DEPARTAMENTO DE ENGENHARIA E ARQUITETURA
DEPARTAMENTO DE CIÊNCIAS EXATAS E DE COMUNICAÇÃO
MESTRADO EM REDES DE COMPUTADORES

Sílvio Fernando Lima de Santana

Proposta de Referência para Projetos de Qualidade de Serviço (*QoS*) em Redes Corporativas

Salvador
2006

DEPARTAMENTO DE ENGENHARIA E ARQUITETURA
DEPARTAMENTO DE CIÊNCIAS EXATAS E DE COMUNICAÇÃO
MESTRADO EM REDES DE COMPUTADORES

Sílvio Fernando Lima de Santana

Proposta de Referência para Projetos de Qualidade de Serviço (*QoS*) em Redes Corporativas

Dissertação apresentada à Universidade Salvador, como parte dos requisitos para a obtenção do título de Mestre em Redes de Computadores.

Orientador:
Prof. Dr. Joberto Sérgio Barbosa Martins

Salvador
2006

FICHA CATALOGRÁFICA
(Elaborada pelo Sistema de Bibliotecas da Universidade Salvador - UNIFACS)

Santana, Sílvio Fernando Lima de
Proposta de referência para projetos de qualidade de serviço (QoS) em redes corporativas / Sílvio Fernando Lima de Santana. - 2006.
185 f.

Dissertação - (mestrado) - Universidade Salvador – UNIFACS, Mestrado em Rede de Computadores, 2006.

Orientador: Prof. Dr. Joberto Sérgio Barbosa Martins.

1. Redes de computação. 2. Redes corporativas. 3. Análise de redes (Planejamento). 4. Redes de computação – qualidade dos serviços. I. Martins, Joberto Sérgio Barbosa, orient. II. Título.

CDD: 004.6

DEPARTAMENTO DE ENGENHARIA E ARQUITETURA
DEPARTAMENTO DE CIÊNCIAS EXATAS E DE COMUNICAÇÃO
MESTRADO EM REDES DE COMPUTADORES

SÍLVIO FERNANDO LIMA DE SANTANA

Proposta de Referência para Projetos de Qualidade de Serviço (*QoS*) em Redes Corporativas

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Redes de Computadores, Universidade Salvador – UNIFACS, do Curso de Mestrado em Redes de Computadores, pela seguinte banca examinadora:

Prof. Joberto Sérgio Barbosa Martins - Orientador
Doutor em Informatique, Universite de Paris VI (Pierre et Marie Curie), U.P. VI, França.
Universidade Salvador - UNIFACS

Prof. José Augusto Suruagy Monteiro
Doutor em Computer Scince, University of Califórnia Los Angeles, U.C.L.A, Estados Unidos.
Universidade Salvador - UNIFACS

Prof. Ricardo José Paiva de Britto Salgueiro
Doutor em Ciências da Computação. Universidade Federal de Pernambuco, UFPE, Brasil.
Universidade Federal de Sergipe- UFS

Salvador, 25 de maio de 2006

DEDICATÓRIA

Com todo amor, dedico esse trabalho a minhas filhas, Fernanda e Isabella, fonte de inspiração e razão maior da minha vida. Nanda, primeira benção, amiga e companheira, sabemos e sentimos que estamos juntos há muitas e muitas vidas. Bella, segunda dádiva, minha pequena guerreira, nos mostrou a grandeza do poder de Deus e a força do amor, nos ensinou como se luta e nos fez ver que a vida sempre vale a pena.

"...que a luz do nosso amor fique a brilhar.
É pra você e pra todo mundo que quer trazer
A paz no coração
Meu pequeno amor
De você me lembrar
Toda vez que a vida mandar olhar pro céu
Estrela da manhã
Meu pequeno grande amor
Pra poder ser livre como a gente quis
Quero te ver feliz ..." (Beto Guedes)

A minha mãe, Vera, pelo amor, pela dedicação à nossa criação no caminho do bem, pelo exemplo de coragem e de força e por ter nos feito ver que sem estudo nunca chegaríamos. Cada um de nós lhe deve o que somos.

A minha esposa, Silvinha, pelo amor, carinho, paciência e apoio. Só Deus saberia explicar quando começou a nossa história.

A meus irmãos, Sérgio, Silvana, Sandro e Lucas, pelo amor e pelo exemplo de retidão de princípios e de caráter. É muito bom estarmos sempre juntos e poder dizer o quanto me orgulho.

A minha inesquecível avó Leandra, a meus tios e tias, de quem só recebi amor. O tempo passou, alguns de nós já precisou partir e a saudade é imensa. O amor jamais permitirá esquecê-los.

A meus primos queridos, todos eles, mas, especialmente, Iulo, por tantos e tantos momentos de afeto e alegria que nos fizeram amigos e irmãos; e Marcelo, "meu Téo", a quem tanto admiro pela pessoa maravilhosa, pelo exemplo de filho que é e por cuidar com tanto amor de um dos grandes amores da minha vida, minha tia Bahia.

A meus sogros, Silvio e tia Marina, por me acolherem como filho, com tanto amor, carinho e confiança.

Aos amigos de hoje e de sempre, que me acompanham desde os tempos de escola e de música, e que tanto enriquecem a minha história.

Cada um de vocês está em cada momento dos meus dias, em tudo o que faço e em tudo o que sou. Vocês estão em mim. Graças a Deus.

AGRADECIMENTOS

Agradeço a Deus, Nosso Senhor, por me proporcionar uma passagem tão maravilhosa, de tanta felicidade e em uma família de amor. Por me ensinar a valorizar o "ser" ante o "ter" e o "sentir" ante o "saber". A Ele, a Quem não me sinto no direito de mais nada pedir, somente agradecer, e a todos os espíritos de luz que me acompanham, pela oportunidade da realização de mais uma etapa.

A minha esposa e filhas pela compreensão e paciência, sobretudo nos momentos de dificuldade e aflição vividos durante esse trabalho.

À Compugraf por ter financiado parte desse projeto e pelos anos de trabalho tão enriquecedores que vivemos juntos.

Às demais empresas por onde passei nesses já não tão poucos anos de carreira: Telebahia, Banco Excel/BBV, IBM Brasil e Petrobras, por tanto contribuírem na minha formação profissional.

À Unifacs, Faculdades Jorge Amado e Qualify por me darem, como professor, a oportunidade de partilhar um pouco da experiência acumulada; e aos meus alunos, que tanto me ensinam a cada dia.

Às empresas relacionadas aos Estudos de Caso, um deles descrito no trabalho, pela confiança depositada.

Aos amigos da Petrobras, Unidade de Telecomunicações, pela generosidade, apoio, carinho e compreensão. Agradeço a todos, sem exceção, mas, especialmente, a Alvina Timbó, Henrique Nápoli e Marcelo Barroso.

A meu irmão, Sérgio, pela valiosa ajuda na revisão do texto, pelo incentivo e pela simplicidade e humildade com que lida com seu imenso conhecimento e capacidade.

Ao meu orientador, Prof. Dr. Joberto Martins, à coordenação do mestrado, representada pelo Prof. Dr. Suruagy, aos demais professores e funcionários.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AF	*Assured Forwarding*
AQM	*Active Queue Management*
ATM	*Asynchronous Transfer Mode*
BA	*Behavior Aggregate*
Bc	*Burst Commited*
bps	*bits* por segundo
CAC	Controle de Admissão de Conexões
CAR	*Commited Access Rate*
CBQ	*Class-based Queueing*
CBWFQ	*Class-based Weighted Fair-Queueing*
CE	*Congestion Experienced*
CIM	*Common Information Model*
CIR	*Commited Information Rate*
CLP	*Cell Loss Priority*
COPS	*Common Open Policy Service*
CoS	*Class of Service*
CQ	*Custom Queueuing*
cRTP	*Compressed Real-Time Protocol*
CWND	*Congestion Window*
DE	*Dircard Elegibility*
DiffServ	*Differentiated Services*
DMTF	*Distributed Management Task Force*
DoS	*Denial Of Service*
DSCP	*Differentiated Services Code Point*
ECN	*Explicit Congestion Notification*
ECT	*ECN Capable Transport*
EF	*Expedited Forwarding*
ELSR	*Edge Label Switching Router*
ESP	*Encapsulating Security Payload*
EXP	*MPLS Experimental Field*
FIFO	*First-in first-out*
FoIP	*Fax over IP*
FQ	*Fair-Queueing*
FRF	*Frame Relay Forum*
IETF	*Internet Engineering Task Force*
IntServ	*Integrated Services*
IP	*Internet Protocol*
IPDV	*Inter-packet delay variation*
IPP	*IP Precedence*
IPX	*Internetwork Packet Exchange*
ISDN	*Integrated Services Digital Network*
ISP	*Internet Service Provider*
LAN	*Local Area Network*
LFI	*Link Fragmentation and Interleaving*
LLQ	*Low-latency Queueing*

LSP	*Label Switching Path*
LSR	*Label Switching Router*
MAC	*Medium Access Control*
MIB	*Management Information Base*
MPLS	*Multiprotocol Label Switching*
MSS	*Maximum Segment Size*
MTBF	*Mean Time Between Failures*
MTTR	*Mean Time to Repair*
MTU	*Maximum Transfer Unit*
NBMA	*Nonbroadcast Multiple Access*
OD	Origem-Destino
PBN	*Policy-based Network*
PBNM	*Policy-based Network Management*
PCIM	*Policy Core Information Model*
PCM	*Pulse Code Modulation*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PHB	*Per-hop Behavior*
PoP	*Point-of-Presence*
PPP	*Point-to-Point Protocol*
pps	Pacotes por segundo
PQ	*Priority Queueuing*
PVC	*Private Virtual Circuit*
QoS	*Quality of Service*
RED	*Random Early Detection*
RIO	*RED IN and OUT*
RSVP	*Resource Reservation Protocol*
RTCP	*Real-time Control Protocol*
RTP	*Real-time Protocol*
SLA	*Service Level Agreement*
SNA	*Systems Network Architecture*
SNMP	*Simple Network Management Protocol*
SPX	*Sequenced Packet Exchange Protocol*
TC	*Traffic Conditioning*
Tc	*Time Commited*
TCP	*Transmission Control Protocol*
TDM	*Time Division Multiplex*
ToS	*Type of Service*
UDP	*User Datagram Protocol*
VLAN	*Virtual Local Area Network*
VoIP	*Voice over Internet Protocol*
VPN	*Virtua Private Network*
WAN	*Wide Area Network*
WFQ	*Weighted-fair Queueing*
WRED	*Weighted Random Early Detection*
WRR	*Weighted Round Robin*
WWW	*World Wide Web*

RESUMO

Essa dissertação descreve uma abordagem para projetos de Qualidade de Serviço em redes *IP*. Com o objetivo de apresentar uma referência para novos projetos, foram estudados as arquiteturas e mecanismos de *QoS* disponíveis, bem como as ferramentas utilizadas na entrega da solução fim-a-fim. O modelo propõe um roteiro para a elaboração de projetos de integração para as novas redes convergentes, partindo da análise, caracterização e definição de requisitos das aplicações em rede. A partir desses requisitos, conhecidos como índices ou parâmetros da *QoS*, tanto os relacionados ao perfil próprio de funcionamento das aplicações, quanto os determinados pelo grau de criticidade contextual, inferidos pela importância da aplicação no ambiente analisado, são obtidos os valores de configuração das políticas. Com isso, pode-se definir os parâmetros de configuração para cada uma das etapas da implementação da *DiffServ*, arquitetura de *QoS* padrão recomendada para grandes redes. O modelo foi aplicado em um estudo de caso, no qual foi elaborado e implantado um projeto de *QoS*, integrando voz e aplicações de dados de diferentes características e prioridades. Essa utilização indicou que o uso de um modelo de referência simplifica as etapas do projeto, minimiza os riscos de erros de dimensionamento e de configuração e facilita a documentação da solução final.

Palavras chave: Projetos de Redes, Qualidade de Serviço, Serviços Diferenciados e Redes Convergentes.

ABSTRACT

This thesis describes an approach to QoS projects in IP-networks. In order to propose a reference to new projects, available QoS architectures and resources were studied, as well as tools used for delivery of the end-to-end solution. The model proposes a guideline for ellaborating integration projects for new converged networks. By means of these requirements, knows as QoS indexes or parameters, not only the ones related to the application itself, but also the ones related to the relevance of the application in its context, the values of policy configuration were obtained. As a result, it was possible to define the configuration parameters for each implementation phase of the DiffServ, which is the QoS standard recommended for large networks. The model was used in a case study, where a QoS project was developed for integrating voice and data applications with different characteristics and priorities. The case study demonstrated that the use of a reference model simplifies the phases of the project, reduces the risk of errors of sizing and configuration and facilitates the documentation of the final solution.

Key-words: Network Projects, Quality of Service, Differentiated Services and Converged Networks.

1 INTRODUÇÃO

1.1 Objetivos e considerações iniciais

Projetos tradicionais de redes de computadores normalmente são baseados em regras e considerações padrão, que variam basicamente de acordo com a abrangência da rede, com os serviços oferecidos e com os requisitos do projeto. Essas variáveis dizem respeito às exigências quanto aos parâmetros formais utilizados: disponibilidade, desempenho, escalabilidade[1], atraso, confiabilidade, gerenciabilidade[2], custo-benefício, nível de segurança da solução, dentre outros.

Contemplar, total ou parcialmente, cada um desses itens interfere diretamente no resultado final da solução, tanto do ponto de vista do atendimento aos requisitos do negócio, quanto no custo final do projeto.

Outro fator importante, concernente aos projetos de redes mais modernos, sobretudo aqueles com abordagem convergente, trata do atendimento ao tráfego de aplicações multimídia[3]. A caracterização dessas aplicações e o bom entendimento dos seus requisitos de funcionamento permitem a especificação correta de equipamentos e parâmetros de configuração, possibilitando o funcionamento satisfatório.

[1] Refere-se à capacidade de expansão sem necessidade de um novo projeto.
[2] Avalia o quão possível, fácil ou flexível seria o gerenciamento da rede.
[3] Aplicações que transportam tráfego originado por mídias diversas, tais como dados, voz e vídeo.

Para essas aplicações, a avaliação de qualidade pode seguir tanto os métodos tradicionais, subjetivos, quanto os novos padrões, modernos e objetivos. Não se pretende aqui fazer tal análise e será considerado bom funcionamento apenas o atendimento aos requisitos das aplicações e não a análise dos resultados. Ao avaliar qualidade de voz, por exemplo, não é pretensão utilizar métodos de análise de qualidade para esse fim, subjetivos ou objetivos, mas somente verificar se as exigências mínimas para o tráfego da voz foram ou não atendidas, de acordo com as especificações do projeto.

1.2 Motivações

O uso de soluções de *QoS* (*Quality of Service* – Qualidade de Serviço) – em redes de computadores está normalmente associada à demanda pelo tráfego de aplicações com características de tempo real, como voz e vídeo. Integrar essas aplicações a redes corporativas, sobretudo àquelas que transportam dados e aplicações críticas do ponto de vista do negócio, exige um projeto específico, que vai além do projeto de rede tradicional. Esse trabalho visa justamente abordar as etapas, arquiteturas e soluções para um projeto de rede centrado na garantia da *QoS* para usuários e aplicações.

A motivação para esse estudo, além da busca pela integração total de aplicações em rede, independente do seu modelo de funcionamento, tipo de fluxo ou nível de criticidade, foi a dificuldade em encontrar na literatura materiais que, de forma completa e articulada, guiassem um projeto de integração ao sucesso. Dessa forma, o objetivo será propor uma

metodologia baseada em etapas, que siga as modernas técnicas e arquiteturas formais de *QoS* para as redes *IP* (*Internet Protocol*), buscando chegar a um modelo conclusivo para utilização em projetos corporativos.

Aqui não serão abordadas as soluções para *QoS* na *Internet*, sendo o objetivo central a apresentação de soluções para redes corporativas, que são o objeto do Estudo de Caso apresentado.

1.3 Estrutura da dissertação

A primeira etapa do trabalho, referente aos capítulos 1 a 3, serviu para o estudo do comportamento das aplicações em rede e dos mecanismos usados na caracterização do tráfego. São inúmeras as técnicas utilizadas e propostas atualmente. Algumas delas visam a obtenção de matrizes de tráfego na *Internet* e aqui serão mostradas de forma resumida. Outras, mais simples, são suficientes para caracterização das redes corporativas. Além disso, são mostradas as arquiteturas padrão de *QoS* e suas principais características. Sobretudo é detalhada a *DiffServ* - Arquitetura de Serviços Diferenciados, que servirá de base para a proposta final de modelo de projeto. Foram estudadas as etapas da *DiffServ*, necessárias à implantação de solução fim-a-fim, descritas como classificação e marcação, gerência de filas, escalonamento e política de descartes, policiamento e suavização do tráfego e mecanismos de eficiência de enlaces.

Foi dada especial atenção às etapas de classificação e marcação que, efetivamente, consistem no reflexo das políticas administrativas de Qualidade de Serviço em identificação das aplicações, formação dos agregados de fluxos e modificações dos cabeçalhos do pacote *IP*, permitindo o tratamento diferenciado das classes na rede *DiffServ*. Aqui também foram abordados os mecanismos que buscam dar maior eficiência ao uso dos recursos de enlace. São tratadas as técnicas de suavização, fragmentação e intercalação, e compressão.

O capítulo 4 procura trazer uma orientação de como a *DiffServ* pode ser usada em um projeto de *QoS* para redes *IP*. Abordou-se as principais aplicações utilizadas em redes corporativas com o objetivo de mapear as características dessas aplicações em recomendações de projeto, sempre utilizando a *DiffServ* como orientação.

Em seguida, no capítulo 5, os mecanismos utilizados para a entrega de um projeto de *QoS* fim-a-fim com Serviços Diferenciados foram explorados. Foram detalhadas as funções e etapas de implementação, procurando esclarecer o que cada um desses mecanismos procura resolver, como são implementados e como a rede reage à sua operação.

O capítulo 6 descreve as características desejáveis e os padrões disponíveis para o gerenciamento de redes baseadas em políticas. Diversas aplicações proprietárias têm sido utilizadas nesse sentido, tentando fornecer interfaces para o gerenciamento dos índices de *QoS* e das *SLA*s (*Service Level Agreements* – Acordos de Nível de Serviço) e para a configuração e aplicação das políticas de *QoS*, atividade tipicamente chamada de

provisionamento[4]. Isso dificulta a interoperabilidade, independência e transparência na solução final. Existem ainda as soluções baseadas no *SNMP* (*Simple Network Management Protocol*) e aquelas que seguem o padrão para definição e distribuição de políticas do *IETF* (*Internet Engineering Task Force*), o *COPS* (*Common Open Policy Service*) (DURHAM et al, 2000). O *COPS* surge como uma proposta importante, com a possibilidade de integração total, independente da especificação final dos elementos de *hardware* do projeto. Além disso, o capítulo aborda as técnicas de medição em redes *IP* baseadas em políticas, procurando esclarecer os pontos chave envolvidos nessa tarefa, essencial para a avaliação de atendimento aos requisitos das aplicações especificados no projeto.

Por fim, o trabalho é finalizado com a apresentação de um estudo de caso real, selecionado a partir das implementações realizadas durante a pesquisa. O modelo proposto foi aplicado em ambiente de testes de laboratório, com a utilização de equipamentos de diversos fornecedores e em alguns projetos reais. Um desses projetos foi selecionado para apresentação no trabalho e descreve a solução de *QoS* adotada para a rede de uma grande empresa.

1.4 A Demanda por Qualidade de Serviço

No início das telecomunicações, existiam, em geral, duas redes separadas, uma para transporte de voz e outra para o transporte de dados. Cada uma das redes tinha um objetivo único que seria transportar um tipo específico de informação. A rede de telefonia, iniciada

[4] O termo provisionamento tem sido utilizado na linguagem técnica para referir-se às atividades necessárias ao fornecimento de um serviço ou recurso.

com a invenção do telefone por Graham Bell no século XIX, foi projetada para transportar voz. A rede *IP*, por outro lado, foi projetada para dados.

Nas redes telefônicas, o equipamento terminal foi concebido para ser simples. O aparelho telefônico consistia basicamente em um transmissor analógico, projetado para produzir uma flutuação de corrente elétrica a partir da pressão acústica da voz. Para efeitos práticos, essa seria a única função que um terminal deveria desempenhar. A rede, entretanto, era muito mais complexa, concebida com a "inteligência" necessária para prover os diversos serviços de voz. Uma conexão telefônica permanece dedicada à chamada enquanto ela durar. Uma vez finalizada, os circuitos podem ser usados para novas chamadas. Esses circuitos, utilizados para estabelecer uma chamada, alocados dinamicamente, são normalmente referenciados como troncos e diferem dos acessos (*loops*), que consistem em linhas permanentemente dedicadas a cada usuário individual.

Para as redes telefônicas tradicionais existiam duas medidas chave de qualidade. A primeira dizia respeito à probabilidade de a chamada não ser estabelecida devido à falta de um circuito tronco para suportá-la. Uma vez que a chamada fosse estabelecida, a outra medida de qualidade poderia ser avaliada e dizia respeito à qualidade da voz, propriamente dita. A qualidade da voz dependia da qualidade fim-a-fim da conexão durante a chamada, relativa às perdas, ruído, eco etc. Essas redes originais eram projetadas, portanto, com dois objetivos principais. O primeiro seria assegurar que haveria recursos (troncos) suficientes, em situações normais de funcionamento, para comutar as chamadas, mantendo a probabilidade de rejeitar chamadas pela falta de recursos em valores baixos, por exemplo, 1%. E o segundo objetivo

seria projetar a rede com recursos de transmissão otimizados e suficientes para garantir chamadas com valores de perdas, ruído, eco, atraso etc., dentro de limites que não afetassem a qualidade.

As redes *IP* surgiram com propósitos completamente diferentes da rede de telefonia. O *IP* foi projetado para transportar dados. Diferente da voz, tráfego de dados não era, e ainda não é na sua predominância, serviço de tempo real. Os dados poderiam ser armazenados na rede e entregues depois. Se fossem entregues com erro, poderiam ser retransmitidos.

Como a informação transportada pela rede *IP* era diferente da transportada pela rede telefônica, a filosofia usada no seu projeto também foi diferente. A rede foi concebida para ser o mais simples possível. A principal função era repassar pacotes de um nó ao próximo. Todos os pacotes eram tratados da mesma forma, armazenados em um só *buffer* e encaminhados por ordem de chegada. Outra grande diferença é que a "inteligência" estava nos equipamentos terminais, tipicamente computadores.

