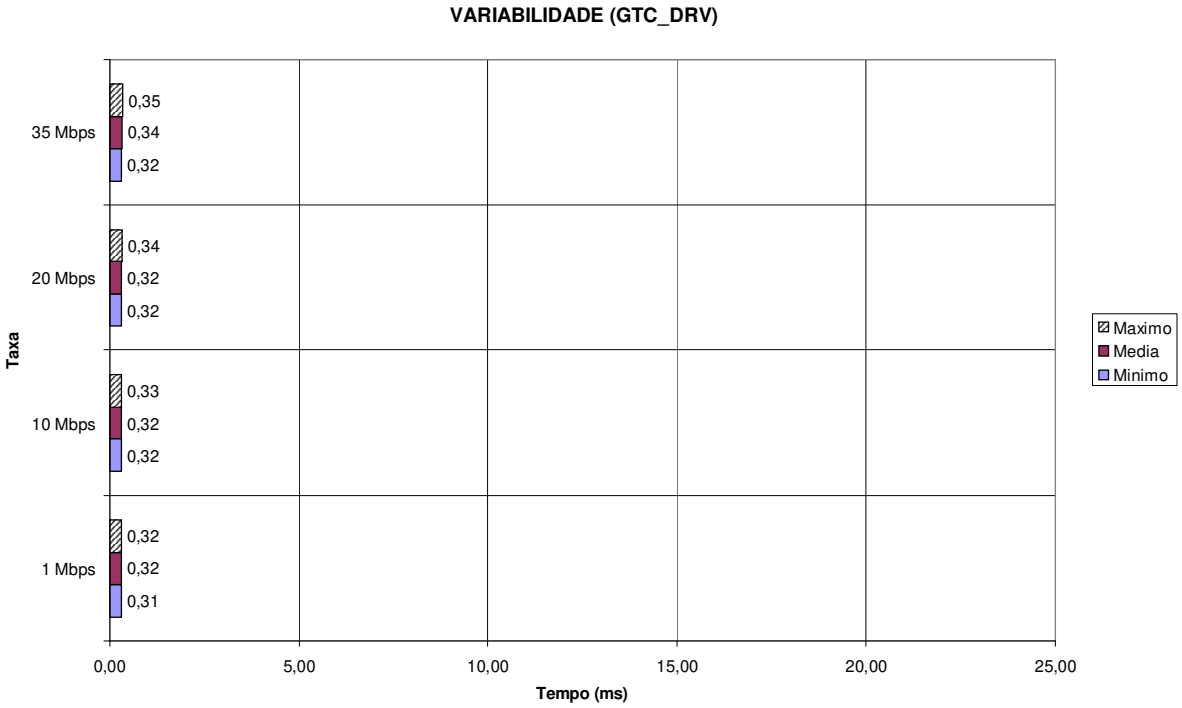


Figura 8 – Variabilidade dos Tempos de Round Trip no Aplicativo *GTC_DRV* para Diferentes Taxas de Transmissão.

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.

- **O 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.