Por transportar basicamente um só tipo de informação, sem características especiais, como tempo real, por exemplo, a rede pôde ser projetada para operar no modo *best effort* (melhor esforço), ou seja, servindo quando possível e tratando todos os pacotes da mesma forma. Essa característica resultou em paradigmas de projeto mais simples. O objetivo principal era assegurar que os terminais teriam os protocolos apropriados para garantir a transmissão segura dos dados, com a rede operando da forma mais simples possível.

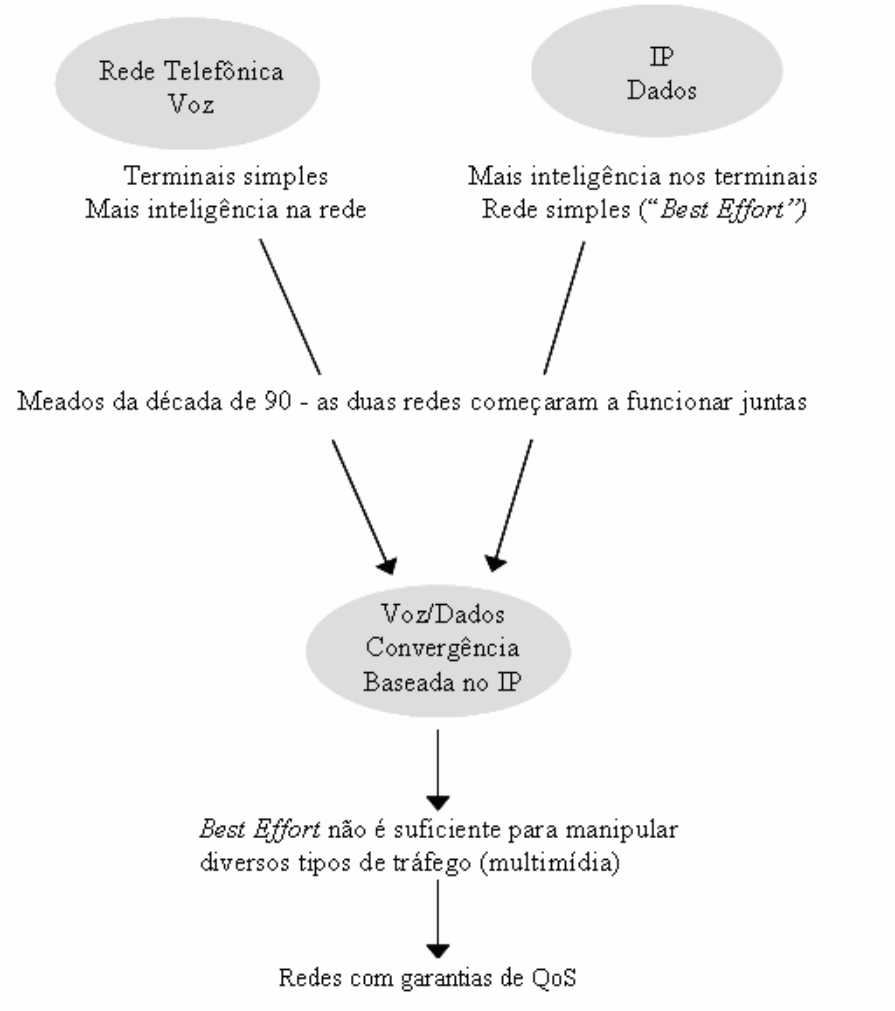


Figura 1-1: Evolução das Telecomunicações

Apesar das características distintas dos tráfegos de voz e dados, incluindo os diferentes requisitos de *performance*, seria possível projetar redes com capacidade de transportar os dois tráfegos e que servissem da melhor maneira a cada um dos tipos de carga. Em meados dos anos 90, surgiram as primeiras propostas de junção das redes, quando se iniciou a chamada convergência de voz e dados, com a idéia de que uma rede única e ubíqua fosse capaz de transportar os dois tráfegos (figura 1-1). Os grandes objetivos dessa estratégia seriam dar

maior eficiência e gerar economia. O que consistia, nessa época, em conceitos abstratos, hoje é realidade. Adicionalmente, além do tráfego de voz, as redes modernas suportam tráfego de vídeo, formando as chamadas redes multiserviço.

Com a convergência, contudo, novos desafios técnicos surgiram. Em uma rede convergente, a operação padrão da rede *IP* – *best effort* – não é suficiente para atender aos diversos requisitos de *performance*, muitas vezes conflitantes, dos tantos tipos de informação transportados.

A perspectiva de convergência para o *IP* evoluiu com a *Internet*. Com a popularização da *Internet*, o protocolo *IP* se tornou padrão para as aplicações que necessitavam de comunicação entre diferentes nós da rede. Dessa forma, as novas aplicações, quase que na sua totalidade, passaram a ser concebidas para executar sobre a arquitetura *TCP*/*IP* (*Transmission Control Protocol/Internet Protocol*). Aplicações antigas, como as baseadas na arquitetura *SNA* (*Systems Netwok Architecture*), por exemplo, foram portadas, com a criação de padrões específicos que possibilitaram a interoperabilidade com o *IP*. Os sistemas operacionais de rede, por sua vez, que utilizavam diversas soluções além do *TCP*/*IP* nas redes locais, tais como *IPX/SPX* (*Internetwork Packet Exchange / Sequence Packet Exchange)* e *AppleTalk*, passaram a implementar o *IP* como padrão para troca de mensagens. Houve, nesse caso, a convergência das aplicações de dados para o *IP*.

Além dos serviços de dados, surgiu a possibilidade de transportar tráfego de outras mídias, como a voz e vídeo, sobre o *IP*. Nesse contexto, a importância das tecnologias de *QoS* têm

crescido substancialmente nos últimos anos. Hoje, *QoS* é, sem dúvidas, um dos itens mais importantes para o desenvolvimento das redes de pacotes modernas, baseadas em *IP*.

Cada vez mais aplicações e serviços foram portados para o *IP*, o que impulsionou a criação de novas soluções e padrões, contribuindo para a disseminação da arquitetura. A utilização de um mesmo padrão, arquitetura, solução ou protocolo para transporte de aplicações com características tão diversas denomina-se convergência tecnológica. Tal convergência, anteriormente tentada com soluções e arquiteturas para as camadas inferiores, como *ISDN* (*Integrated Services Digital Network* – Rede Digital de Serviços Integrados), *ATM* (*Asynchronous Transfer Mode* – Modo de Transferência Assíncrono) e *Frame Relay*, exige da rede funcionalidades que permitam o atendimento aos requisitos de cada uma das aplicações transportadas.

Como as redes *IP*, por padrão, tratam todo o tráfego da mesma forma (serviço de melhor esforço ou *best effort*), surgiu o grande desafio das redes convergentes modernas: adaptar o *IP* de forma a tratar aplicações com requisitos distintos de forma distinta, atender tais requisitos e possibilitar o tráfego concorrente dessas diferentes aplicações na rede, sem prejuízos a nenhuma delas. Ao conjunto de mecanismos que propõem esse suporte a múltiplos tráfegos, com requisitos distintos, em uma rede *IP*, chama-se *QoS – Quality of Service*, ou Qualidade de Serviço.

Tanto as corporações, quanto os provedores de serviço, têm interesse na convergência de serviços para uma só infra-estrutura. As empresas vêem a possibilidade de reduzir custos,

concentrando investimentos e mão-de-obra em uma só tecnologia. Além disso, como todas as aplicações foram migradas para *IP*, precisam garantir que as aplicações críticas, normalmente relacionadas ao negócio, estejam protegidas das demais aplicações de menor relevância. Os serviços migrados (como voz e vídeo) também precisam de recursos estritos, pelas suas características de tráfego. Surgiu também a necessidade de otimizar a utilização dos recursos da rede. Esse fatores demandam a implementação de *QoS* nas redes corporativas.

Os provedores, por sua vez, diante da necessidade de oferecer serviços personalizados aos clientes, precisam atender aos requisitos explícitos de cada cliente (serviço) e das aplicações em uma mesma rede compartilhada. Esse fato demanda a implementação de políticas que disciplinem e garantam o funcionamento dos serviços e aplicações nas redes de provedores de serviço, comumente chamadas de *backbone*.

Os *backbones* dos provedores de serviço, projetados para grande demanda de tráfego, por muito tempo tiveram baixa utilização. Como os recursos eram pouco utilizados, os serviços eram garantidos. Com a demanda cada vez maior por novos serviços e com a crescente exigência contratual da manutenção de padrões de funcionamento desses serviços, essas redes tiveram que ser adaptadas para que, independente do nível de utilização, possam garantir os serviços como vendidos ao cliente.

2 REQUISITOS DE FUNCIONAMENTO DAS APLICAÇÕES EM REDE

QoS pode ser definida sob dois pontos de vista (PARK, 2005). O primeiro consiste na *QoS* experimentada pelo usuário final, representando um equipamento terminal, como um aparelho telefônico, um computador etc. ou também podendo representar um ser humano, usuário desses dispositivos. O segundo diz respeito à *QoS* sob o ponto de vista da rede.

Da perspectiva do usuário final, *QoS* se refere à sua percepção da qualidade que recebe da rede para um serviço ou aplicação específica, como, por exemplo, voz, dados ou vídeo. Da perspectiva da rede, o termo *QoS* se refere à capacidade de fornecer a qualidade esperada pelo usuário final. Essa capacidade, por sua vez, pode ser dividida em dois tipos. Primeiro, para fornecer *QoS*, uma rede deve ser capaz de diferenciar classes de tráfego para que o usuário final possa tratar uma ou mais classes de forma diferente de outras. Segundo, uma vez que a rede diferencia as classes de tráfego, deve ser capaz de tratar essas classes também de forma distinta, fornecendo garantias de recursos e diferenciação dos serviços dentro da rede (figura 2-1).

A percepção do usuário final é determinada por testes, muitas vezes subjetivos, que refletem e são função de variáveis da rede, como atraso, variação do atraso (*jitter*), taxa de perda de pacotes etc. (MCCABE, 1998). Essas variáveis serão aqui chamadas de parâmetros ou índices de *QoS*. Os valores dessas variáveis, normalmente associadas a uma medição de desempenho da rede, depende dos mecanismos de *QoS* implementados.

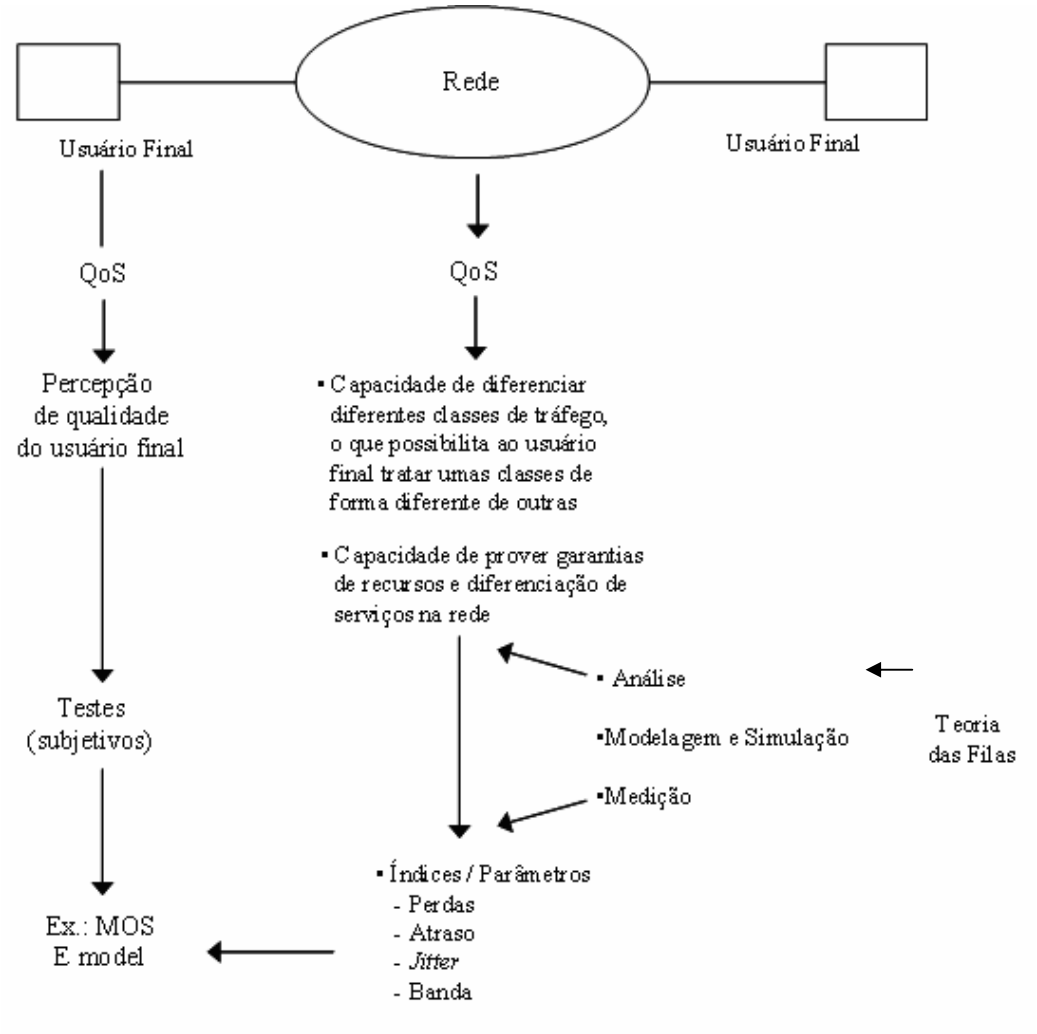


Figura 2-1: Definição de *QoS*

Fonte: PARK, 2005, p.5 (adaptado)

Uma vez que a rede transporta uma mistura de tráfegos com diferentes requisitos de desempenho, um determinado índice ou parâmetro importante para um serviço ou aplicação em particular pode não ser tão relevante para outros tipos de serviço ou aplicação e vice-versa. Os mecanismos de *QoS* implementados na rede devem considerar os diversos conflitos quanto aos requisitos de *performance*, buscando uma solução otimizada que possa atender a todos os serviços.

Ainda, o conhecimento dos requisitos das aplicações é necessário para que seja possível a especificação das garantias que a rede deve prover. Alguns desses requisitos são baseados nas características intrínsecas do tráfego gerado por essa aplicação. Outros dependem do grau de importância da aplicação no contexto da rede. Uma aplicação de correio eletrônico ou transferência de arquivos, por exemplo, pode ter pouca importância em uma determinada rede, mas pode ser extremamente crítica em outra.

Em resumo, algumas aplicações são "mais importantes" que outras, dependendo do contexto. Adicionalmente, as aplicações demandam recursos diferentes para funcionarem satisfatoriamente, como, por exemplo, requisitos de tempo real, latência e largura de banda mínima.

2.1 Parâmetros de Qualidade de Serviço

Os parâmetros ou índices servem de base para o dimensionamento dos recursos da rede e a especificação do projeto de *QoS*, de maneira a atender aos requisitos das aplicações. Aqui são listados os parâmetros mais comuns:

a) Vazão / capacidade

Refere-se à capacidade de transmissão de informações de um determinado meio por unidade de tempo, normalmente expressa em *bits* por segundo. Algumas aplicações necessitam de quantidade específica de banda para funcionar de acordo com a

performance esperada. O projeto de rede deve dimensionar vazão suficiente para o funcionamento adequado das aplicações. A tabela 2-1 (MARTINS, 1999), a seguir, ilustra o consumo típico de banda para algumas aplicações.

Tabela 2-1: Largura de banda necessária para as aplicações

Aplicação	Banda necessária (vazão gerada)
Aplicações Transacionais	1 kbps a 50 kbps
Voz	10 kbps a 64 kbps
Aplicações *Web* (*WWW*)	10 kbps a 500 kbps
Vídeo (*Streaming*)	100 kbps a 1 Mbps
Aplicação Conferência	500 kbps a 1 Mbps
Vídeo *MPEG*	1 Mbps a 10 Mbps
Aplicação Imagens Médicas	10 Mbps a 100 Mbps
Aplicação Realidade Virtual	80 Mbps a 150 Mbps

b) Atraso

Atraso refere-se ao tempo necessário para que um pacote chegue ao destino. O atraso total é a soma dos atrasos fixos ou constantes, e variáveis. Atrasos constantes são originados pelas características tecnológicas da solução adotada. Por exemplo, tempos gastos com codificação e serialização em uma aplicação *VoIP* (voz sobre *IP*) compreendem atrasos constantes e dependem da velocidade dos enlaces (Tempo de serialização por pacote(s) = tamanho do pacote (*bits*) / Taxa da linha (bps)), capacidade de processamento do codificador, complexidade do CODEC utilizado etc. Os atrasos variáveis são resultado de condições "ambientais". Isto é, estão relacionados a um estado da rede. A causa mais

importante desse tipo de atraso é o congestionamento. Em situações de contenção, os pacotes têm que aguardar nos *buffers* para que sejam despachados para a interface.

c) Variação do atraso (*jitter*)

O *jitter* pode ser definido como a diferença entre os atrasos na recepção de pacotes consecutivos. Esse tipo de atraso afeta os fluxos de tempo real e, se não é controlado, causa sérios problemas de degradação de *performance* a aplicações sensíveis à alteração na cadência de recepção de fluxos de pacotes. Para reduzir o *jitter*, são utilizados *buffers* de *jitter* que recebem os pacotes em tempos variados e despacham paulatinamente, mantendo as diferenças aproximadamente constantes. O uso desses *buffers* provoca aumento do atraso total, o que requer cuidados no dimensionamento. O efeito do *jitter* no fluxo de pacotes transmitidos quando chegam ao receptor pode ser observado através da figura 2-2.

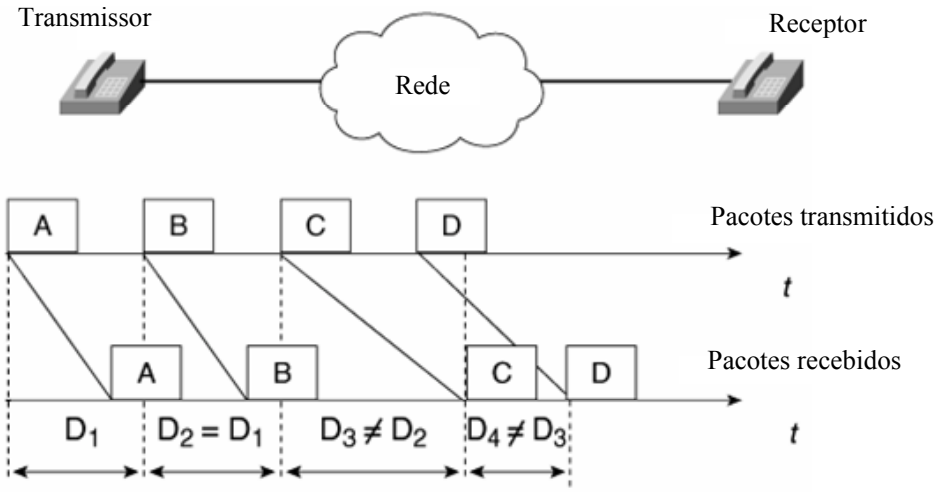


Figura 2-2: Efeito do *jitter* no fluxo de pacotes recebidos

d) *Skew* (obliqüidade)

É um parâmetro utilizado para medir a diferença entre os tempos de chegada de diferentes mídias que deveriam estar sincronizadas. Em certas aplicações existe uma dependência entre duas mídias, como áudio e vídeo, ou vídeo e dados.

e) Taxa de perda de pacotes

Mede a relação entre o número de pacotes entregues corretamente ao destino e o total de pacotes transmitidos. É expressa em porcentagem de pacotes perdidos e normalmente está relacionada à disponibilidade do serviço. Em uma rede altamente disponível, a taxa de perda de pacotes, em períodos sem congestionamento, deve ser zero. Em casos de congestionamento, os mecanismos de *QoS* devem entrar em operação justamente para decidir que pacotes, preferencialmente, devem ser descartados.

Outros parâmetros adicionais podem ser considerados e alguns deles constam nos chamados Acordos de Nível de Serviço – *SLA* (*Service Level Agreement*), assinados entre usuário e provedor do serviço: disponibilidade, *MTBF* (*Mean Time Between Failures* – Tempo Médio entre Falhas), *MTTR* (*Mean Time to Repair* – Tempo Médio para Reparo), taxa de erros etc.

Alguns desses parâmetros afetam a rede como um todo ou todos os serviços demandados pelos usuários e não um serviço ou aplicação em particular.

2.2 Tipos de aplicações

As aplicações em rede podem reagir de forma diferente diante da mudança de valores de variáveis relacionadas ao comportamento da infra-estrutura. Essa reação é normalmente associada à capacidade ou não que tem a aplicação de monitorar congestionamento, seja no receptor, seja na rede. Por esse parâmetro são assim classificadas:

a) Aplicações elásticas ou adaptativas

   Utilizam algum mecanismo de controle de fluxo que permite a variação da taxa de transmissão dependendo da monitoração que o transmissor faz da rede e da capacidade do receptor. As aplicações sobre *TCP*, que utiliza o mecanismo das janelas deslizantes, são, normalmente, adaptativas.

b) Aplicações não adaptativas ou não elásticas

   Nesse caso, o transmissor não monitora a ocupação do canal e não tem retorno sobre a capacidade do receptor. Dessa forma, tendem a transmitir a uma taxa constante. Aplicações sobre *UDP* (*User Datagram Protocol*) tendem a ser não adaptativas.

Há ainda aplicações com requisitos estritos quanto aos valores assumidos pelos parâmetros de qualidade. Sem que esses valores estejam dentro de limites bem definidos, essas aplicações

podem ter o desempenho degradado ou, simplesmente, não funcionar. São elas as aplicações de tempo real.

c) Aplicações de tempo real

Pelas características do tráfego gerado, precisam que os pacotes sejam entregues em intervalos de tempo fixo. A rede deve atender aos requisitos definidos pela aplicação, como tentar minimizar a distorção do tempo entre pacotes. Aplicações em tempo real não utilizam mecanismos de retransmissão de pacotes em caso de descartes, visto que cada pacote carrega informações significativas para aquele intervalo de tempo. Normalmente, utilizam transporte *UDP*, por este ser mais leve (menos *overhead*), não ter algoritmo de retransmissão ou controle de fluxo.

O ideal é que utilizem mecanismos que dinamicamente calculam e ajustam a mudança do atraso em resposta ao *jitter* médio inserido pela rede. Aplicações com essa característica são mais tolerantes ao *jitter* e utilizam o *RTP* (*Real Time Protocol*) para codificar os fluxos de dados, auxiliando na reprodução da taxa de amostragem original.

Aplicações em tempo real podem ser sub-classificadas em interativas e não interativas.

Não interativas – não há interação entre transmissor e receptor. A transmissão de conteúdo ocorre em um único sentido. Para esse grupo de aplicações, os pacotes devem ser entregues na mesma ordem em que foram gerados, e com cadência sempre constante

entre eles. São sensíveis à perda de pacotes e mais sensíveis ao *jitter* que ao atraso. Exemplos de aplicações em tempo real não interativas são *vídeo broadcast*, TV corporativa, vídeo sob demanda e rádio.

Interativas – há interação entre transmissor e receptor, com tráfego nos dois sentidos. Em função dessa característica, são sensíveis ao atraso e ao *jitter*. Além disso, exigem baixa taxa de perda de pacotes. Exemplos de aplicações em tempo real interativas são videoconferência e telefonia *IP*.

Cada aplicação tem características próprias, requerendo da rede recursos específicos de funcionamento. Analisados os parâmetros de qualidade de serviço citados no item 3.1, é possível generalizar os níveis de sensibilidade das aplicações a esses índices, conforme a tabela 2-2 a seguir:

Tabela 2-2: Sensibilidade das aplicações aos requisitos de *QoS*

	Voz	Vídeo	Dados	Dados Críticos
Banda	Baixa a Média	Média a Alta	Média a Alta	Baixa a Média
Perda de pacotes	Baixa	Baixa	Alta	Média a Alta
Atraso	Alta	Alta	Baixa	Média a Alta
Jitter	Alta	Alta	Baixa	Baixa a Média

Em resumo, algumas aplicações necessitam de garantias específicas para funcionarem em níveis de qualidade previstos no projeto. Esse conjunto de parâmetros, relacionados à qualidade das aplicações e necessários ao atendimento a um determinado fluxo de dados gerado, tanto os já detalhados, como outros (consumo de memória e *buffers*, níveis de utilização de *CPU* etc.), determinam a complexidade da solução de Qualidade de Serviço (*QoS*) que deve ser adotada para o ambiente em questão.

A pilha de protocolos *TCP/IP* e o protocolo *IP*, especificamente, provêem serviço do tipo *best effort*. Os pacotes são transmitidos de um ponto a outro sem nenhuma garantia especial de atendimento a requisitos de banda ou de atraso. Nesse modelo, as requisições são tratadas de acordo com a estratégia "primeiro a chegar, primeiro servido". Isso significa que todas as requisições têm a mesma prioridade e são processadas em seqüência. Não há possibilidade de fazer reservas de banda para um fluxo específico ou de dar maior prioridade de atendimento a determinado fluxo. Essas limitações, aliadas à crescente necessidade de dar tratamento diferenciado às aplicações em rede, geraram dois modelos de *QoS* para as redes *IP*: Serviços Intergrados (*IntServ – Integrated Services)* e Serviços Diferenciados (*DiffServ – Differentiated Services)*, detalhados adiante.

2.3 Estimativas de requisitos e acordos de nível de serviço

Uma das etapas do projeto de qualidade de serviço é a especificação, a mais precisa possível, dos requisitos das aplicações para a rede em questão. Nessa fase, o projetista tenta obter como resultado um padrão de comportamento (*baseline*), que servirá de referência para as demais etapas do projeto. Esse perfil definirá limites de funcionamento satisfatório das aplicações considerando as suas características intrínsecas e as variáveis ambientais, relacionadas às particularidade da rede.

Aplicações em tempo real, como o *VoIP*, por exemplo, têm requisitos estritos de funcionamento, pelas características próprias ao tráfego de voz. A qualidade de serviço torna-se um aspecto operacional fundamental para o desempenho dessas aplicações. A

obtenção de um *baseline*, para esses casos, torna-se uma tarefa menos complexa, visto que os parâmetros rígidos de funcionamento exigirão garantias mínimas da rede. Essas garantias, são comumente expressas e solicitadas em termos de uma "Solicitação de Serviço" ou "Contrato de Serviço", chamados tipicamente de *SLA* (*Service Level Agreement)* (MARTINS, 1999).

A *SLA* deve refletir os parâmetros de funcionamento especificados para o serviço (*SLS – Service Level Specification* – Especificação do Nível do Serviço), definindo claramente quais requisitos devem ser garantidos para que a aplicação possa executar com qualidade. Um exemplo típico de *SLA* para uma aplicação de voz sobre *IP* poderia mencionar os índices de *QoS* listados a seguir:

Taxa de Perdas $<$ 10 %

Atraso $\leq$ 150 mseg

Jitter $\leq$ 40 mseg

Além desses parâmetros, referentes ao funcionamento com qualidade de uma conversação telefônica, a *SLA* pode especificar variáveis que refletem particularidades de funcionamento do ambiente. Por exemplo, para o caso exemplificado, a *SLA* poderia determinar mais um parâmetro, como o número de chamadas simultâneas. Uma vez que a rede garanta esta *SLA*, tem-se como resultado que a aplicação *VoIP* em questão poderá executar com a garantia prevista de qualidade de voz para os seus usuários que se comunicam simultaneamente através da rede *IP*.

Para definir a largura de banda necessária e o tratamento dado aos pacotes de voz para atendimento da *SLA*, basta, portanto, que sejam calculadas as necessidades para garantia de

atendimento a uma conversação (canal), multiplicando-se esses valores pela quantidade total de chamadas simultâneas previstas no projeto. Assim, para que o desempenho de uma chamada seja garantido, o projeto deve prever recursos suficientes para o atendimento satisfatório a todas as chamadas previstas. Esses recursos variam com o tipo de codificação (CODEC) de voz utilizada e com a previsão ou não do uso de mecanismos de eficiência de enlaces, como compressão e fragmentação. Em resumo, para o tráfego de voz, citado aqui como exemplo, pode ser elaborado um modelo padrão que, teoricamente, serviria para qualquer ambiente com requisitos semelhantes de qualidade.

As maiores dificuldades estão concentradas na classificação das aplicações de dados. Embora seja necessário conhecer o funcionamento de cada aplicação utilizada na rede, é preciso classificá-las, buscando retratar em formato de parâmetros de rede (baixo nível de abstração), requisitos do negócio (alto nível de abstração). Esse ponto será detalhado no capítulo 4.

3 ARQUITETURAS OU MODELOS DE QUALIDADE DE SERVIÇO

Sem um mecanismo de *QoS*, uma rede *IP*, como já mencionado, provê serviço *best effort*. Nesse caso, todos os pacotes são tratados e encaminhados da mesma forma. Os mecanismos de *QoS* dão à rede *IP* a capacidade de diferenciar pacotes e tratá-los de forma diferenciada.

O *IETF* tem proposto e desenvolvido diversas arquiteturas para suporte a Qualidade de Serviço (*QoS*) na *Internet* (figura 3-1). Predominaram entre elas a Arquitetura de Serviços Integrados (*IntServ*) e a Arquitetura de Serviços Diferenciados (*DiffServ*). A solução *Multiprotocol Label Switching* (*MPLS*) não consiste em um padrão de qualidade de serviço, mas, por redefinir através de um novo modelo o mecanismo de encaminhamento de pacotes na rede *IP,* que busca essencialmente maior desempenho e, por facilitar a integração com as demais arquiteturas, é, freqüentemente, relacionada a *QoS*. Neste capítulo, serão abordadas as funcionalidades básicas das arquiteturas *IntServ* e *DiffServ*.

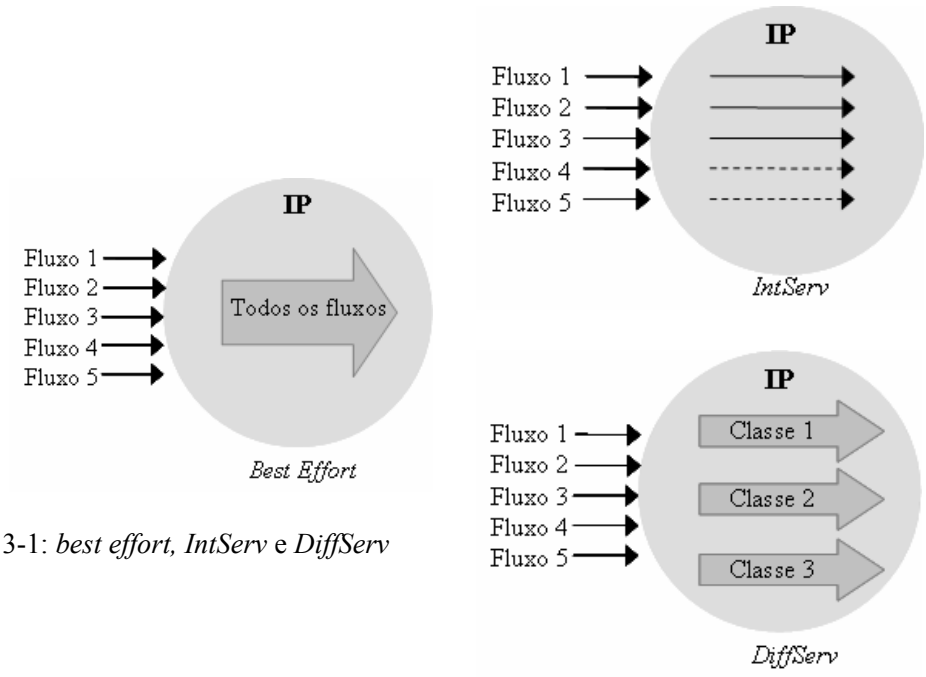


Figura 3-1: *best effort, IntServ* e *DiffServ*

3.1 Serviços Integrados

O modelo de Serviços Integrados (*IntServ*) (BRADEN, 1994) propõe que o fornecimento dos recursos necessários a uma aplicação sejam alocados com base em fluxos entre uma origem e um destino (figura 3-2). Um fluxo é definido como uma seqüência específica de pacotes gerados entre um transmissor e um receptor. Os recursos, determinados na requisição da aplicação (banda, por exemplo), chamados de "*flowspec"*, são solicitados pela aplicação dinamicamente e devem ser repassados a cada dispositivo da rede existente entre a origem e o destino. Se todos os dispositivos dispõem dos recursos solicitados, eles são reservados para utilização pela aplicação. Essa reserva é mantida enquanto houver demanda daquele fluxo. Quando não houver mais demanda, a reserva será desfeita e os recursos serão liberados para serem usados por outros fluxos.

O protocolo utilizado para troca de mensagens de reserva e sinalização é o *RSVP* (*Resource Reservation Protocol)* (BRADEN, 1997). Através de um conjunto de mensagens específicas do *RSVP* (figura 3-2), os recursos solicitados podem ser reservados em cada dispositivo da rede. O *RSVP* não é responsável pela qualidade de serviço, mas pelo transporte das informações necessárias à solicitação, estabelecimento e manutenção da reserva que garantem a qualidade de serviço pretendida pela aplicação. Devido ao *RSVP* ser um protocolo *simplex*, as reservas são feitas em uma direção. Para conexões *full-duplex*, como videoconferência e chamadas telefônicas, é necessária a reserva nos dois sentidos.

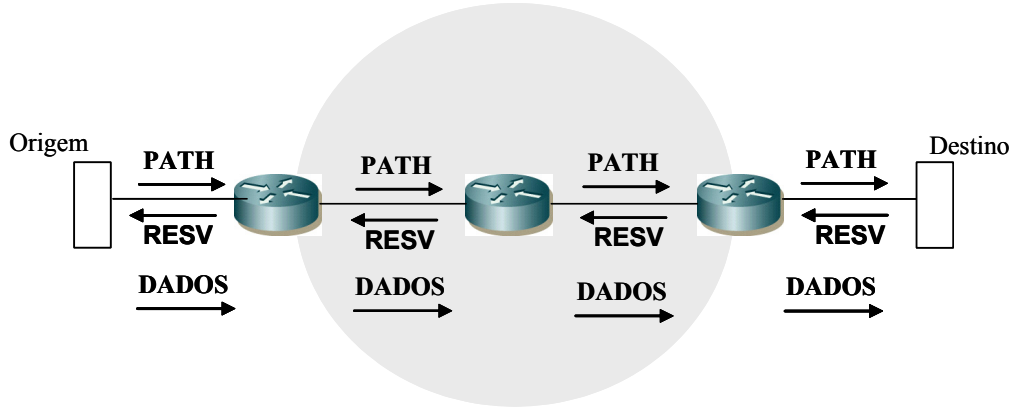


Figura 3-2: Operação *RSVP*

A reserva de recursos é iniciada pelo receptor. Previamente ao envio de dados da aplicação, o transmissor envia ao receptor uma indicação dos parâmetros que contêm os requisitos de *QoS* necessários ao fluxo em questão. Essa indicação é feita através de mensagens de sinalização *RSVP* (*PATH*). O receptor, caso disponha dos recursos solicitados, envia de volta uma mensagem *RSVP* (*RESV*), que estabelece efetivamente a reserva. A propagação das mensagens é feita salto a salto entre origem e destino e todos os elementos intermediários têm que dispor dos recursos solicitados para que a reserva seja feita. Caso algum dos dispositivos não disponha dos recursos, as reservas já feitas até aquele ponto são desfeitas.

Uma vez instalada a reserva em cada roteador ao longo do caminho, o fluxo de dados que a requereu tem a garantia da *QoS* fim-a-fim, conforme especificado na requisição da reserva, desde que não ocorram mudanças de rotas ou falhas nos roteadores. Serviços específicos de policiamento e ações de recondicionamento de tráfego são empregados na rede para assegurar que fluxos de dados não conformes ou não gerados pela aplicação requisitante não afetem a oferta de *QoS* acordada. O *IETF* especificou formalmente duas classes de serviço para a

IntServ , para uso com o *RSVP* (tabela 3-1): os Serviços Garantidos e os Serviços de Carga Controlada. Os Serviços Garantidos, definidos na RFC-2211, emulam um circuito dedicado entre origem e destino do fluxo de dados, para uso exclusivo da aplicação requisitante (WROCLAWSKI, 1997). Os Serviços de Carga Controlada, definidos na RFC-2212, tendem a emular uma rede sem ou com pouco congestionamento.

Tabela 3-1: Classes na Arquitetura *IntServ*

Classe de Serviço *IntServ*	Tipo de Serviço Oferecido	Utilização
Serviço Garantido – *Guaranteed Service (GS)*	Estão sempre garantidos os índices de *QoS* acordados (atraso, *jitter*, largura de banda, taxa de perda de pacotes) independente do nível de utilização da rede.	Aplicações em tempo real
Carga Controlada – *Controlled Load (CL)*	Estão garantidos os índices de *QoS* acordados (atraso, largura de banda e taxa de perda de pacotes). Caso haja disponibilidade, a rede pode oferecer níveis acima do mínimo garantido. Nessa classe, não há garantia explícita quanto ao *jitter*.	Aplicações tolerantes a *jitter*.
best effort (BE)	Utiliza a banda que não foi alocada para as outras classes.	Aplicações não críticas.

Para suportar *IntServ*, os roteadores têm que ser capazes de prover a *QoS* apropriada para cada fluxo, de acordo com o modelo. A função básica necessária à provisão de diferentes níveis de tratamento é o controle de tráfego, que consiste basicamente de três componentes:

- Controle de Admissão - utilizado no início da chamada para verificar se o nó tem recursos suficientes para atender a qualidade de serviço solicitada. O controle de admissão contém o algoritmo de decisão utilizado pelo roteador para determinar se há recursos de rede suficientes para aceitar a *QoS* solicitada para um novo fluxo de dados. A aceitação de um fluxo novo de dados sem a garantia de que a rede dispõe de recursos suficientes prejudicaria garantias anteriores e o novo fluxo, nesse caso, precisa ser rejeitado. Se o

novo fluxo for aceito, a solicitação de reserva no roteador designa o classificador de pacotes e o escalonador de pacotes para reservarem a *QoS* reservada para o fluxo. O controle de admissão é chamado em cada roteador ao longo do caminho de reserva, para tomar uma decisão de aceitação / rejeição sempre que um *host* requisitar um serviço com *QoS*.

- Escalonador de Pacotes (*packet scheduler)* – Gerencia o encaminhamento de pacotes baseado em sua classe de serviço, usando mecanismos de gerenciamento de filas e algoritmos de escalonamento. Deve assegurar que a entrega do pacote esteja associada aos parâmetros de *QoS* de cada fluxo. Pode também policiar ou moldar o tráfego para adequá-lo a determinado nível de serviço. Um escalonador deve ser implementado no ponto onde os pacotes são enfileirados;

- Classificador de Pacotes - O classificador de pacotes identifica os pacotes de um fluxo *IP* em *hosts* e roteadores que irão receber certo nível de serviço. Para realizar um controle efetivo de tráfego, cada pacote de entrada é mapeado pelo classificador em uma classe específica. Todos os pacotes que são classificados na mesma classe obtêm o mesmo tratamento por parte do programador de pacotes. A escolha de uma classe baseia-se nos endereços de origem e de destino e no número da porta no cabeçalho do pacote existente ou em um número de classificação adicional que precisa ser adicionado a cada pacote. Uma classe pode corresponder a uma ampla categoria de fluxos de dados. Por exemplo, todos os fluxos de vídeo de uma videoconferência com vários participantes podem

pertencer a uma classe de serviço, o que não impede outro tipo de abordagem, com apenas um fluxo pertencendo a uma classe de serviço específica.

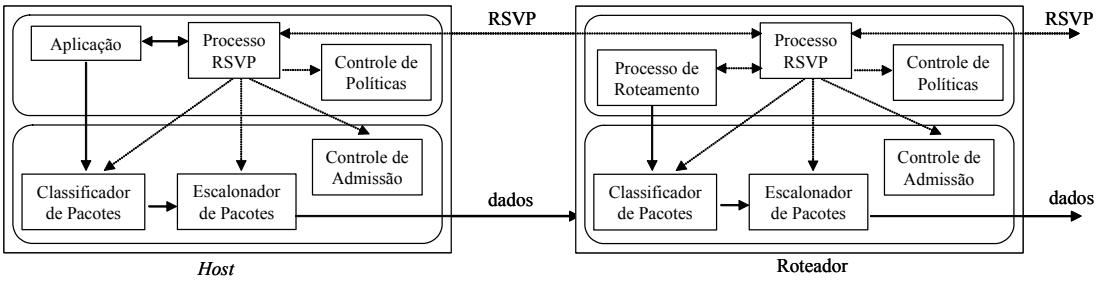


Figura 3-3: A Arquitetura *IntServ*

Fonte: Rodrigues, 2001, p.785 (adaptado)

A arquitetura *IntServ* (figura 3-3), contudo, não é apropriada para grandes redes por conta dos problemas inerentes ao projeto. Esse modelo não é escalável, por requerer a manutenção de estado por fluxo em cada nó da rede. Além disso, cada pacote deve ser classificado em diferentes classes de serviço. As reservas sob demanda para cada fluxo introduzem um alto grau de complexidade nos nós da rede. Por essa razão, a implantação dessa arquitetura em redes reais se torna uma tarefa complexa. Por conseqüência, teve pouco sucesso do ponto de vista de aceitação como solução efetiva e prática para *QoS*.

Para resolver os problemas de escalabilidade e flexibilidade relacionados à *IntServ*, o *IETF* propôs o Modelo de Serviços Diferenciados – *DiffServ*.

3.2 Serviços Diferenciados

Ao contrário da arquitetura *IntServ*, na *DiffServ* (BLAKE, 1998) a rede não mantém um estado de reserva de recursos. Nesta arquitetura, os pacotes são classificados de acordo com a classe de serviço a que pertencem. A classe corresponde a um agregado de fluxos com requisitos semelhantes de *QoS*.

Se na *IntServ* os recursos são reservados dinamicamente, aqui eles são alocados previamente para cada uma das classes definidas em cada um dos dispositivos e utilizados quando há disputa por recursos.

Na *DiffServ* cada dispositivo da rede desempenha tarefas específicas sobre o tráfego manipulado. Ao conjunto de dispositivos configurados para seguir a mesma política de *QoS DiffServ* dá-se o nome de domínio *DiffServ* (figura 3-4) .

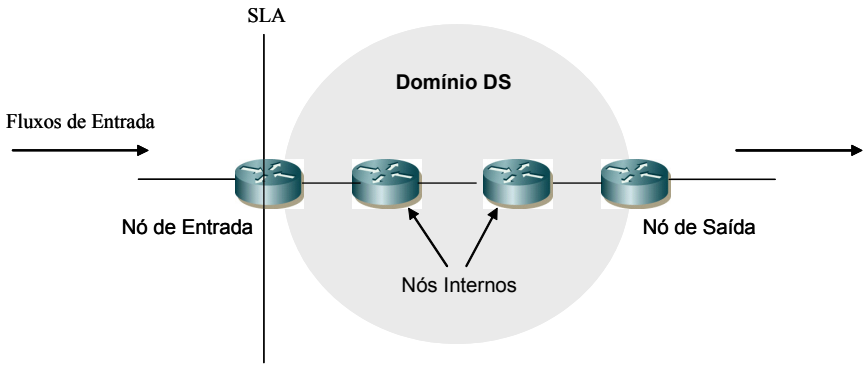


Figura 3-4: Domínio *DiffServ*

Os elementos localizados nas extremidades da rede, ou seja, mais próximos da origem e destino do tráfego gerado, são, normalmente, responsáveis pelas tarefas de classificação e marcação do tráfego. Dessa forma, são eles que identificam o tráfego, fazendo o agrupamento dos fluxos em classes, marcando, em seguida, os pacotes de cada classe, também chamadas de *Behavior Aggregate* (*BA*), com um mesmo identificador, possibilitando, com isso, que outros roteadores ao longo do caminho consigam identificar as classes de forma simples. O processo de marcação é feito com a manipulação do conteúdo do campo *DS* (*Differentiated Service*) do cabeçalho *IP* com um valor apropriado, denominado *DSCP* (*Diffrentiated Service Code Point*).