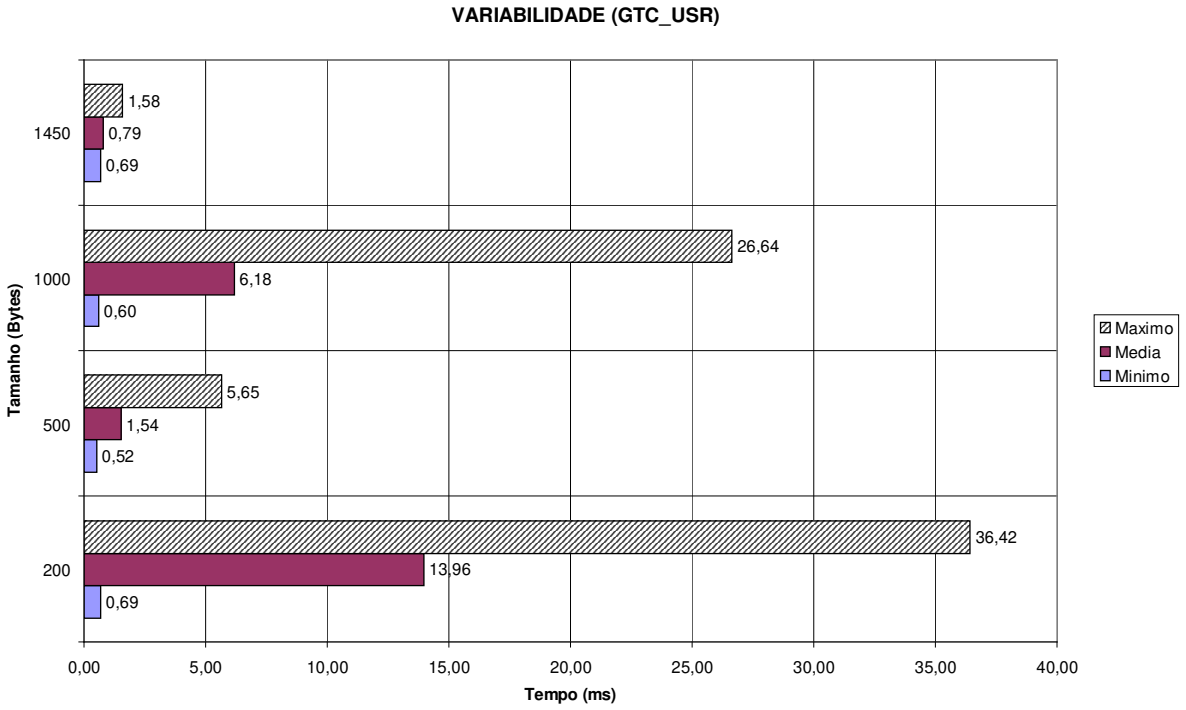


Figura 9 – Variabilidade dos Tempos de *Round Trip* no Aplicativo *GTC_USR para Diferentes Tamanhos de Pacotes*