Essas tarefas demandam maior processamento do roteador, o que torna inviável sua implementação em roteadores de *backbone*, que precisam tratar grandes volumes de tráfego com a maior agilidade possível. Além da classificação e marcação, os dispositivos de borda devem executar outras funções necessárias à implantação de *QoS* fim-a-fim com *DiffServ*. As funções incluem policiamento e suavização do tráfego, gerenciamento de filas, controle de admissão, além dos mecanismos de eficiência de enlaces – compressão e fragmentação, todas elas descritas no capítulo 5. A esse conjunto de funções, costuma-se chamar Condicionamento de Tráfego – *Traffic Conditioning* (*TC*).

Os outros roteadores, internos ao domínio *DiffServ*, têm a função de tratar o tráfego já previamente classificado e marcado. Esses dispositivos devem priorizar o tráfego de acordo com os recursos previamente alocados para cada uma das classes definidas no projeto. Essa tarefa, de menor complexidade, requer menos recursos computacionais do roteador. A esse

conjunto de funções, executadas pelos roteadores internos ao domínio, chama-se *PHB* (*per hop behavior* – comportamento no nó). A aplicação de um comportamento do tipo *PHB* é acionada em cada elemento no interior da rede através do valor de *DSCP*. Essa arquitetura faz com que o processamento mais complexo, relacionado às tarefas de classificação e admissão do tráfego, fique localizado na borda da rede. No interior do domínio *DiffServ*, os elementos ativos respondem à marcação, contida no campo *DSCP* do pacote, de forma apropriada.

O campo *DS* do cabeçalho *IP* é composto de 8 *bits* e substitui o campo *ToS* (*Type of Service*) (BLAKE et al, 1998), definido na especificação original do *IP* (RFC-791). Os primeiros 6 *bits* do *DS* formam o *DSCP*. O campo *DS* é dividido em duas partes: *Class Selector* (3 primeiros *bits*) e *Priority Discard* (3 últimos *bits*). Os dois *bits* restantes são utilizados para sinalização de congestionamento, conforme detalhado no capítulo 5, seção 5.5.

Assim, pacotes que chegam à rede são identificados e condicionados, de acordo com a política da rede, e associados a uma classe (ou *BA – Behavior Aggregate*). De forma simples, recebem um mesmo valor de *DSCP*.

O campo *ToS* sofreu algumas modificações ao longo dos anos, conforme mostrado a seguir:

Inicialmente, os *bits* do campo *ToS* tinham o seguinte significado (tabela 3-2):

- Bits 0 a 2: Prioridade do pacote (*IP Precedence*);
- Bit 3: Requisitos de atraso - 0 = atraso normal (*normal delay*); 1 = baixo atraso (*low delay*);

- Bit 4: Requisitos de vazão - 0 = vazão normal (*normal throughput*); 1 = alta vazão (*high throughput*);
- Bit 5: Requisitos de confiabilidade – 0 = normal (*normal reliability*); 1 = alta (*high reliability*);
- Bits 6 e 7: reservados para uso futuro.

Tabela 3-2: Bits do Campo *ToS* pela RFC-791

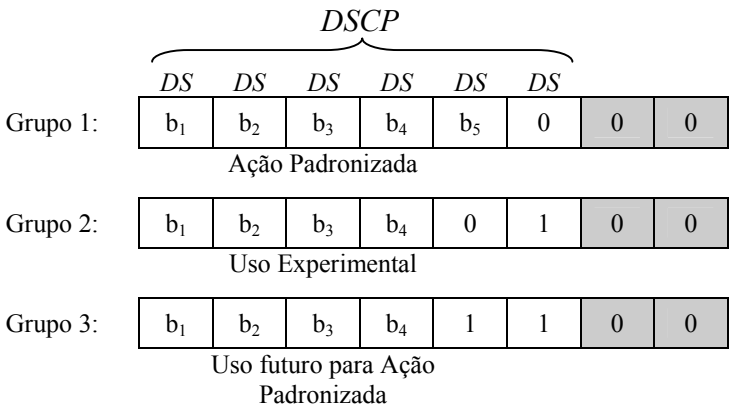
0	1	2	3	4	5	6	7
Precedence			D	T	R	0	0

Essa especificação original foi remodelada (ALMQUIST, 1992). Nessa recomendação, a definição dos selecionadores de serviços passou a conter 4 *bits* e foi retirada a possibilidade de solicitação de múltiplos serviços simultaneamente, não sendo possível solicitar serviços com baixo atraso e alta vazão, por exemplo. Os três valores definidos anteriormente foram mantidos e um quarto valor foi incluído para indicar o custo financeiro (*monetary cost*):

- Bit 6: Custo financeiro - 0 = normal (*normal monetary cost*); 1 = baixo (*low monetary cost*);
- Bit 7: reservados para uso futuro.

A arquitetura *DiffServ* redefiniu o significado do campo *ToS*. O conteúdo do campo é interpretado pelos roteadores do domínio *DiffServ* para indicar um tratamento particular, ou *PHB*, em cada nó da rede.

O campo *DS* pode gerar até 64 combinações diferentes. Este conjunto de combinações é dividido em três grupos (figura 3-5), sendo o primeiro grupo de 32 combinações para utilização em ações padronizadas, o segundo grupo, de 16 combinações para uso experimental e o terceiro grupo, de 16 combinações, também para uso experimental, devendo, preferencialmente, ser alocado para futuras ações padronizadas em caso de exaustão do primeiro grupo. A disposição dos *bits DS* e os grupos *DSCP* são mostrados na figura 3-5.



Onde $b_i = 0$ ou $b_i=1$, $1 \leq i \leq 6$

Figura 3-5: Grupos *DSCP*

Foi definido também o conceito de seletor de classe (*Class Selector*) (NICHOLS, 1998), como qualquer uma das oito combinações possíveis com os três primeiros *bits* do campo *DS*. Para definição de *Class Selector*, o formato do campo é $b_1b_2b_3000$, onde $b_i=0$ ou $b_i=1$, com $1 \leq i \leq 3$. O valor do *Class Selector* será determinado pelos *bits* b_1, b_2 e b_3, assumindo, portanto, valores entre 000000 e 111000.

Essa redefinição do campo *ToS* garante compatibilidade com o significado inicial definido na RFC-791, possibilitando interoperabilidade entre dispositivos que suportam *DSCP* e dispositivos que suportam somente *ToS* (tabela 3-3). Os três primeiros *bits* do campo *DS* têm o mesmo significado dos *bits* de prioridade do campo *ToS* (figura 3-6).

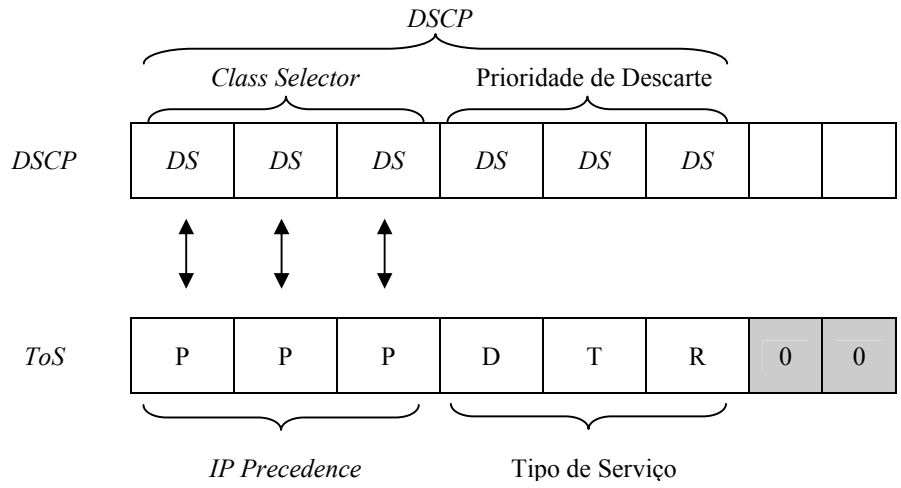


Figura 3-6: Correspondência entre os *bits DSCP* e *ToS*

Tabela 3-3: Correspondência de valores *CS – IP Precedence*

DSCP Class Selector	*ToS IP* Precedence	Bits
CS0	0 - *Routine*	000000
CS1	1 - *Priority*	001000
CS2	2 - *Immediate*	010000
CS3	3 - *Flash*	011000
CS4	4 – *Flash-override*	100000
CS5	5 - *Critical*	101000
CS6	6 - *Internet*	110000
CS7	7 - *Network*	111000

A arquitetura *DiffServ* define dois grupos *PHB* (*Per Hop Behavior*) (BLAKE et al, 1998): *Assured Forward* (*AF*) e *Expedited Forward* (*EF*).

O *PHB AF* é composto de quatro classes e três níveis de prioridade de descarte de pacotes por classe, perfazendo um total de 12 *PHB* (GROSSMAN, 2002). Em função da crescente demanda por serviços com garantia de entrega na *Internet*, o *PHB AF* foi desenvolvido para que os provedores pudessem oferecer, de forma padronizada, diferentes níveis de garantia de entrega de pacotes aos usuários (HEINANEN et al, 1999a). Por definição, o objetivo das classes *AF* é evitar a ocorrência de congestionamentos persistentes por meio de mecanismos de descarte de pacotes, policiamento ou suavização de tráfego.

As classes são identificadas pelo formato *AF*ij, onde i representa o número da classe, sendo $1 \leq i \leq 4$ e j a prioridade de descarte dentro da classe, sendo $1 \leq j \leq 3$. Os valores decimais de i e j, convertidos em notação binária, identificam o conteúdo dos cinco primeiros *bits* do campo *DS*. O sexto *bit* deve ser sempre zero, representando ações padronizadas.

Quanto maior o valor de i, maior a prioridade da classe *AF* e quanto maior o valor de j, maior a prioridade de descarte do pacote dentro da classe *AF*. Essa prioridade de descarte define que, caso ocorra disputa por recursos entre pacotes de uma mesma classe, aquele que tiver menor prioridade deverá ser encaminhado primeiro. Os valores e prioridades do *PHB AF* são mostrados na tabela 3-4.

O *PHB EF* (*Expedited Forwarding*) tem como objetivo oferecer serviço com baixa taxa de perda de pacotes, baixo atraso, baixo *jitter* e garantia de largura de banda fim-a-fim em um domínio *DiffServ* (JACOBSON, 1999). Esse *PHB* é, portanto, recomendado para uso no

transporte de voz e vídeo interativos. O valor do *DSCP* recomendado para o *PHB EF* é 101110.

Tabela 3-4: *PHB AF* - Classes e Prioridades

Ação Padronizada

Classe de Serviço			Prior.Descarte		0	Classe	Prioridade para Descarte
0	0	1	0	1	0	*AF*11	Baixa
0	0	1	1	0	0	*AF*12	Média
0	0	1	1	1	0	*AF*13	Alta
0	1	0	0	1	0	*AF*21	Baixa
0	1	0	1	0	0	*AF*22	Média
0	1	0	1	1	0	*AF*23	Alta
0	1	1	0	1	0	*AF*31	Baixa
0	1	1	1	0	0	*AF*32	Média
0	1	1	1	1	0	*AF*33	Alta
1	0	0	0	1	0	*AF*41	Baixa
1	0	0	1	0	0	*AF*42	Média
1	0	0	1	1	0	*AF*43	Alta

4 CONSIDERAÇÕES DE PROJETOS DE QoS COM DIFFSERV

Esse capítulo busca oferecer uma visão abrangente das fases de projeto e implementação de *QoS* com *DiffServ*. Tal processo requer um levantamento minucioso preliminar acerca dos objetivos e importância de cada aplicação sob o ponto de vista do negócio, bem como a definição de que tipo de tratamento deve ser dado a cada uma delas.

Serão abordadas aqui aplicações com requisitos distintos de *QoS*. São elas: voz, sinalização de chamadas, vídeo interativo, vídeo unidirecional (*streaming)*, dados sem prioridade (*best effort)*, aplicações de uso intensivo de banda (*bulk data*), dados transacionais, dados de missão crítica, tráfego de roteamento, tráfego de gerenciamento e suporte, além de tráfego não classificado, indesejado ou não cooperativo (*worst effort*).

4.1 Aspectos fundamentais

Um projeto de integração, até pouco tempo, consistia basicamente nas seguintes etapas (OPPENHEIMER, 2004) (figura 4-1):

- Análise de requisitos – Identificação dos objetivos do projeto; levantamento de requisitos de usuários; estudo das aplicações e seus requisitos ou parâmetros de funcionamento: disponibilidade, desempenho, escalabilidade, atraso, confiabilidade, gerenciabilidade, facilidade de uso, custo etc.;

- Definição de níveis de serviço – A partir da análise de requisitos, chega-se aos critérios finais de avaliação e gerenciamento da rede, de forma a garantir que os parâmetros estejam sempre entre valores pré-determinados;

- Projeto lógico – Caracterização do fluxo de tráfego entre sistemas, levantamento dos interesses de tráfego, projeto de endereçamento, projeto de roteamento, planejamento da solução de gerenciamento da rede;

- Projeto físico – Especificação de tecnologias e produtos que implementem o projeto lógico selecionado. Essa fase difere drasticamente, dependendo da abrangência da solução: rede local (*LAN*), rede de longa distância (*WAN*) ou rede corporativa (*LAN* e *WAN*) e consiste na especificação de elementos de interconexão, cabeamento, especificação dos enlaces, provedores de serviço, etc.

Hoje, além dessas etapas, importantes e obrigatórias, podem-se citar mais duas fases absolutamente relevantes:

- Projeto de Segurança – Definição da Política de Segurança Corportiva, definição de interesses de tráfego entre elementos da rede, definição de serviços da rede, elaboração de modelo de interconexão entre localidades e sub-redes, projeto de interligação com *Internet* e parceiros, dentre outros;

- Projeto de Qualidade de Serviço – Elaboração das políticas de atendimento a redes, sub-redes, *hosts* aplicações e serviços de acordo com os requisitos previamente levantados; aplicação e gerenciamento das políticas.

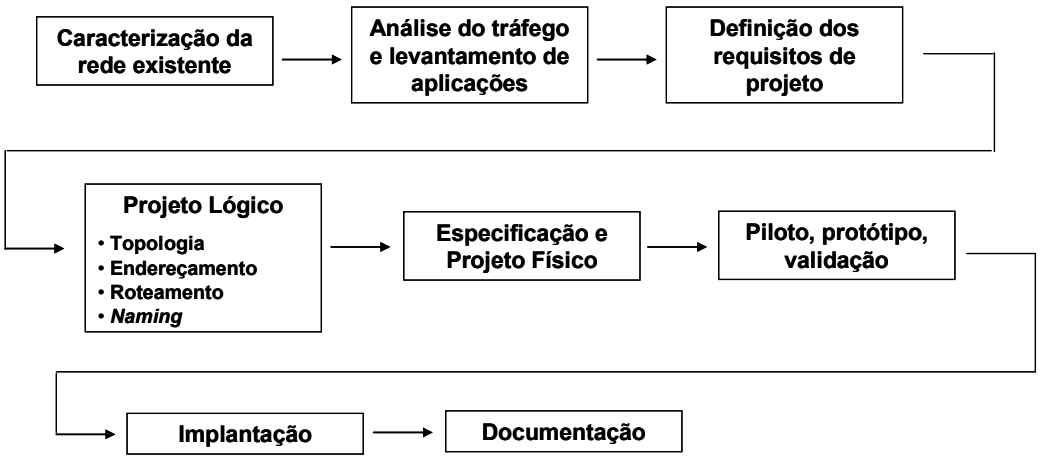


Figura 4-1: Etapas do Projeto de Redes

4.2 Métodos de medição de tráfego e caracterização das redes

Uma das etapas necessárias à elaboração de um projeto de qualidade de serviço é a caracterização do tráfego que flui pela rede. Uma diversidade de técnicas e ferramentas pode ser usada para essa finalidade. Para as redes corporativas, a atividade de medição de tráfego e construção de perfis é menos problemática, sobretudo se considerada a média de complexidade da grande parte das redes e a previsibilidade de comportamento do tráfego (interesses de tráfego, localização de serviços e clientes etc.).

Na *Internet*, entretanto, esse ponto é alvo de várias pesquisas e obter um modelo que possibilite a construção de perfis de tráfego através de medições pode demandar muito tempo. Uma das possibilidades, nesse intento, é a utilização das matrizes de tráfego. O uso de matrizes de tráfego (MT) é um recurso importante, pois fornece uma representação sucinta da troca de dados entre pontos finais em uma rede *IP*. Uma matriz de tráfego descreve o volume de tráfego que atravessa a rede, do ponto de entrada até um ponto de saída dessa rede. As MTs facilitam e até possibilitam a execução de muitas tarefas de gerenciamento necessárias para garantir que a rede opere com alta performance e eficiência, como planejamento de capacidade, análise de confiabilidade, dentre outras. A questão chave é como obter uma matriz de tráfego e com base em que tipo de informação.

Em redes corporativas, é possível obter matrizes de tráfego diretamente. Mas normalmente o que se faz é aplicar técnicas de inferência estatística sobre informações obtidas com uso do *SNMP*, o que permite que se mensure o volume de tráfego total envolvendo um enlace. Esse volume é a soamtória dos volumes medidos para cada fluxo, que, por sua vez, é determinado por um par Origem-Destino (OD).

Quando se pensa no contexto dos provedores de *Internet* (*ISPs – Internet Service Providers*), o desafio é bem mais complexo. Na *Internet*, os nós de origem e destino não são estações, são *PoPs* (*Points-of-Presence* – Pontos de Presença). Isso significa que a topologia de rede, nesse caso, é muito mais complexa, pois cada *PoP* mantém diversos roteadores de núcleo e de acesso, que, por sua vez, estão provendo acesso a incontáveis *links*, trafegando grandes volumes de dados. Nesse contexto, o uso de estatística mostra-se limitado. Primeiro, porque se baseia geralmente em dados

muito escassos, considerando o tráfego total da rede, ou seja, as informações obtidas passam a não ser representativas do tráfego real da rede. Segundo, porque, mesmo revelando volumes de tráfego de *links*, não permite que se conheça a origem e o destino do tráfego. Outro problema do uso de técnicas de estatística é que elas normalmente geram um aumento considerável de processamento (MEDINA et al., 2004).

A solução para o problema tem sido buscada no uso de informações obtidas a partir de um conhecimento *a priori* de um modelo razoável do tráfego em questão. Existem várias abordagens propostas para esse fim, seja propondo e utilizando um algoritmo ou solução específica, seja utilizando as ferramentas de medição padrão.

Para as redes corporativas, objeto desse estudo, são utilizadas diversas ferramentas, muitas delas baseadas em *SNMP* e do tipo *NetFlow*, para a obtenção de dados de medição confiáveis, que permitam a construção dos perfis de tráfego e, mais que isso, permitam inferir alguma tendência ou projeção sobre o comportamento da rede.

4.3 Os requisitos do negócio

O tratamento diferenciado de aplicações em rede requer duas análises básicas: a primeira deve considerar as características intrínsecas do tráfego gerado por essa aplicação, para que a rede possa garantir que os requisitos mínimos de funcionamento sejam atendidos. Essa análise é absolutamente necessária, sobretudo para aplicações com características de tempo real. A

segunda diz respeito ao peso da aplicação no contexto da rede em questão. Ou seja, deve considerar, não somente requisitos técnicos de funcionamento, mas também as necessidades inseridas pelo nível de criticidade associado, que é proporcional à importância da aplicação para o negócio.

Valores de classificação marcados no campo *DSCP*, apesar de, por definição do modelo, sugerirem refletir graus de prioridade e tipos de serviço específicos, não necessariamente são assim implementados. Como a marcação do campo é livre, é possível ao administrador de rede especificar critérios próprios de marcação, sem obediência à recomendação. O que quer dizer que, apesar de constituir uma violação à recomendação, nada impede que um pacote marcado com *EF – Expedited Forward,* por exemplo, receba menor prioridade do mecanismo de escalonamento e menos recursos da rede que aplicações com pacotes marcados com *DSCP* de menor prioridade. Ainda, há casos em que as aplicações marcam os pacotes gerados com valores de *DSCP* altos, mas que, no contexto da rede, essa aplicação pode ser menos relevante que aplicações consideradas de missão crítica.

O projeto de *QoS* deve, portanto, seguir os padrões definidos pela arquitetura, fazendo com que as marcações reflitam estritamente o requerido pela classificação. Além de garantir padronização e conformidade com as recomendações, essa estratégia facilita a integração entre domínios *DiffServ* diferentes. Há, contudo, dificuldades nas etapas de identificação de aplicações, classificação em alto nível (grau de criticidade) e, por fim, na análise de requisitos.

Para a seleção das aplicações e classificação subjetiva, relativa à sua importância, não há outro caminho senão conhecer o negócio e, nesse ambiente, buscar a inserção de cada uma das aplicações levantadas. Em alguns cenários, essa tarefa pode ser facilitada pelas características da rede e pelo tipo de segmento da empresa. Sabe-se que um banco, por exemplo, deverá considerar críticas as aplicações que afetam o funcionamento das agências, assim como uma indústria deverá privilegiar as aplicações de automação, por exemplo. Em outros casos, contudo, essa classificação é subjetiva e depende de informações operacionais.

No caso das aplicações de tempo real, a classificação deve refletir também os seus requisitos de funcionamento. Nesse caso, classificar uma aplicação de *VoIP*, do ponto de vista do negócio, como pouco importante, por exemplo, não deve afetar na montagem das políticas de *QoS* da rede, garantindo os recursos mínimos necessários ao bom desempenho. As restrições originadas pela política de alto nível podem, entretanto, refletir em outros itens da *QoS* e, portanto, da *SLA*.

Um bom exemplo, nesse caso, seria o controle de admissão, limitando o número máximo de chamadas simultâneas. Isso evitaria que aplicações *VoIP* demandassem da rede recursos tais a ponto de prejudicar as demais aplicações tidas como críticas. Somente limitar os recursos fornecidos poderia, ao invés de impedir novas chamadas, permiti-las, mas com degradação de desempenho, o que dificilmente seria desejado. Outro fator que serviria de exemplo para casos em que a prioridade de aplicações de dados devem sobrepor a *VoIP* é a disponibilidade. Pode-se supor que, em caso de falha em um enlace principal, estando a rede funcionando por um enlace de contingência e de menor velocidade, seja preferível interromper os serviços de voz, garantindo *performance* para os dados críticos.

4.4 Requisitos de *QoS* para *VoIP* (voz sobre *IP*)

A integração de *VoIP* requer fornecimento de prioridade e garantias explícitas para o serviço, além de requerer banda para o tráfego de sinalização. Essas duas classes foram examinadas separadamente.