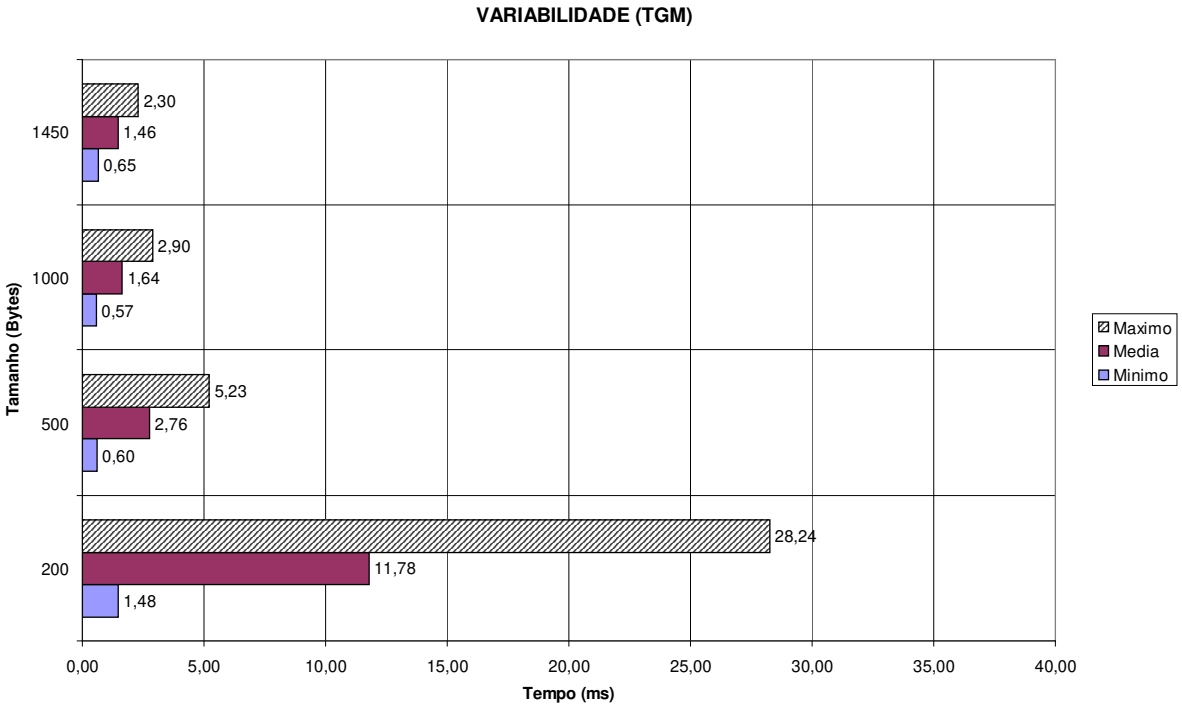


Figura 10 – Variabilidade dos Tempos de *Round Trip* no Aplicativo *TGM para Diferentes Tamanhos de Pacotes*

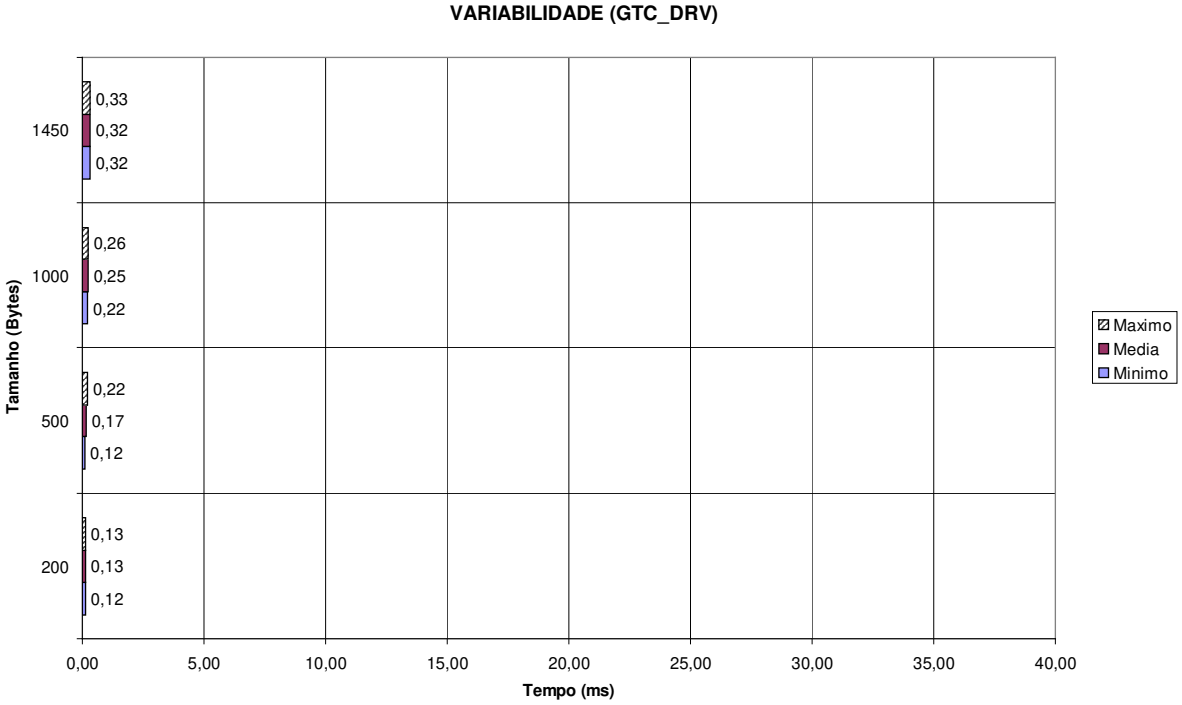


Figura 11 – Variabilidade dos Tempos de *Round Trip* no Aplicativo *GTC_DRV para Diferentes Tamanhos de Pacotes*

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. Sistema de Reserva de Recursos

Um protótipo dos componentes 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.

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.

Figura 7 - Tela do Agente para Requisição de Reservas

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. A Figura 7 mostra a tela do Agente para Reserva com todas as informações necessárias para a caracterização do perfil/reserva de uma sessão.

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 seguir é apresentada uma tabela com as diversas ações especificadas para o protocolo:

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.

CONCLUSÃO

A arquitetura descrita neste trabalho possibilita o tráfego “multicast” e a gerência dos seus recursos de rede no modelo Diffserv, permitindo um bom desempenho das aplicações multimídia, com as de teleconferências de vídeo/áudio e as de recuperação de documentos multimídia pela Internet.

Tal arquitetura baseia-se na verificação antecipada da disponibilidade de recursos e a sua reserva nos ramos da árvore de distribuição de dados de uma sessão “multicast”, sendo 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 “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 de redes nos domínios de serviços diferenciados para cada sessão “multicast”. Além disso, os SRRs são os responsáveis pela configuração dos roteadores destes domínios.

Os roteadores implementam o serviço “multicast Premium” através do PHB “Expedited Forwarding” e implementam as seguintes disciplinas de enfileiramento: “Priority Queueing”, “Deficit Round-Robin” e “Start-Time Fair Queueing”. O condicionamento do tráfego utiliza o algoritmo do balde de permissões (“Token Bucket”).

Dois tipos de protocolos "multicast" são usados: um intradomínio e outro interdomínio. O protocolo interdomínio usado é o MASC/BGMP e o protocolo intradomínio é o DVMRP. Na etapa atual de desenvolvimento deste ambiente só é possível a comunicação entre os SRRs e roteadores dentro de um mesmo domínio. Como trabalho futuro, será implementado o protocolo MASC/BGMP a fim de possibilitar o tráfego “multicast” interdomínio.

A comunicação entre os elementos da arquitetura está ocorrendo por intermédio de um protocolo proprietário. A próxima etapa será a implementação de um protocolo de comunicação padronizado, como o RSVP.

Além disso, contrastamos duas abordagens para implementação de aplicativos geradores de tráfego, objetivando medidas de desempenho do ambiente de serviços diferenciados implementado.

A abordagem convencional realiza amostragens de tempo na 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 proposta, 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.

Como resultado, produzimos um conjunto de ferramentas para geração de tráfego que executam sobre o sistema operacional TROPIX, também desenvolvido no NCE/UFRJ, cujo núcleo foi adequadamente alterado para suportar os pacotes do protocolo específico que criamos para a geração. Destacamos igualmente a ferramenta desenvolvida que gera pacotes com tamanhos aleatórios, produzindo um tráfego mais semelhante ao que ocorre em uma rede real. 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 propomos, em experimentos onde taxa de transmissão e tamanhos de pacotes são exaustivamente variados.

REFERÊNCIAS BIBLIOGRÁFICAS

[01] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, Junho, 1994

[02] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Dezembro 1998.

[03] V. Jacobson, "Congestion Avoidance an Control", ACM SIGCOMM'88, Agosto 1988

[04] S. Berson, S. Vicent, "Aggregation of Internet Integrated Services State", Internet Draft, Agosto 1998.

[05] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", Work in Progress, ftp://ftp.ee.lbl.gov/papers/ dsarch.pdf.

[06] S. Berson, R. Lindell, R. Braden, "An Architecture for Advance Reservations in the Internet", Julho, 1998.

[08] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, Dezembro de 1998.

[09] Information Sciences Institute, "Internet Protocol", RFC 791, Setembro 1981.

[11] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Fowarding PHB Group", RFC 2597, Junho 1999.

[12] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Fowarding PHB", RFC 2598, Junho 1999.

[14] R. Bless and K. Wehrle. A Lower Than Best-Effort Per-Hop Behavior. Internet-Draft – draft-bless-diffserv-lbe-phb-00.txt, Setembro 1999.

[17] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, "Recommendation on Queue Management and Congestion Avoidance in the Internet", RFC 2309, Abril 1998.

[21] http://www.netperf.org

[22] M. Leocádio, P. H. Aguiar, "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.

[23] http://www.tropix.nce.ufrj.br/

[24] M. K. McKusick, K. Bostic, M. J. Karels, J. S. Quarterman, "The Design and Implementation of the 4.4 BSD Operating System", Addison-Wesley, 1996.

[30]] COMER, Douglas E.. Interligação em Rede com TCP/IP, vol 2. Rio de Janeiro: Campus, 1999.

[31] KESHAV, Srinivasan. An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network. Addison-Wesley, 1998.