4.4.1 Voz

A lista a seguir resume os requisitos e recomendações de *QoS* para o tráfego de voz:

- Pacotes de voz devem ser marcados com *DSCP EF* (DAVIE et al., 2002);
- Taxa de perda de pacotes não deve ser maior que 1%;
- O atraso em um sentido não deve ser maior que 150 ms;
- O *jitter* deve ser menor que 30 ms;
- A banda para as chamadas deve ser garantida, sendo o requisito dependente do tipo de amostragem, CODEC utilizado e *overhead* de camada 2.

A qualidade da voz é diretamente afetada pelos três principais parâmetros de *QoS*: perdas, atraso e *jitter*.

Taxa de Perdas – Provocam cortes nas chamadas, prejudicando a conversação. Para atenuar os efeitos das perdas de pacotes, são utilizados mecanismos específicos, conhecidos como *PLC* (*Packet Loss Concealment)*. Esses mecanismos mascaram os efeitos das perdas ou descartes de

pacotes *VoIP*. O método de *PLC* usado depende do tipo de CODEC. Uma das formas mais simples e mais frequentemente utilizadas consiste em repetir a última amostra recebida, aumentando a atenuação em cada repetição. Essa técnica é adequada para CODECs menos complexos, que usam forma de onda (*waveform CODECs),* como o G.711. A técnica é eficiente para compensação de perdas em casos de amostras com até 20 ms.

O intervalo de geração de pacotes (*packetization interval*) determina o tamanho das amostras contidas em um pacote. Assumindo que esse valor seja de 20 ms (padrão), a perda de dois ou mais pacotes provocam degradação na qualidade da voz. Entretanto, assumindo uma distribuição aleatória das perdas dentro de um mesmo fluxo de voz, a probabilidade de que uma perda não possa ser compensada torna-se bastante pequena e perfeitamente aceitável.

CODECs de baixa taxa de *bits*, baseados em *frame*, como o G.729 e G.723, utilizam técnicas de *PLC* mais sofisticadas, que podem compensar entre 30 e 40 ms de perdas, mantendo a qualidade tolerável. Nesses casos, o intervalo de empacotamento determina o número de *frames* carregados por pacote. Como no G.711, se esse tempo é maior que a perda, o algoritmo *PLC* pode interpolar. O *PLC* não pode compensar a perda de um pacote.

Os projetos de redes para transporte de *VoIP* normalmente consideram a taxa de perdas muito próximas de zero. Valores entre 1 e 2% são, tipicamente, utilizados como referência.

Atraso (latência) – se excessivo, provoca degradação da qualidade de voz. A meta normalmente usada em projetos é manter o atraso unidirecional em menos de 150 ms (*ITU* G.114). Devem ser

considerados os vários componentes do atraso: atraso de rede (propagação pelo *backbone*, de escalonamento, por conta do congestionamento e de serialização no enlace de acesso) e atraso de serviço (*VoIP* CODEC e *buffer* de *jitter*).

Jitter – Buffers de *jitter*, também conhecidos como *playout buffers*, são usados para mudar a chegada assíncrona de pacotes de um fluxo síncrono, tornando atrasos de rede variáveis em atrasos constantes no destino. O *buffer* de *jitter* trabalha entre o atraso e a probabilidade de que ocorram falhas na comunicação devido a pacotes atrasados.

Há de se considerar que, se o *buffer* for muito grande, imporá à rede contenções muitas vezes desnecessárias, acentuando o atraso fim-a-fim, ainda que a rede disponha de atraso menor. Em contrapartida, se for muito pequeno para acomodar o *jitter*, pode ocorrer estouro ou incapacidade do *buffer* (*overflow* e *underflow*). No *underflow*, o *buffer* está vazio quando o CODEC precisa processar uma amostra. No *overflow*, o *buffer* já está cheio quando outro pacote chega, o que impede o enfileiramento desse novo pacote. Ambos causam degradação da qualidade de voz.

Para solucionar esses problemas, são utilizados algoritmos de *buffer* de *jitter* adaptativos, que se ajustam dinamicamente à necessidade da rede. Normalmente, esses algoritmos operam da seguinte forma:

- Instantaneamente, aumentam o tamanho do *buffer* para o valor de *jitter* corrente medido, seguindo um *buffer overflow*;
- Gradativamente, diminuem o tamanho do *buffer* quando o *jitter* atual medido é menor que o tamanho do *buffer*;

- Utilizam *PLC* para interpolar, compensando perdas de pacotes em caso de *buffer underflow*.

Por conta dos requisitos estritos de níveis de serviço, os pacotes de voz devem ser acomodados no *PHB EF*, sendo marcados, portanto, com *DSCP EF* (46). Além disso, principalmente para minimizar o atraso, é recomendado que se utilizem filas específicas e de alta prioridade no despacho de pacotes.

O requisito de banda para fluxos de *VoIP* é calculado adicionando-se ao tamanho da amostra, em *bytes*, os 40 *bytes* referentes aos cabeçalhos do *IP*, *UDP* e *RTP*, assumindo que o *cRTP* (*compressed RTP* – RTP comprimido) não esteja habilitado. Se esse valor é multiplicado por 8, obtém-se o valor em *bits* que, finalmente, deve ser pela taxa de geração de pacotes (*packetization rate*), cujo valor padrão é de 50 pacotes por segundo.

Considerando, por exemplo, a utilização do CODEC G.711 para *VoIP*, com a taxa de geração de pacotes padrão de 50 pps, um novo pacote é gerado a cada 20 ms (1 segundo / 50 pps). A carga de cada pacote é 160 *bytes*. Com os cabeçalhos *IP* (20 *bytes*), *UDP* (8 *bytes*) e *RTP* (12 *bytes*), esse pacote chega ao tamanho de 200 *bytes*. Convertendo-se o valor em *bits*, tem-se 1600 bps por pacote. Como em um segundo são gerados 50 pacotes, a banda de camada de rede requerida para o fluxo é 80 Kbps. Esse valor não inclui o *overhead* da camada de enlace e também não considera a utilização de mecanismos de eficiência de *links*, como compressão de cabeçalhos (*cRTP*).

Para que sejam obtidos valores mais precisos no provisionamento de banda para o tráfego de *VoIP*, deve-se incluir nos cálculos o *overhead* do enlace, que depende da tecnologia utilizada (tabela 4-1).

Tabela 4-1: *Overhead* da Camada de Enlace

Enlace	*Overhead*
802.1Q *Ethernet*	Até 32 *bytes* (com preâmbulo)
PPP – Point-to-Point Protocol	12 *bytes*
MLP – Multilink PPP	13 *bytes*
Frame Relay	4 *bytes*
Frame Relay com *FRF*.12	8 *bytes*

4.4.2 Sinalização

A lista a seguir resume os requisitos e recomendações de *QoS* para o tráfego de sinalização de chamadas:

- Pacotes de sinalização devem ser marcados com *DSCP CS3* ou *AF31* (DAVIE et al., 2002);
- Requer, tipicamente, 150bps (mais *overhead* de enlace) por chamada para tráfego de controle de voz. Mais banda pode ser requerida, dependendo do protocolo de sinalização utilizado.

Em certos casos, pode ser interessante agregar em uma mesma classe de serviço os tráfegos de voz e sinalização. Isso se justifica, sobretudo, quando o tráfego resultante da sinalização para as chamadas de voz é inexpressivo, o que ocorre quando a quantidade de canais de voz ou chamadas é pequena. Definir uma só classe simplifica o modelo, facilitando a administração, e evita a

definição de uma classe com necessidade de reserva de recurso irrisória em relação à banda total disponível. Nesses casos, os recursos necessários aos dois tipos de tráfego são somados e utiliza-se prioritariamente a marcação com *DSCP EF*. No estudo de caso apresentado, diante da pequena quantidade de canais por ponto remoto, optou-se por agregar todo o tráfego de *VoIP* (voz + sinalização) em uma só classe.

Para as redes que comutam grandes quantidades de chamadas simultaneamente, entretanto, deve-se manter mídia e sinalização em classes independentes. Esse é o caso, por exemplo, das redes de operadoras de telecomunicações que utilizam *VoIP* para transporte de voz.

4.5 Requisitos de *QoS* para vídeo

São dois os principais tráfegos de vídeo: vídeo interativo (videoconferência) e vídeo unidirecional – *streaming* (*unicast* ou *multicast*). Cada um dos dois foi examinado separadamente:

4.5.1 Vídeo interativo

A lista a seguir resume os requisitos e recomendações de *QoS* para o tráfego de videoconferência:

- Pacotes devem ser marcados com *DSCP AF41*;
- Taxa de perdas não deve ser maior que 1%;
- Atraso unidirecional de até 150 ms;

- *Jitter* menor que 30 ms.

A taxa nominal da videoconferência corresponde à taxa de amostragem do fluxo de vídeo e voz e não ao requisito de banda da rede para funcionamento da aplicação. Dessa forma, para calcular a banda total a ser reservada para a videoconferência, é necessário que sejam adicionados à taxa nominal os *overheads* do *IP*, *UDP*, *RTP* e camada 2. Como o tamanho dos pacotes e a taxa variam, a porcentagem correspondente a *overhead* também varia, o que inviabiliza o cálculo preciso para todos os fluxos. Um valor admitido como seguro e usado como padrão para o *overhead* é 20%. Assim, para que uma videoconferência de 384Kbps de taxa seja atendida adequadamente, deve ser fornecida uma banda de 460Kbps.

4.5.2 *Video streaming*

A lista a seguir resume os requisitos e recomendações de *QoS* para o tráfego de vídeo unidirecional:

- Marcação de pacotes com *DSCP CS4*;
- Taxa de perda de pacotes inferior a 5%;
- O atraso deve estar entre, no máximo, 4 e 5 segundos, dependendo da capacidade de bufferização da aplicação;
- Não há requisitos significantes quanto ao *jitter*;
- Garantias de banda dependem do formato de codificação e da taxa do fluxo de vídeo;

- Aplicações de vídeo sem propósito organizacional (*unicast* ou *multicast*) devem ser classificadas como tráfego indesejado e, portanto, ter os pacotes marcados como *DSCP CS1*. Para essa classe, deve ser reservada uma mínima porcentagem da banda. É importante excluir desses casos vídeos com conteúdo relevante, tais como *e-learning* e palestras, casos em que são requeridas garantias de serviço.

4.6 Requisitos para tráfego de dados

As características do tráfego de dados variam de acordo com a aplicação. Por isso, é crucial, apesar de muitas vezes complexo, que se conheça o comportamento padrão das aplicações utilizadas na rede. Sobretudo, é importante que se consiga diferenciar o modo de operação e o nível de importância para o negócio das aplicações que tocam diretamente que afetam diretamente os processos internos da empresa e que, normalmente, terão tratamento diferenciado no projeto de *QoS*.

As aplicações de dados se dividem em quatro classes principais: *best effort*, transferência de dados (*bulk data*), dados transacionais ou interativos e aplicações de missão crítica.

4.6.1 Dados *best effort* (sem prioridade)

- O tráfego deve ser marcado com *DSCP* 0;

- Deve ser fornecida banda adequada para o tráfego *best effort*, visto que, normalmente, a maior parte das aplicações identificadas pertence a essa classe.

A classe *best effort* é a classe padrão para os fluxos de dados. Somente se a aplicação for selecionada para ter tratamento preferencial ou diferenciado, ela será removida da classe *best effort*. Apesar de não ter prioridade em relação às demais classes, com exceção da classe de tráfego identificado como não cooperativo, a classe *best effort* normalmente acomoda a maior parte das aplicações, o que, em situações de congestionamento, provoca grande insatisfação dos usuários. Por esse motivo, é recomendado que se garanta banda para a classe e, apesar de controvérsias existentes quanto à quantidade da banda reservada, 25% do total tem sido considerado como dimensionamento padrão.

4.6.2 Transferência de arquivos (*bulk data*)

As seguintes recomendações podem ser usadas para essas aplicações:

- Os pacotes devem ser marcados como *DSCP AF11*. O tráfego excedente da classe pode ser remarcado para *AF12* ou *AF13*;
- Deve ter alguma garantia de banda, mas o tráfego deve ser controlado para não provocar congestionamento no enlace.

Essa classe é planejada para aplicações que têm pouca interatividade e não são sensíveis a descartes. Essas aplicações tipicamente prolongam sua operação por longos períodos de

tempo e podem funcionar em segundo plano (*background*). Incluem *FTP*, *e-mail*, operações de *backup*, sincronização de bases de dados ou serviços de replicação, distribuição de conteúdo e qualquer outra aplicação cujos usuários não necessitem intervir enquanto aguardam a finalização do processo.

A vantagem em prover alguma garantia de banda para essas aplicações, ao invés de somente policiar seu tráfego, é que elas podem utilizar banda eventualmente não usada das outras classes, aumentando a *performance* em momentos de baixo tráfego.

4.6.3 Dados transacionais e dados interativos

Ao tráfego de dados transacionais sugerem-se as seguintes recomendações:

- Marcação do tráfego com *DSCP AF21*; o tráfego excedente pode ser remarcado pelo mecanismo de policiamento para *AF22* ou *AF23*;
- Fornecimento de banda adequada para garantir a interatividade da aplicação.

Essa classe de aplicações combina dois tipos de aplicações similares: aplicações transacionais cliente-servidor e aplicações de mensagens interativas. O requisito de tempo de resposta separa a classe das aplicações cliente-servidor genéricas. Por exemplo, nesse tipo de aplicação, como *SAP* e *softwares* de apoio a decisão, o usuário aguarda que a operação seja concluída antes de prosseguir, o que sugere comportamento transacional.

4.6.4 Dados de missão-crítica

Para essa classe sugere-se:

- Marcação do tráfego com *DSCP AF31*. O tráfego excedente pode ser marcado pelo mecanismo de policiamento como *AF32* ou *AF33*;
- Banda adequada, para que se forneça tempo de resposta satisfatório e compatível com o nível de interatividade desejado.

Essa classe deve abranger aplicações próprias ao negócio em questão e críticas ao resultado final. Ou seja, deve acomodar o tráfego da aplicação de dados mais importante para o negócio. Como se pode notar, trata-se de uma definição não técnica a decisão sobre que aplicações serão tratadas como de missão-crítica, envolvendo questões políticas e organizacionais.

Recomenda-se, ainda, que seja associado o menor número possível de aplicações a essa classe.

4.6.5 Considerações sobre *DLSW – Data Link Switching*

DLSW é uma técnica utilizada para transportar uma conexão *SNA* (*Systems Network Architecture*) ponto-a-ponto ou *NETBIOS* por um *socket TCP/IP*. Esse padrão oferece várias vantagens:

- Pode ser usada na substituição de conexões diretas, sem que seja necessária nenhuma modificação nos pontos finais ou nas aplicações;

- Pode substituir, não somente conexões entre nós *SNA* periféricos – *PU2*, mas também conexões entre *PU4* pré-*APPN* e nós *PU5*.

Existem também limitações importantes:

- Não substitui nenhum nó da rede *SNA*, consistindo simplesmente em um túnel de transporte. Por isso, normalmente aumenta a complexidade da rede e os potenciais pontos de falha;
- Acrescenta *overhead* significativo, visto que o tráfego orientado a conexão *SNA* de camada 2 é encapsulado no *TCP*;
- Não há como estabelecer níveis de serviço diferenciados por sessão. Todas as sessões serão tratadas da mesma forma dentro da rede *IP*, tendo a *QoS* assegurada a todo o tráfego *DLSW*, sem prioridades individuais para cada sessão.

Em resumo, *DLSW* é uma solução ideal quando se pretende compartilhar rede *IP* com tráfego legado *SNA*, eliminando a necessidade de estrutura de comunicação exclusiva, sem que modificações sejam necessárias nos pontos finais da conexão.

O tráfego *SNA* é normalmente associado ao negócio final da empresa e transporta aplicações de extrema criticidade. Essas aplicações, muitas *on-line*, têm requisitos de atraso e banda, mas não são sensíveis a *jitter*. Desse modo, essas seriam as recomendações para o tráfego *DLSW*:

- Identificação do tráfego através dos endereços de origem e destino dos *DLSW peers*, através das portas *TCP* (entre 1981 e 1983 e 2065) ou pela identificação direta do protocolo encapsulado;
- Marcação com *AF21* (transacional) ou tráfego de missão-crítica (*DSCP* 25), o que melhor convier para a rede em questão;
- Deve ser fornecida banda suficiente à classe, de acordo com o perfil da aplicação e de modo a evitar descartes de pacotes.

Algumas implementações proprietárias, como o *DLSW+* da Cisco Systems, reconhecem o tráfego *DLSW* e fazem a marcação automaticamente, utilizando o padrão antigo do campo *ToS* definido na RFC-791, com *IP Precedence* 5. O recomendado é que essa marcação seja desabilitada para que não interfira no funcionamento da classe associada ao tráfego de tempo real.

4.6.6 Plano de controle

QoS torna-se irrelevante se a rede não funcionar. Diante dessa afirmação óbvia, garantir recursos para tráfego de controle, que inclui roteamento e gerência de rede, é essencial.

4.6.6.1 Tráfego de roteamento

- O tráfego de roteamento *IP* deve ser marcado com *DSCP CS6*. Esse comportamento é o padrão adotado pela maioria dos fornecedores de equipamentos;

- Protocolos de roteamento internos normalmente são resguardados por mecanismos de priorização automáticos, internos ao roteador. Para os protocolos de roteamento externos, como *BGP – Border Gateway Protocol –* recomenda-se que seja criada uma classe explícita com banda mínima garantida.

4.6.6.2 Tráfego de gerenciamento e suporte

- Recomenda-se a marcação do tráfego de gerência com *DSCP CS2*;
- As aplicações de gerenciamento devem ser protegidas explicitamente com a reserva de banda mínima para funcionamento.

A garantia de banda para essas aplicações garantirá a execução de tarefas de análise de *performance* e identificação/resolução de problemas. Além disso, é importante que se tenha garantias para o tráfego de aplicações como *SNMP*, *NTP* e *Syslog*, dentre outras.

4.6.7 Tráfego *worst effort* (*scavenger, less than best effort*)

Essa classe visa segregar aplicações indesejáveis da classe *default*, associando prioridade inferior à da classe *best effort.* Tais aplicações normalmente trazem muito pouca ou nenhuma contribuição aos objetivos corporativos e tipicamente dizem respeito a entretenimento. Incluem *peer-to-peer* (*Kazaa, eMule* etc.) e jogos em rede, dentre outras. Associar essas aplicações a uma fila de banda mínima faz com que o tráfego gerado virtualmente deixe de ter recursos em

momentos de congestionamento, mas permite que estejam disponíveis se a banda não estiver sendo utilizada para finalidades do negócio.

4.6.7.1 Estratégia de combate a *DoS* (*Denial of Service*) e *worms* com a classe *worst effort* (*scavenger*)

Nos últimos anos, tem-se observado grande aumento na freqüência e no nível de sofisticação de ataques do tipo *DoS* e de ataques com uso de *worms*. Além disso, os danos causados por esses ataques também são maiores a cada dia (NEUMAN, 2000).

A proposta mostrada aqui consiste em evitar que ataques desses tipos gerem grandes impactos aos serviços da rede e se baseia em policiar o tráfego dos equipamentos finais, fazendo a remarcação para *DSCP CS1* (*scavenger*), caso o volume gerado seja considerado anormal. Tipicamente, tráfegos constantes maiores que 5% da velocidade nominal da porta são considerados anormais (SZIGETI et al, 2004).

Outra alternativa funcionaria bem nos casos em que a utilização de aplicações em rede é totalmente controlada. Qualquer nova aplicação, não identificada ou relacionada na matriz de aplicações, seria automaticamente marcada com *DSCP CS1*. Essa ação protegeria as classes críticas e também a classe *best effort*, evitando a degradação das aplicações acondicionadas nessa classe.

5 MECANISMOS DA *DIFFSERV*

Nesse capítulo, serão detalhadas as funções necessárias à implementação da *QoS* fim-a-fim com o modelo *DiffServ*. Essa arquitetura prevê blocos específicos trabalhando em conjunto para a execução das funções de condicionamento de tráfego (*TC*) e *PHB* (*Per Hop Behavior*). A figura 5-1 a seguir mostra a estrutura funcional descrita em (BLAKE, 1998).

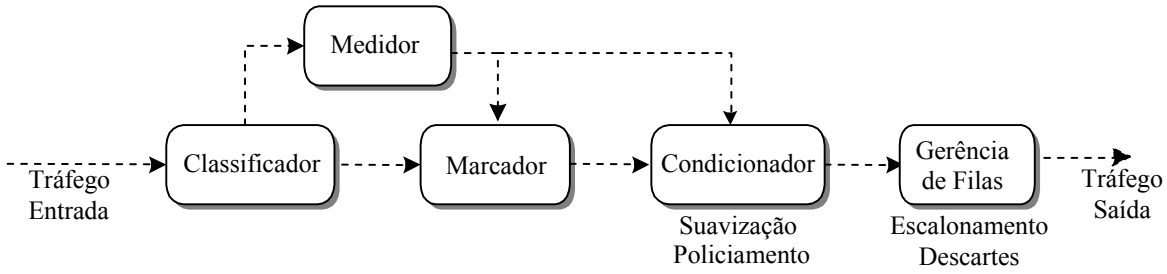


Figura 5-1: Mecanismos da *DiffServ*

5.1 Importância dos elementos roteadores

As funções básicas de um roteador incluem receber pacotes por uma interface de entrada, decidir a melhor opção de rota e encaminhar esses pacotes para uma interface de saída, fazendo o encapsulamento de camada 2 correspondente ao enlace conectado. Além disso, deve executar os mecanismos necessários para a montagem e manutenção das tabelas de rotas.

Contudo, os roteadores atuais precisam fazer muito mais que as tarefas básicas citadas. Dentre essas atividades adicionais, associadas ou não às funções necessárias à implementação da *QoS* fim-a-fim, estão as seguintes:

- Compressão de *payload* (*stack, predictor* etc.);
- Compressão de cabeçalhos (*TCP* e *RTP header compression*);
- Policiamento e suavização do tráfego;
- Marcação de pacotes;
- Descartes seletivos;
- Gerenciamento de filas com algoritmos complexos;
- Criptografia;
- Roteamento baseado em políticas;
- Contabilização de pacotes (*IP accounting, NetFlow accounting*);
- Filtros (*ACLs - access control lists*);
- Tradução de endereços e portas (*NAT*, *PAT*).

Os roteadores têm papel fundamental no sucesso do projeto de *QoS*, pois participam de todas as etapas da implementação da *DiffServ*. São opcionais apenas a classificação e marcação do tráfego, que podem ser feitas anteriormente, seja nos equipamentos finais de usuário, seja nos concentradores de rede local (*switches*).

Desse modo, todas as funções detalhadas a seguir (classificação e marcação, gerenciamento de filas, policiamento e moldagem, mecanismos usados para evitar congestionamento e mecanismos de eficiência de enlaces) são normalmente implementadas nos roteadores.

5.2 Classificação e marcação

A tarefa de classificar o tráfego consiste em agrupar fluxos em agregados, a fim de receberem a mesma marcação e posterior tratamento no domínio *DiffServ*. Somente após a identificação do tráfego, as políticas podem ser aplicadas a ele. Essa tarefa pode se basear nos seguintes critérios:

- Camada 1 – Parâmetros como interface física, sub-interface, *PVC* (*permanent virtual* circuit – circuito virtual permanente) etc. associados ao tráfego a classificar;
- Camada 2 – endereço *MAC*, campo *CoS* (*class of service*) 802.1Q/p do quadro, identificação de *VLAN* (*virtual LAN*), *MPLS EXP* (*experimental bits*)[5], *ATM CLP* (*cell loss priority*) e *bit DE* (*discard eligible*) do quadro *Frame Relay;*
- Camada 3 – endereços *IP* de origem e/ou destino, análise do campo *ToS* (*IP Precedence*) ou *DiffServ code point* (*DSCP*);
- Camada 4 – verificação de portas *UDP* ou *TCP*;
- Camada 7 – verificação de assinatura das aplicações e informações contidas nos cabeçalhos ou *payloads.*

[5] Como *MPLS* encapsula o pacote antes do encapsulamento no quadro de camada, alguns autores referem-se à tecnologia como de camada 2,5.

O recomendado, para que se mantenha a granularidade e capacidade de escala da arquitetura, é que, tanto a classificação, quanto a marcação (com valores *DSCP*) sejam feitas o mais próximo possível da origem do tráfego. Se a marcação é feita de forma correta, os elementos intermediários não precisam repetir a mesma classificação. Ao contrário, podem administrar as funções de aplicação das políticas, como escalonamento, baseando-se nas marcações previamente feitas, nos equipamentos de entrada da rede.

A seguir serão citados os principais mecanismos usados na marcação do tráfego:

– *Ethernet* 802.1Q/p

Como mostrado na figura 5-2, os quadros *Ethernet* podem ser marcados de acordo com a sua importância através dos *bits* de prioridade 802.1p (*CoS*) do cabeçalho 802.1Q.

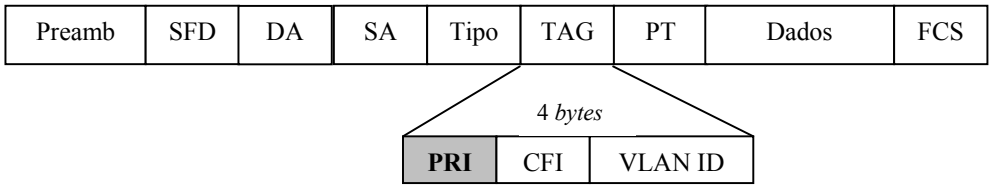


Figura 5-2: Quadro *Ethernet*: 802.1Q/p *CoS*

Somente 3 *bits* estão disponíveis para marcação 802.1p e, com isso, 8 classes de serviço podem ser definidas para quadros *Ethernet*. Os valores de *CoS* são idênticos aos de *IP Precedence* e tipicamente são atribuídos de acordo com a tabela 5-1.

Como os valores são os mesmos, pode-se fazer o mapeamento de *IP Precedence* para *CoS* e vice-versa. Entretanto, valores de *DSCP* (compostos por 6 *bits*) não podem ser mantidos

quando mapeados para valores 802.1Q/p *CoS*. Por isso, pode haver perda de granularidade e algumas classes podem ser perdidas na tradução.

Tabela 5-1: Valores de *CoS*/*IP Precedence* por tipos de aplicação

Valor *CoS* / *IP Precedence*	Aplicação
7	Reservado
6	Reservado
5	Voz
4	Videoconferência
3	Sinalização de chamadas
2	Dados de alta prioridade
1	Dados de média prioridade
0	Dados *best effort*

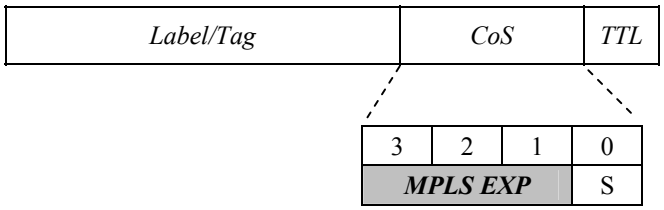
- *Frame-Relay Discard Eligible* e *ATM Cell Priority*

O *bit* DE do campo endereço do cabeçalho *Frame Relay* e o *bit CLP* da célula *ATM* têm o mesmo propósito. São usados para sinalizar que quadros ou células são menos importantes e, portanto, elegíveis ao descarte em caso de congestionamento. Somente dois valores são possíveis: 0 e 1. Em casos de congestionamento, quadros ou células com *DE* ou *CLP* marcados com 1 serão descartados antes daqueles marcados com zero.

- *MPLS Experimental*

O rótulo (*label*) *MPLS* contém três *bits* para diferenciação de classes de serviço, chamados de *MPLS EXP*. Os valores possíveis de *MPLS EXP* são os mesmos do 802.1Q/p e *IP Precedence*

(figura 5-3) e, com isso, pode-se fazer o mapeamento dessas marcações para o *MPLS EXP*. Valores *DSCP* não podem manter a mesma granularidade, conforme citado no item anterior.



Label/Tag: 20 *bits*
MPLS Experimental (*CoS*): 3 *bits*
Stack indicator - indicador de pilha de *labels* (S): 1 *bit*
Time-to-Live – tempo de vida (*TTL*): 8 *bits*

Figura 5-3: *MPLS EXP bits* no *Label MPLS*

A figura 5-4 a seguir (SZIGETI et al, 2004) mostra a relação entre as marcações *IP* (*IP Precedence* ou *DSCP*) e *MPLS EXP*.

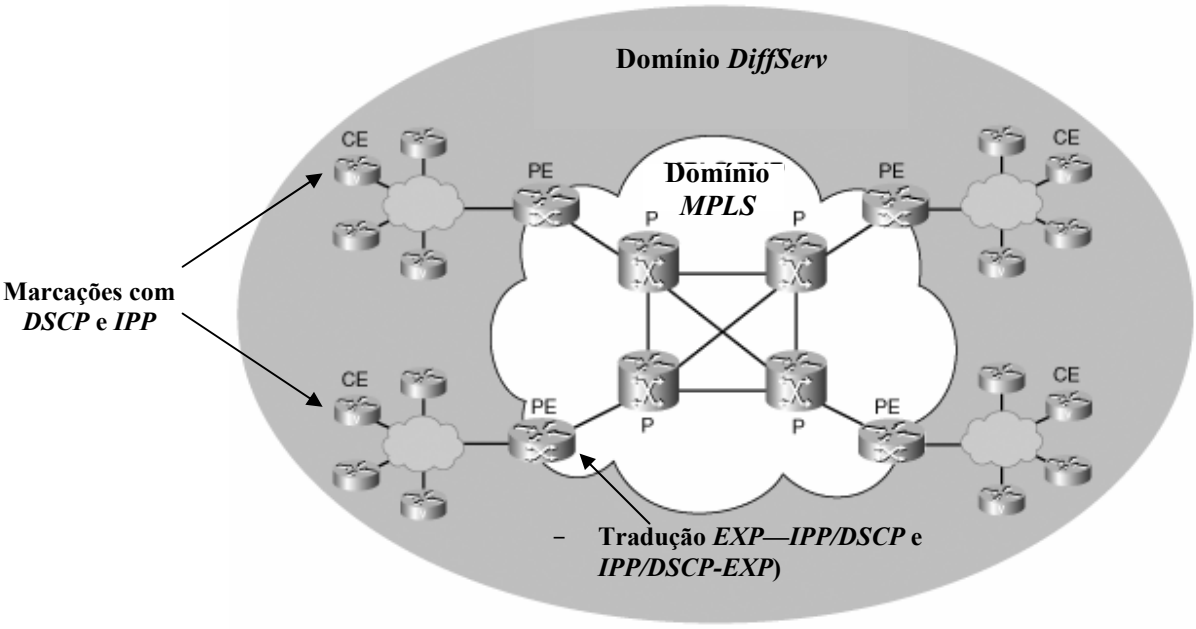


Figura 5-4: Relação entre marcações *IP* e *MPLS*

Fonte: SZIGETI et. al, 2004 (adaptado)

Apesar dos mecanismos de marcação em camada 2, a marcação em camada 3 é mais usual, sobretudo por manter significância fim-a-fim, sem a necessidade de traduções intermediárias. A marcação em camada 3 utiliza o campo *ToS* (*Type of Service*) do pacote *IP*, conforme descrito na sessão 4.2 (Serviços Diferenciados) e, preferencialmente, deve utilizar os 6 *bits* do campo *DSCP*.

A marcação com *IP Precedence* deve ser usada somente em casos de necessidade de interoperabilidade com equipamentos que não suportem *DSCP* (figura 5-5).

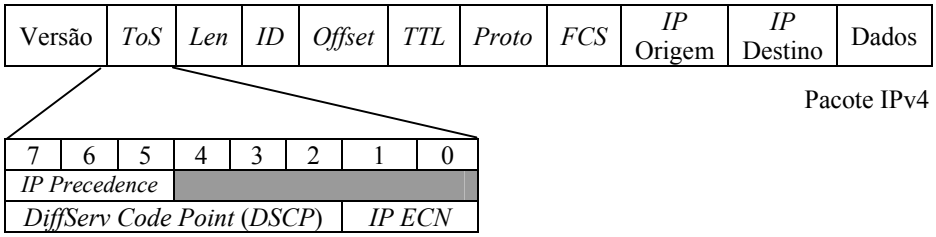


Figura 5-5: Campo ToS (Type of Service) do pacote IPv4 (bits IP Precedence e DSCP)

A tabela 5-2, a seguir, mostra um resumo com as opções de marcação em camadas 2 e 3.

Tabela 5-2: Resumo de opções de marcação em camadas 2 e 3

Tecnologia	**Camada**	**Campo de Marcação**	**Número de Bits**	**Valores**
Ethernet	2	802.1Q/p	3	0 a 7
Frame Relay	2	*DE bit*	1	0 a 1
ATM	2	*CLP bit*	1	0 a 1
MPLS	2	*EXP*	3	0 a 7
IP	3	*IP Precedence*	3	0 a 7
IP	3	*DSCP*	6	0 a 63

5.3 Gerência de filas e escalonamento

Mecanismos de gerenciamento de filas e escalonamento são necessários para acomodar o tráfego quando a taxa de chegadas de pacotes é maior que a capacidade de transmissão. Isso ocorre em um dos seguintes casos:

- A interface de entrada é mais rápida que a de saída.
- A interface de saída recebe pacotes de múltiplas interfaces.

Nesta seção, serão mostrados os principais mecanismos de escalonamento e gerência de filas usados em roteadores, que se constituem na mais importante das funções por afetarem os quatro principais índices de *QoS*: banda, atraso, *jitter* e perdas de pacotes.

Conceitualmente, gerência de filas e escalonamento são processos complementares, mas distintos (ODOM, 2004):

- Gerência de filas define a lógica utilizada para ordenar pacotes nos *buffers* de saída de uma interface. Esse processo somente entra em operação quando a interface (ou *link*) está congestionada e deixa de atuar tão logo cesse o congestionamento.

- Escalonamento é o processo que decide qual será o próximo pacote transmitido. Diferentemente dos mecanismos de gerência de filas, atua tanto no congestionamento, quanto em situações de tráfego normal. Se não há congestionamento, os pacotes são transmitidos na ordem que chegam. Se há congestionamento, os mecanismos de filas entram em operação. O

escalonador sempre tem que decidir que fila será servida primeiro. Essa decisão é tomada a partir da utilização de diversos algoritmos de escalonamento lógico:

- *Strict Priority* – As filas de baixa prioridade serão servidas somente quando as filas de alta prioridade estão vazias. Esse tipo de escalonamento pode ocasionar total falta de recursos para as filas menos prioritárias, o que é conhecido como *starvation*;

- *Round-robin* – As filas são servidas em seqüência. Apesar de sempre fornecer recursos a todas as filas, podem gerar atrasos imprevistos para pacotes carregados nas filas que carregam tráfego de tempo real e demais tráfegos sensíveis ao atraso, que necessitariam de tratamento prioritário;

- *Weighted fair* – Os pacotes nas filas recebem um peso, normalmente de acordo com a prioridade. As filas com pacotes de maior prioridade são servidas mais frequentemente que as demais. Apesar de diminuir os problemas mostrados nas duas estratégias citadas anteriormente, não provê garantias de banda, requeridas por fluxos de tempo real. O banda resultante por fluxo varia, dependendo do número de fluxos presentes e do peso de cada um deles.

A quantidade de *buffers* (memória) para filas é finita. Quando esse recurso acaba, os pacotes podem ser descartados na chegada (*tail drop*) ou, seletivamente, antes que todos os *buffers* se esgotem. Esse último processo está relacionado aos mecanismos usados para evitar o congestionamento (*congestion avoidance*), descritos mais detalhadamente na seção 5.5. Esses

mecanismos funcionam melhor com aplicações baseadas em *TCP*. Os descartes seletivos fazem com que o *TCP* ajuste o tamanho das janelas, diminuindo a taxa de transmissão para valores menores. Essa reação tende a diminuir, em conseqüência, o congestionamento.

Os mecanismos de prevenção a congestionamento complementam os mecanismos de gerência de filas. A relação entre os dois, de forma simplificada, consiste em que os mecanismos de gerência do congestionamento (filas e escalonamento) gerenciam a frente da fila, enquanto os mecanismos de prevenção (descartes) gerenciam o final da fila.

O gerenciamento de filas se constitui hoje em um mecanismo fundamental para a diferenciação dos serviços em classes distintas. Nessa matéria, uma das tarefas mais complexas é definir o tamanho ideal das filas, sobretudo devido às características tão distintas dos diversos tipos de tráfego. Filas muito grandes diminuem a probabilidade de descarte de pacotes, já que terão mais recursos (*buffers*) disponíveis para acomodá-los. Entretanto, pacotes enfileirados poderão demorar mais para serem despachados, o que aumenta a variação do atraso (*jitter*), crítico às aplicações de tempo real. Em contrapartida, filas muito pequenas disponibilizam poucos *buffers*, diminuindo o *jitter*, mas aumentando a probabilidade de descartes.

As implementações iniciais de gerência de filas usavam a estratégia simples *FIFO* (*first-in first-out*), que significa que o primeiro pacote a chegar será o primeiro a ser servido, não levando em consideração o conceito de prioridade ou classes de tráfego. Conseqüentemente, não afeta na ordem de saída do pacote. Por não implementar diferenciação entre os tipos de tráfego, o mecanismo *FIFO* pode ser útil em redes onde não há escassez de recursos.

Mecanismos mais complexos foram introduzidos, sobretudo a partir de quando requisitos de serviço específicos fizeram com que os roteadores precisassem identificar, diferenciar e priorizar pacotes de importância distintas.

Os mecanismos podem ainda ser divididos em duas partes:

- Filas de *Hardware* – ainda utilizam a estratégia *FIFO*, necessária para que os *drivers* das interfaces transmitam pacotes um a um. São também conhecidas como filas de transmissão ou *TxQ*;
- Filas de *Software* – fazem o escalonamento dos pacotes baseado nos requisitos de *QoS*.

Os mecanismos básicos são os listados a seguir:

- *PQ* (*Priority Queueing*) – Consiste na definição de filas de prioridades diferentes (alta, média, normal e baixa, por exemplo). O escalonador serve as filas por ordem de prioridade até que estejam completamente vazias (figura 5-6). Para aplicações de alta prioridade, o mecanismo funciona de forma satisfatória, porém, para as filas de prioridade inferior, tende a provocar falta de recursos (*starvation*).

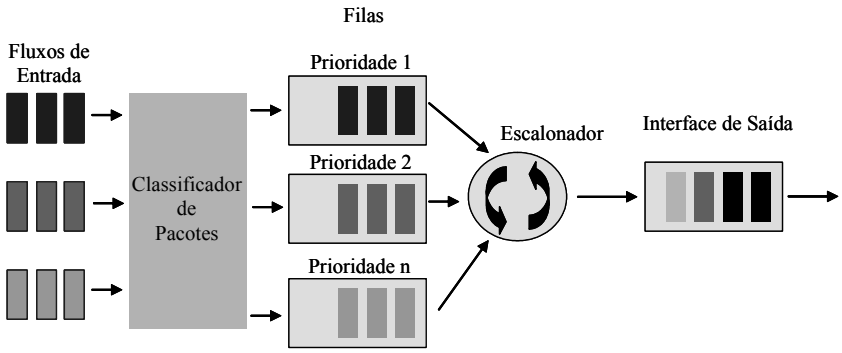


Figura 5-6: Escalonador *Priority Queueing (PQ)*

– *CQ* (*Custom Queueing*) ou *CBQ* (*Class Based Queueing)* ou *WRR* (*Weighted Round Robin*) - Surgiu para resolver os problemas do *PQ*, relacionados à falta total de recursos para algumas filas. Introduziu o escalonador cíclico (*round-robin*) baseado em contagem de *bytes* (figura 5-7). Além de prevenir a falta total de banda para as classes menos prioritárias, foi o primeiro mecanismo a fornecer garantias efetivas de banda. Apesar de ser um mecanismo mais justo que o *PQ*, perde a capacidade de fornecer priorização total para fluxos de tempo real.

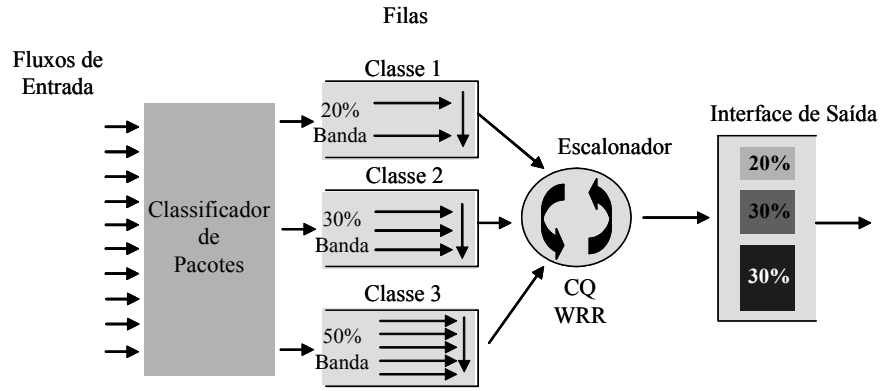


Figura 5-7: Escalonador *Custom Queueing (CQ)*

– *FQ (Fair-Queueuing)* e *WFQ* (*Weighted Fair Queueing*) – O algoritmo *fair-queueing* (*FQ*), utilizado pelo *WFQ*, divide a banda da interface pelo número de fluxos, garantindo uma distribuição justa para todas as aplicações (figura 5-8). O *FQ* se baseia no fato de que cada pacote consome certo tempo para ser transmitido (*finish time*), sendo esse tempo proporcional ao tamanho do pacote. Como cada pacote faz parte de um fluxo (origem, destino, aplicação), o *FQ* consegue organizar as filas de forma automática e dinâmica. Como os fluxos de pacotes maiores levam mais tempo para serem transmitidos (maior *finish time*), o *FQ* prioriza os fluxos de pacotes menores (menor *finish time*), encaminhando-os antes. Por adicionar um peso, baseado na prioridade do pacote (*IP Precedence*), ao *FQ*, o *WFQ* criou um método para

favorecer fluxos de alta prioridade. Entretanto, perde a capacidade de garantir banda do *CQ*, pois a alocação de recursos varia constantemente, à medida que fluxos são iniciados e finalizados.

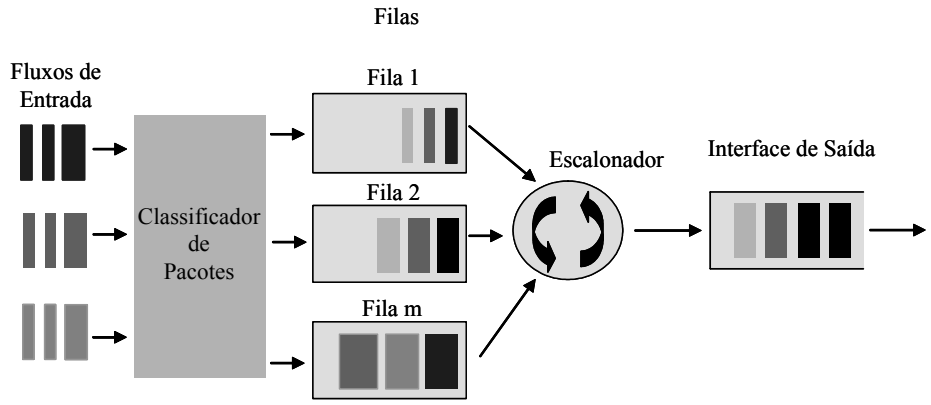


Figura 5-8: Escalonador *Weighted Fair Queueing (WFQ)*

Cada um dos mecanismos básicos e abertos descritos anteriormente apresenta vantagens e desvantagens entre si e todos apresentam problemas para manipular o tráfego originado de múltiplas mídias e com diferentes requisitos das redes convergentes. Foram criados, então, mecanismos proprietários que trabalham de acordo com o modelo das aplicações transportadas, adequando a infra-estrutura aos requisitos de *QoS* necessários ao funcionamento dessas aplicações.

– *CBWFQ (Class Based Weighted Fair Queueing)* – Trata-se de um algoritmo proprietário híbrido que combina a garantia explícita de banda do *CQ* com a capacidade de garantir recurso para todos os fluxos de forma dinâmica do *WFQ* (figura 5-9). O *CBWFQ* permite a

criação de até 256 classes de tráfego, cada uma com sua própria fila reservada. As filas são servidas de acordo com a banda associada à classe.

A principal diferença entre *WFQ* e *CBWFQ* é que, no primeiro, a banda destinada ao fluxo é calculada instantaneamente, enquanto no último, a banda mínima é explicitamente definida.

O mecanismo *CBWFQ* é altamente eficiente para aplicações de dados; contudo, não consegue dar prioridade máxima às aplicações de tempo real, como voz e vídeo-conferência. Isso se deve ao fato de que o mecanismo garante banda mínima, mas não atraso máximo. Por esse motivo, tornou-se necessária a criação de outro mecanismo, que garantisse prioridade total para aplicações de tempo real.

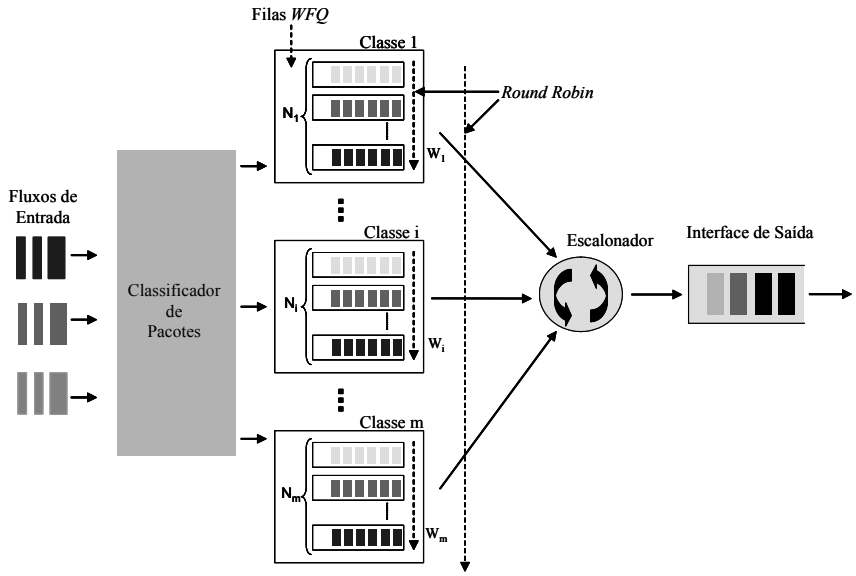


Figura 5-9: Escalonador *Class-Based WFQ*

- *LLQ* (*Low Latency Queueing*) ou *PQ-CBWFQ* – De forma simplificada, pode-se dizer que esse mecanismo proprietário combina as funcionalidades dos três algoritmos básicos: *PQ*, *CQ*

e *WFQ*. *LLQ* funciona como uma junção do *CBWFQ* com o *PQ*. O tráfego associado à fila de prioridade máxima é servido até a banda máxima reservada para essa fila, antes que todas as outras filas *CBWFQ* sejam servidas. Somente é criada uma fila *LLQ* por roteador. Desse modo, *LLQ* deve trabalhar em conjunto com outros mecanismos, preferencialmente *CBWFQ*, atendendo, desta forma, os fluxos de aplicações de dados ou não interativas.

O mecanismo foi projetado para redes que transportam voz e/ou vídeo interativo e contém dois componentes: *PQ*, para tráfego sensível de tempo real; e *CBWFQ*, para os demais fluxos. *PQ* é o mecanismo adequado para tráfego de tempo real, enquanto *CBWFQ* é adequado para aplicações de dados. Quando há tráfego presente na fila *PQ*, ela é servida à exaustão até que não existam mais pacotes ou que a banda máxima configurada para a fila seja excedida.

5.4 Policiamento e suavização (moldagem)

Policiar e suavizar o tráfego são alguns dos mais antigos mecanismos de *QoS*. Esses mecanismos têm objetivos semelhantes: identificar e reagir a violações de perfis de tráfego. Normalmente o fazem de maneira idêntica. Contudo, são diferentes em como respondem à violação (figuras 5-10 e 5-11).

O policiamento faz a verificação instantânea do tráfego e imediatamente executa uma ação pré-configurada, caso ocorra uma violação. O mecanismo é baseado na definição de um limite de tráfego que uma interface pode receber ou transmitir. Caso esse limite seja ultrapassado, por

padrão, os pacotes excedentes são descartados. O mecanismo pode ainda remarcar o tráfego excedente com uma prioridade diferente e transmiti-lo em seguida, fazendo com que a ação sobre o tráfego excedente fique a cargo do próximo roteador do domínio.

Os mecanismos de suavização, ao contrário, fazem uso de *buffers* para armazenar o tráfego excedente, ao invés de descartá-lo como ação principal. O objetivo aqui é encaminhar todo o tráfego que chega à interface, porém de forma disciplinada, para que a taxa de transmissão resultante esteja de acordo com o perfil pré-definido. Os excessos (rajadas) são armazenados em *buffers* temporários e encaminhados em partes iguais e em intervalos constantes. Com isso, é obtido um comportamento mais linear e suave do tráfego. Os descartes somente ocorrerão em situações de congestionamento nas quais o tamanho dos *buffers* não for suficiente para acomodar toda a demanda.

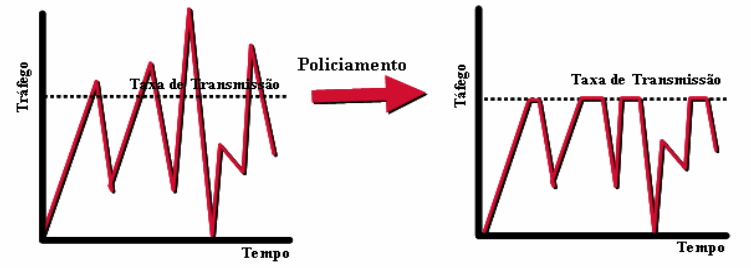


Figura 5-10: Efeito do Policiamento

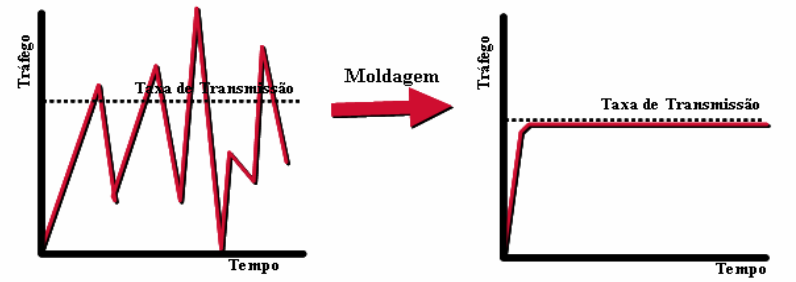


Figura 5-11: Efeitos da Suavização/Moldagem

A suavização de tráfego deve ser utilizada nos seguintes casos:

- É necessário o controle do volume do tráfego injetado em uma rede *IP*;
- Deseja-se disciplinar o tráfego destinado a determinado elemento da rede, evitando a exaustão de recursos;
- Evitar que a sobre assinação (*oversubscription*) nas redes de acesso gere demanda de tráfego não suportada pelos enlaces;
- Provocar o aumento do *RTT* (*Round Trip Time*), auxiliando na percepção do congestionamento pelas aplicações *TCP*, provocando diminuição da taxa de transmissão.

A tabela 5-3 compara a atuação dos mecanismos de policiamento e suavização de tráfego.

Tabela 5-4: Comparação entre Policiamento e Suavização

Policiamento	**Suavização**
Provoca retransmissões do *TCP* com o descarte de pacotes	Tipicamente retarda o tráfego ao invés de descartá-lo, provocando menos retransmissões do *TCP*
Inflexível e inadaptável: decide instantaneamente quanto ao descarte de pacotes	Pode se adaptar ao congestionamento da rede enfileirando o tráfego excedente
Pode ser aplicado tanto ao tráfego de entrada como de saída da interface	Normalmente aplicado ao tráfego de saída
Limitação de banda sem bufferização	Limitação de banda com bufferização

As aplicações reagem de forma diferenciada ao descarte de pacotes em uma rede *IP*. Aplicações que utilizam transporte *TCP* diminuem a taxa de transmissão, resultando no controle sobre o congestionamento. Isso não ocorre com aplicações *UDP*.

Apesar de os mecanismos de policiamento e suavização não serem empregados com o objetivo central de prover *QoS* para aplicações em tempo real, eles regulam e estabilizam o tráfego, evitando que rajadas inesperadas de tráfego de dados (*bursts*) provoquem congestionamento e, conseqüentemente, afetem os valores de atraso e *jitter* pré-definidos para as aplicações de tempo real.

Os mecanismos de suavização são comumente utilizados para garantir os acordos de nível de serviço (*SLA*) ou para compensar redes *NBMA* (*nonbroadcast multiple access*) – *Frame Relay* e *ATM*, por exemplo. Nesses casos, quando é comum a existência de enlaces de velocidades diferentes, conectando elementos concentradores (alta velocidade) a elementos de acesso (baixa velocidade), a utilização de mecanismos de suavização torna-se de extrema importância.

A geração de pacotes no concentrador tende a ocorrer a uma taxa superior à capacidade do enlace de acesso, o que provoca contenção na rede de serviços (provedor). Do contrário, quando a configuração de banda do ponto central é sobre assinada (*oversubscripted*), em momentos de picos de tráfego, a soma dos tráfegos gerados pelos pontos remotos pode ultrapassar a capacidade de banda do enlace com o ponto central, quando utilizar a suavização e o policiamento se torna de grande importância.

Ambos, policiamento e suavização, utilizam na sua implementação o mecanismo de controle denominado *Token Bucket* (balde de fichas) (ARMITAGE, 2000).

5.4.1 *Token bucket*

Policiamento e suavização são modelados com a utilização do modelo matemático *Token Bucket* (balde de fichas) (SZIGETI, 2004). Essencialmente, trata-se de um algoritmo que define um mecanismo de medição para determinar quanto de tráfego pode ser enviado de acordo com as taxas especificadas (figura 5-12).

Este esquema utiliza um conjunto de fichas (*tokens*) que são colocados em um balde (*bucket*) a uma taxa constante. Cada *token* pode representar, por exemplo, um crédito para transmissão de um *byte*. Um pacote é transmitido quando há *tokens* suficientes no *bucket* para a transmissão de todo o pacote.

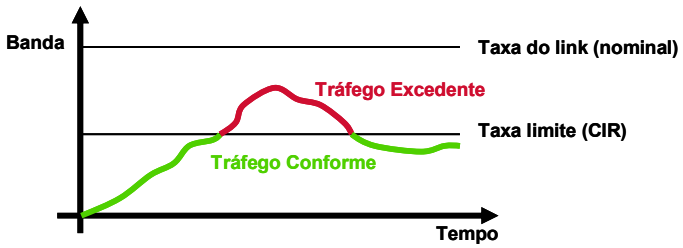


Figura 5-12: Tráfegos conforme e excedente

São definidos uma taxa de dados *CIR* (figura 5-13) e uma rajada *Bc*. A analogia é imaginar um balde com uma determinada capacidade máxima, que contém fichas que são inseridas regularmente, a cada intervalo de tempo *Tc*. Uma ficha corresponde à permissão para transmitir uma quantidade de *bits*. Quando chega um pacote, o seu tamanho é comparado com a quantidade de fichas no balde. Se existir uma quantidade suficiente de fichas, o pacote é enviado. Se não, pode ser, por exemplo, descartado (*policing*) ou inserido em uma fila para

que aguarde até haver fichas suficientes no balde (*shaping)*. Caso o tamanho da fila seja zero, todos os pacotes fora do perfil (que não encontram fichas suficientes) são descartados.

Outro problema resolvido com a utilização do mecanismo é a questão da inatividade das interfaces. São comuns casos em que a velocidade da interface é maior que a taxa de transmissão acordada (*CIR*). Com isso, a interface pode transmitir o tráfego definido no *CIR* em um intervalo menor que 1 segundo, ficando o restante do tempo inativa. Por exemplo, caso a velocidade seja o dobro do *CIR*, o valor de tráfego correspondente ao *CIR* precisará de 500 ms para ser transmitido. Dessa forma, a interface ficaria inativa durante os 500 ms restantes, aguardando o início do próximo intervalo para retomar a transmissão de pacotes. Esse tempo será, certamente, um grande problema para aplicações sensíveis a atraso e *jitter*. Assim, para evitar longos períodos de inatividade, ao invés de transmitir todo o *CIR* de uma só vez, é feita a sua divisão em partes (*Bc*), que são transmitidas em intervalos menores de tempo.

Parâmetros do *Token Bucket*:
- Taxa de carga de tokens - *v*
- Tamanho do balde - *Bc*
- Intervalo de inserção de tokens– *Tc*
- Capacidade do enlace - *C*

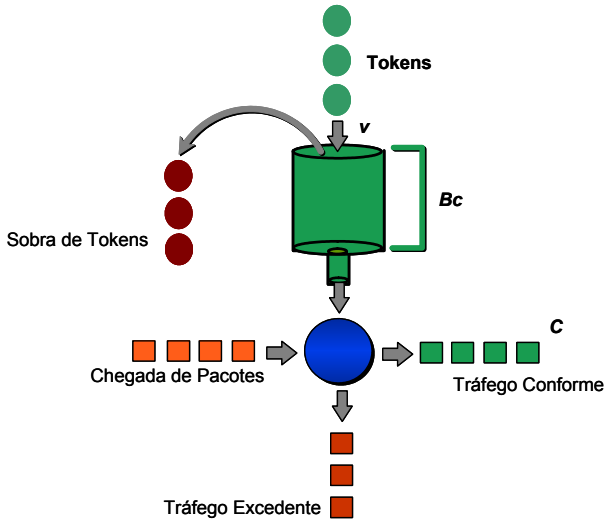


Figura 5-13: Funcionamento do *token bucket*

O mecanismo pode ser adaptado para tolerar mais rajadas temporárias (HEINANEN, 1999b), evitando o descarte e conseqüentes quedas de desempenho das aplicações *TCP* com retransmissões constantes. Essa adaptação inclui a definição de um balde secundário para a bufferização do tráfego excedente. Além disso, o mecanismo *token bucket* pode especificar ações sobre o tráfego excedente, de acordo com o tempo e o volume do excesso, prevendo a remarcação dos pacotes pertencentes ao tráfego acima do *CIR* (não conforme) (HEINANEN, 1999b).

5.5 Mecanismos para evitar congestionamento

Normalmente, uma rede compartilha aplicações *UDP* e *TCP*. Essas aplicações funcionam simultaneamente e o comportamento diferente dos protocolos afeta a forma como são utilizados os recursos da rede. O *TCP* é orientado a conexão, confiável, adaptativo e possui alto *overhead.* O *UDP* possui características opostas (COMER, 1998). Os mecanismos de *QoS* existem para disciplinar o uso da rede, oferecendo tratamento adequado tanto para aplicações *TCP*, como para *UDP*.

Esta seção abordará os mecanismos utilizados para evitar o congestionamento baseados em gerenciamento ativo das filas (*AQM – active queue management*) e que propõem detectar o congestionamento ainda incipiente. São eles:

- *RED – Random Early Detection*
- *WRED – Weighted RED*

– *ECN – Explicit Congestion Notification*

5.5.1 Controles do *TCP*

Mecanismos de controle de congestionamento complementam e se relacionam com os algoritmos de fila e são projetados para atuar no tráfego *TCP*. No estabelecimento de uma conexão *TCP*, são informados o tamanho máximo da janela e o tamanho máximo do campo de dados (*MSS – Maximum Segment Size*). O protocolo opera com duas janelas: a primeira (*window*), definida pelo receptor, e a segunda (*cwnd – congestion window*), resultante da percepção do transmissor quanto ao nível de congestionamento da rede. A janela resultante será o mínimo entre as duas: *real_window* = min (*window*, *cwnd*). Além disso, é definida a variável *ssthresh* (*slow start threshold size*), correspondente à metade do valor da *cwnd* no momento do primeiro descarte (sintoma de congestionamento). O valor de *cwnd*, inicialmente, é definido pelo *MSS*, que, por padrão, é de 536 *bytes*. Para cada confirmação recebida, o valor da *cwnd* é incrementado exponencialmente. Esse mecanismo, conhecido como *Slow Start*, permanece até que o primeiro *timeout* ocorra, significando que houve perda de pacotes. Nesse ponto é acionado o algoritmo *Congestion Avoidance*, que reduz o valor da *cwnd* à metade para cada perda de segmentos, até o mínimo de um segmento (*MSS*). Uma vez cessado o congestionamento, a partir do momento em que recebe a primeira confirmação, o *TCP* volta a incrementar a *cwnd*. A forma como o *cwnd* é incrementado depende se o *TCP* está em *Slow Start* ou em *Congestion Avoidance*. Se *cwnd* ≤ *ssthresh*, o *TCP* está em *Slow Start* e o *cwnd* é incrementado exponencialmente. Se *cwnd* >

ssthresh, o *TCP* está em *Congestion Avoiance* e *cwnd* é incrementado de forma linear (ALLMAN et al., 1999) (figura 5-14).

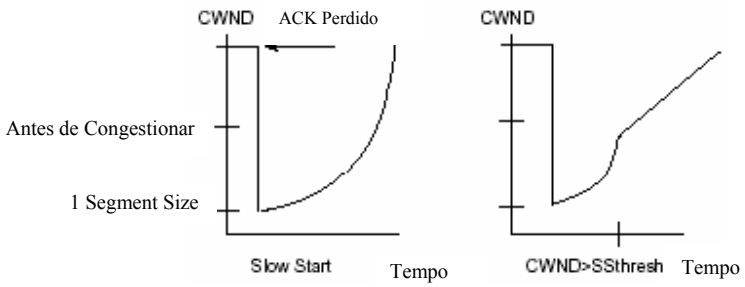


Figura 5-14: *Slow Start* e *Congestion Avoidance*

O *TCP* ainda pode gerar mais de uma confirmação para o mesmo segmento, caso receba segmentos fora de ordem. Isso pode ocorrer, sobretudo, em função de atrasos impostos por diferentes rotas usadas pelos pacotes ou mesmo pela perda de pacotes. Nesse caso, sendo o número de confirmações de um mesmo segmento maior ou igual a quatro, o *TCP* retransmite imediatamente sem esperar pelo *timeout*. Esse procedimento é o funcionamento básico do algoritmo *Fast Retransmit* (retransmissão rápida).

Quando isso ocorre, o *TCP* aciona diretamente o *Congestion Avoidance*, sem passar previamente pelo *Slow Start*. O *TCP* somente retransmite o segmento perdido, pois os demais estão no *buffer* do destino aguardando processamento. Esse procedimento é conhecido como *Fast Recovery* (recuperação rápida).

O objetivo desses mecanismos é o aumento da eficiência e estabilidade das conexões *TCP*, com menor consumo de banda com retransmissões.

5.5.2 Efeitos do *tail drop*

Quando não há nenhum mecanismo especial de descartes habilitado na interface, é utilizado o *tail drop*. Isso significa que, caso os *buffers* da interface estejam cheios, os próximos pacotes são descartados à medida que chegam. Como os pacotes são descartados em seqüência, acabam afetando a maior parte das conexões estabelecidas, provocando o disparo dos mecanismos de controle de congestionamento do *TCP*. As conexões reduzirão a *cwnd* para reduzir o congestionamento. Em seguida, as *cwnd* serão incrementadas com o *Slow Start* e *Congestion Avoidance*. Eventualmente, a *cwnd* das conexões pode chegar ao valor anterior, o congestionamento pode ocorrer novamente e todo o processo se repetir. O que se observa é que todas as conexões passam a ter um comportamento semelhante, quando experimentam descartes de pacotes próximos. Esse efeito chama-se sincronização global (*global synchronization*) (figura 5-15).

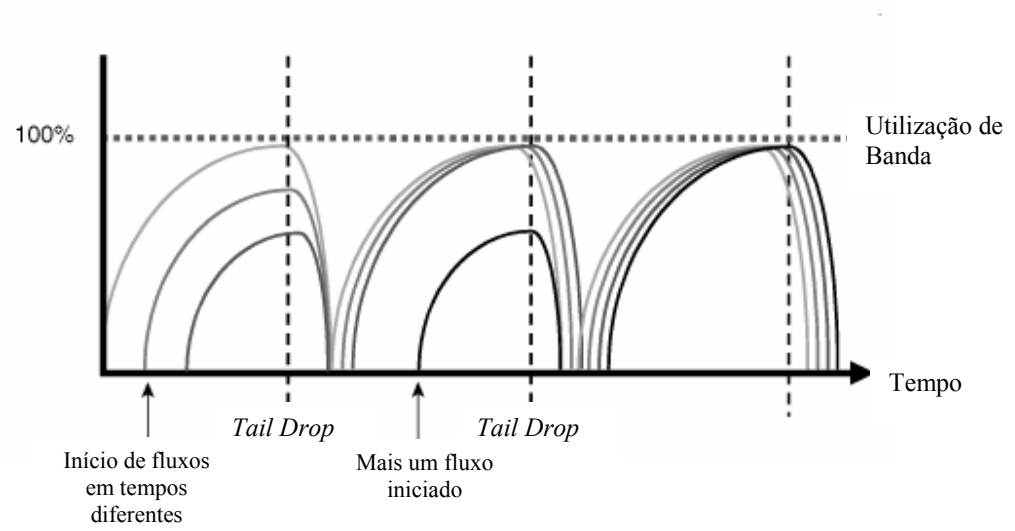


Figura 5-15: *Tail Drop* e *Global Synchronization*

A sincronização global provoca o uso ineficiente da banda e pouco contribui para evitar congestionamento, afetando somente fluxos *TCP* com pacotes descartados. Os demais fluxos sem descarte e os fluxos *UDP*, que não reagem à perda de pacotes, não serão atingidos.

Para abordar essas questões, foi criado o *RED – Random Early Detection* e suas variações, descritos a seguir.

5.5.3 *RED – Random Early Detection*

O *RED* é um algoritmo de descarte aleatório (FLOYD, 1993) utilizado em redes *DiffServ*, por exemplo, com o objetivo de minimizar o congestionamento dentro de uma classe.

A idéia central do funcionamento do algoritmo é, ao invés de esperar que uma fila *FIFO* encha para passar a descartar pacotes, descartando provavelmente pacotes de tráfego importante, ele descarta pacotes com uma certa probabilidade sempre que a fila excede um determinado nível (REZENDE, 1999). O roteador computa o tamanho médio da fila "L" (*avglen*) usando uma média ponderada por uma variável Q_{WEIGHT} e utiliza dois patamares na fila que são o "*min_thresh"* e "*max_thresh"*, onde *min_thresh* < *max_thresh*. Quando um pacote chega ao roteador e o *avglen* é menor que o limite inferior, ele coloca o pacote na fila. Quando o *avglen* é maior que o limite superior, o pacote é sempre descartado. Se *avglen* se encontra entre os dois limites, então o novo pacote é marcado para ser descartado com uma probabilidade *p*, onde *p* é função do tamanho médio da fila e do tempo decorrido desde o descarte do último pacote.

A probabilidade com que um roteador decide descartar um pacote é proporcional à parcela de banda passante que este fluxo está usando no roteador, ou seja, quanto mais pacotes um fluxo envia, maior é a probabilidade de descarte dos pacotes. Esta probabilidade tem um limite superior dada pelo parâmetro $P_{MAX.}$ Quanto maior for P_{MAX} , maior será a agressividade do algoritmo no descarte dos pacotes.

Os valores *min_thresh e max_thresh* são altamente vinculados à caracterização do tipo de tráfego ao qual a rede está sendo submetida. Por exemplo, se o tráfego for muito variável (*bursty*), o *min_thresh* deve ser suficientemente grande para manter uma boa utilização do enlace. Além disso, a diferença entre os dois limiares deve ser maior do que o incremento do tamanho médio da fila em um *RTT* (*Round Trip Time*). É considerada uma boa regra para uma rede que transporta vários tipos de tráfego utilizar *max_thresh* = 2 * *min_thresh* (REZENDE, 1999). Os valores escolhidos devem conciliar baixo atraso (requisitos de aplicações de tempo real, como as de voz) e alta utilização para garantir a eficiência de aplicações caracterizada por um tráfego intenso e persistente, como, por exemplo, uma aplicação *FTP*. Os limiares mínimo e máximo são determinados pelo tamanho da fila desejada, sendo que este deve ser o tamanho que alcança os resultados desejados, tais como custo/benefício entre maximizar a vazão e minimizar o atraso.

O mecanismo *RED* possui dois algoritmos distintos. O algoritmo para computar o tamanho médio da fila e o algoritmo para calcular a probabilidade de marcação de pacotes para descarte. Essa proposta tem o objetivo de marcar pacotes em intervalos regulares e de forma equilibrada, a fim de evitar ajustes e sincronizações globais e a fim de evitar um descontrole em relação ao tamanho médio da fila.

O algoritmo do tamanho médio determina o grau de explosividade que será permitido na fila do roteador e leva em consideração o período em que a fila está vazia, estimando o número "m" de pequenos pacotes que podem ter sido transmitidos pelo roteador durante o período ocioso. Após o período ocioso, o roteador computa o tamanho médio da fila como se "m" pacotes tivessem chegado em uma fila vazia durante aquele período.

Os roteadores de borda de um domínio *DiffServ* monitoram e marcam pacotes dos agregados de fluxos de uma classe de serviços assegurados (*AF*), levando em consideração a preferência de descartes. Quando apenas dois níveis de precedência de descarte são utilizados, os pacotes que obedecem ao perfil de serviço contratado são marcados como *IN* (*in profile*) e os pacotes que estão além do perfil de serviço são marcados como *OUT* (*out profile*). O gerenciamento das filas nos roteadores internos ao domínio de diferenciação de serviços é normalmente realizado pela adoção de dois algoritmos *RED*: um para pacotes *IN* e outro para pacotes *OUT*. Este mecanismo é conhecido como *RIO* (*RED In Out*).

O funcionamento do *RIO* é o mesmo do *RED*, havendo, entretanto, a distinção entre os pacotes *IN* e *OUT*, a partir do momento em que, na avaliação dos pacotes *IN*, o cálculo do tamanho da fila leva em consideração apenas os pacotes *IN*, enquanto que, na avaliação dos pacotes *OUT*, o cálculo do tamanho médio da fila leva em consideração todos os pacotes, isto é, *IN* + *OUT*. Conseqüentemente, o tamanho médio da fila dos pacotes *OUT* será maior e maior será a probabilidade de que essa fila chegue mais rapidamente ao limite superior, o que implicaria no descarte de um maior número de pacotes *OUT*. O *RIO* descarta todos os pacotes *OUT* em caso de congestionamento persistente. Caso não resolva, após descartar os pacotes *OUT*, ele começa a

descartar os pacotes *IN*, na tentativa de controlar o congestionamento. Fica caracterizada, dessa forma, certa prioridade para os pacotes *IN*.

O algoritmo *RED* estabelece que nenhum pacote deva ser marcado para descarte caso o tamanho médio da fila seja inferior a um limiar inferior e que todos os pacotes sejam marcados para descarte caso o tamanho médio da fila exceda um limiar superior. Se todos os pacotes marcados são de fato descartados, assegura-se que o tamanho médio da fila nunca excede significativamente o limiar superior.

5.5.4 *WRED – weighted RED*

WRED é uma melhoria do algoritmo *RED* que cria um grau de influência sobre a estratégia aleatória na seleção dos pacotes para descarte, por considerar um peso, equivalente ao *IP Precedence – IPP*, dentro do processo de seleção.

Dentro do algoritmo de *WRED*, um ponto inicial mínimo para um dado valor de *IPP* determina o tamanho da fila em que os pacotes com aquele *IPP* começam a ser aleatoriamente descartados. Um valor máximo determina a profundidade da fila, a partir da qual todos os pacotes com o valor de *IPP* associado serão descartados. Estes pontos são ajustáveis, como o denominador de probabilidade da marcação, que determina a agressividade quanto aos descartes para um valor de *IPP*. Por exemplo, um valor de denominador de probabilidade de 10 indica que até 1 em cada 10 pacotes com esse valor de precedência será descartado aleatoriamente, sendo que a taxa máxima

de 1 em 10 acontece no ponto inicial máximo, com a taxa de descartes crescendo linearmente até este máximo, como mostrado na figura 5-16.

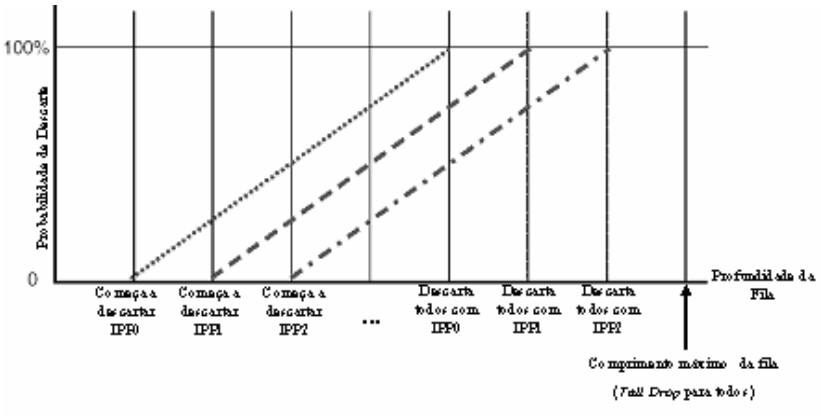


Figura 5-16: Comportamento do algoritmo *WRED*

O algoritmo *WRED* pode considerar, ao invés de valores de *IPP*, valores de *DSCP*, quando é conhecido como *WRED* baseado em *DSCP* (*DSCP-based WRED*). Nesse caso, consiste em um algoritmo que usa os valores de preferência de descarte do *PHB AF*, conforme a marcação dos pacotes, para influenciar na probabilidade de descarte à medida que as filas enchem. Um exemplo de operação simplificada é mostrado na figura 5-17 a seguir.

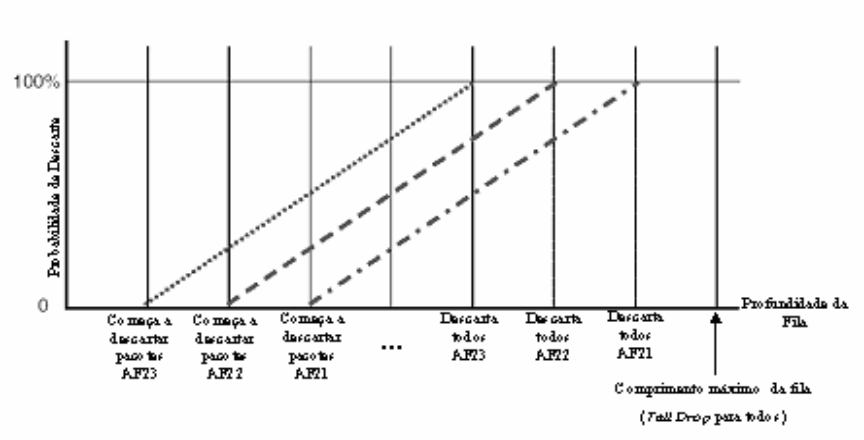


Figura 5-17: Comportamento do algoritmo *DSCP-based WRED*

5.5.5 *ECN – Explicit Congestion Notification*

Tradicionalmente, a única maneira de informar a um transmissor um estado de congestionamento, fazendo-o diminuir a taxa de transmissão, era o descarte de pacotes. Entretanto, alguns resultados de pesquisa sugerem que o *TCP* teria melhores condições de se adaptar às condições momentâneas da rede, caso o transmissor fosse explicitamente notificado da ocorrência de congestionamentos.

Dois tipos de soluções têm sido considerados: uma delas utiliza a mensagem *Source Quench* do protocolo *ICMP* para a notificação. A outra, utiliza a marcação de pacotes *IP* na rede e a identificação pelo *TCP* receptor da ocorrência de congestionamento através de *ACK*s. Esta última ficou conhecida pelo nome genérico de *ECN* (*explicit congestion notification* – notificação explícita de congestionamento).

O *ECN* propõe uma nova e mais eficiente forma de notificação de congestionamento, utilizando os dois *bits* restantes e anteriormente não usados do campo *ToS* do pacote *IP* (RAMAKRISHNAN, 2001). Com isso, os dispositivos conseguem sinalizar que passam por congestionamento. Os dois *bits* são assim definidos (figura 5-18):

- *ECT – ECN-Capable Transport* – indica se o dispositivo suporta *ECN*;
- *CE – Congestion Experienced* – em conjunto com o *bit ECT* indica se houve congestionamento no percurso do pacote.

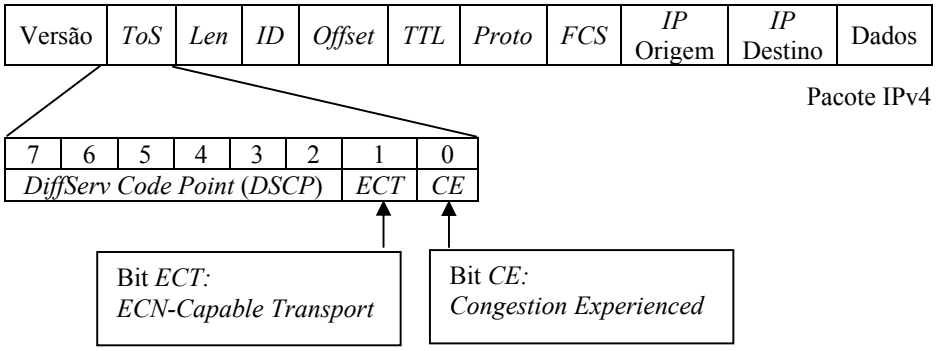


Figura 5-18: Campo *ToS* com *ECN*

Quando o *bit ECT* do campo *ToS* é igual a 0, a indicação é de que o *host* não suporta *ECN*. As demais combinações, listadas na tabela 5-5, são utilizadas para a notificação. Quando a combinação é 11, há indicação de congestionamento, denominada *CE – Congestion Experienced.* As combinações 10 e 01 são tratadas da mesma forma pelos roteadores e podem ser tratadas de forma diferenciada pelos *hosts* transmissor e receptor. A RFC-3168 (RAMAKRISHNAN, 2001) não detalha a utilização dessas combinações.

O *ECN* aprimora o resultado do *RED*, pois o roteador, ao invés de descartar pacotes quando o comprimento da fila está entre valores limite mínimo e máximo, pode marcá-los com a combinação *CE* e transmiti-los. Chegando ao destino, o receptor deverá gerar pacotes de retorno para o transmissor com os *bits ECN* marcados (*CE*). Ao receber esses pacotes, o *TCP* do transmissor deve reagir ao congestionamento com a diminuição da taxa de transmissão através do algoritmo *Congestion Avoidance.*

Durante períodos de congestionamento, *RED* (e *WRED*) descarta pacotes quando o comprimento médio da fila excede um valor específico (*threshold*). O *ECN* serve como uma extensão do *RED*,

marcando os pacotes, ao invés de descartá-los, para sinalizar a existência de congestionamento quando o comprimento médio da fila excede um valor limite.

Tabela 5-5: Bits *ECN*

Bits ECN		Significado
0	0	Sistema final não entende *ECN*
0	1	Sistema final não entende *ECN*
1	0	Não há congestionamento
1	1	Notificação de congestionamento

O principal objetivo do *ECN* é provocar a diminuição de fluxos de pacotes de aplicações *TCP* antes que ocorram descartes. O mecanismo pode ser de boa utilidade para aplicações que não toleram perdas de pacotes. Atualmente, entretanto, o mecanismo *ECN* é pouco utilizado e alguns fatores podem ser levantados quanto à dificuldade de disseminação do mecanismo. O primeiro é a maior complexidade de sua implementação, já que ambos, origem e destino da conexão, têm que estar preparados para seu uso. O segundo é a dependência deste mecanismo do uso do *RED* nos roteadores presentes no caminho da conexão entre origem e destino. Levando em consideração que alguns provedores, sobretudo os de trânsito, que alegam que perdas de pacotes em seus roteadores são próximas a zero, não vêem utilidade no uso do *RED*, a disseminação do *ECN* torna-se ainda mais comprometida. O principal argumento é que, enquanto provedores de acesso seriam potencialmente beneficiados com uma significativa redução nos descartes de pacotes, nos grandes provedores de trânsito do núcleo da *Internet* ocorreria justamente o contrário. Ou seja, eles iriam descartar pacotes que, sem o *RED*, provavelmente, não seriam descartados.

É importante observar, ainda, que *ECN* é aplicável somente a aplicações *TCP*, não prevendo qualquer notificação para aplicações *UDP* em caso de congestionamento.

5.6 CAC – controle de admissão de conexões

Como discutido anteriormente, a utilização de mecanismos de gerência e controle de congestionamento tem por objetivo principal reduzir os efeitos do congestionamento sobre o tráfego prioritário, através do controle do tráfego das aplicações. Nesses casos, não são aplicadas políticas de admissão. Se dados das aplicações experimentam congestionamento, pacotes são normalmente descartados. Protocolos como o *TCP* detectam e recuperam as perdas através de retransmissões, o que assegura a manutenção da sessão para o usuário final e a diminuição dos níveis de contenção.

Entretanto, políticas de controle de congestionamento não surtem o mesmo efeito para aplicações de tempo real, como voz e vídeo-conferência, quando se deseja manter os níveis de qualidade previstos e necessários ao bom funcionamento das chamadas estabelecidas. Quando pacotes dessas aplicações são admitidos na rede, eles não devem ser descartados. Aplicações *TDM* (*time division multiplexing* – multiplexação por divisão do tempo) tradicionais, como chamadas de voz, modem, fax e vídeo, assumem que a banda requerida está disponível e não fazem recuperação de informações perdidas. Nesses casos, como o significado da informação está relacionado ao tempo da sua chegada, não há sentido em prover detecção de erros e mecanismos de recuperação, como a retransmissão. Se um pacote não foi entregue ao destino dentro de determinado intervalo de tempo, considera-se que ele foi perdido e que não chegará mais. Isso significa dizer que a máxima “antes tarde do que nunca” não se aplica aos fluxos de tempo real.

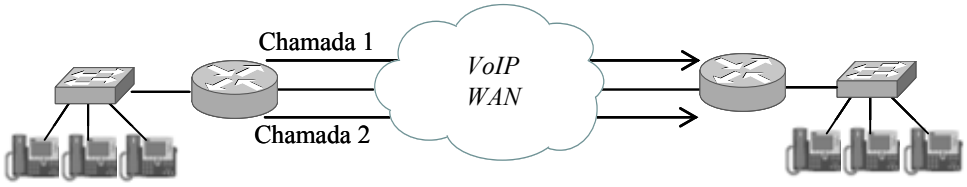
Por definição, a banda na rede é finita e pontos de congestionamento podem ocorrer. Ambos, tráfego de tempo real e tráfego gerado pelas aplicações convencionais, passarão pelos pontos de contenção. Se os pacotes não podem ser descartados para resolver o congestionamento, os fluxos de pacotes adicionais que causam esse congestionamento não devem ser admitidos na rede (SZIGETI, 2004). Por esse motivo, pode-se afirmar que CAC consiste, em essência, em mecanismos para evitar congestionamento para aplicações de tempo real. Depois de admitido, um fluxo de tempo real deve ser transportado. Se não há recursos disponíveis suficientes para esse transporte, que garantam o atendimento aos índices de *QoS* requeridos, o fluxo deve ser rejeitado ou redirecionado antes de admitido na rede.

Formalmente, CAC pode ser definido como uma decisão determinística (admissão ou rejeição) que ocorre antes do estabelecimento de uma nova chamada. Essa decisão é tomada por um algoritmo de alocação de recursos que determina se a rede possui recursos suficientes para o estabelecimento da conexão e que é função da caracterização antecipada do tráfego, dos parâmetros de *QoS* especificados, da disponibilidade de recursos e do tráfego existente. A decisão, ainda, envolve um conjunto de ações executadas pela rede durante a negociação da admissão de uma nova conexão.

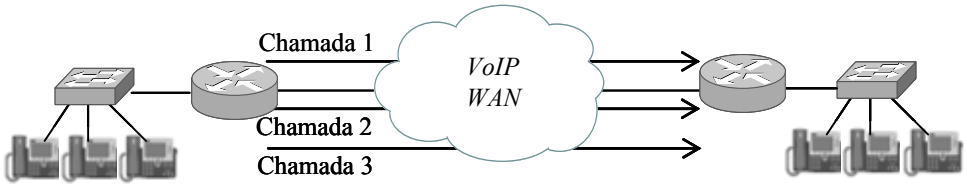
A admissão de uma nova conexão deve ocorrer sem prejuízo para as conexões já estabelecidas. Portanto, o CAC é um controle preventivo de congestionamento, que impede que uma carga excessiva de tráfego prejudique o desempenho experimentado pelos usuários da rede (LEE, 2001). O grau de utilização dos recursos da rede é conseqüência direta da política de admissão, uma vez que rejeições desnecessárias de conexões que poderiam ter sido admitidas com sucesso

sub-utilizará os recursos da rede. O oposto também é verdade: um algoritmo que incorretamente admita muitos fluxos induzirá violações de *QoS* e conseqüente degradação do serviço.

A utilização dos mecanismos de controle e gerência de congestionamento, mostrados anteriormente, tem como um dos objetivos centrais a proteção do tráfego de tempo real dos demais tráfegos gerados pelas aplicações convencionais. O uso de mecanismos de CAC, ao contrário, visa proteger o tráfego de tempo real dele mesmo (figura 5-19), ao evitar que novas conexões admitidas degradem o nível de serviço das conexões pré-estabelecidas.



Ambas as chamadas são admitidas e servidas adequadamente, pois a banda pré-alocada é suficiente para isso.



Se uma terceira chamada é admitida, as três passam a ser atendidas com degradação, pois, como a banda é insuficiente, pacotes das três serão descartados.

Figura 5-19: Protegendo o tráfego de voz do tráfego de voz

Caso uma chamada de voz, por exemplo, não seja admitida, ela pode ser redirecionada para uma gravação, alertando ao usuário que a rede está ocupada e não pode aceitar novas conexões. A chamada pode, ainda, ser redirecionada por outra rota ou pode ser completada através das redes de serviços de voz tradicionais das operadoras.

Segundo (SZIGETI, 2004), os algoritmos de CAC podem ser classificados em três categorias:

- Locais – o nó toma a decisão quanto à admissão, baseado em informações locais do próprio nó. São aplicados, por exemplo, nos *gateways* de voz e podem avaliar banda de voz disponível, disponibilidade de canais, número máximo de conexões, banda disponível em um enlace, etc.;
- baseados em medição – nesse caso, usualmente, são enviados pacotes de *probes* para o destino que retornam informações sobre a disponibilidade de recursos da rede. Tipicamente, informações retornadas quanto à taxa de perda de pacotes e atraso fundamentam a decisão do mecanismo CAC para chamadas de voz;
- baseados em recursos, divididos em dois tipos: os que calculam a necessidade ou disponibilidade de recursos e os que reservam os recursos para a conexão. Os recursos incluem banda em um enlace, *time-slot* em um tronco *TDM* para um *gateway* de voz, processamento, memória, etc.

As classificações dos algoritmos dizem respeito, fundamentalmente, às diferenças quanto ao esquema de alocação dos recursos: baseados em parâmetros e baseados em medições; e quanto ao elemento tomador de decisão: os baseados nos roteadores centrais (*router-based*) e os baseados na borda ou CAC "pelas bordas" (*hosts*, *gateways* ou roteadores de entrada da rede).

5.7 Mecanismos de eficiência de enlaces

Os mecanismos especialmente criados para otimizar a utilização dos enlaces se referem àqueles que são habilitados no modo ponto-a-ponto. Ou seja, funcionam dos dois lados de um enlace de dados. Essas ferramentas combinam funcionalidades de camadas 2 e 3 e consistem, basicamente, em dois conjuntos de funções:

- Técnicas de compressão de cabeçalhos – mecanismos que atuam na tentativa de reduzir os requisitos de banda, tanto para voz, como para dados;
- Fragmentação e intercalação de enlace – minimizam o atraso de serialização comuns em enlaces de baixa velocidade.

Os mecanismos são classificados como ferramentas de *QoS*, pois são freqüentemente utilizados em conjunto com outras técnicas para fornecimento da *QoS* fim-a-fim.

5.7.1 Compressão

As técnicas de compressão de cabeçalhos são descritas por diversos padrões. O primeiro esquema foi desenvolvido por Van Jacobson para compressão de cabeçalhos *IP/TCP* em enlaces de baixa velocidade (JACOBSON, 1990). O método foi aperfeiçoado pela solução *IPHC* (*IP header compression*) (DEGERMARK, 1999) e também passou a suportar compressão do *RTP* (CASNER et al., 1999). O desenvolvimento mais recente é o *ROHC* (*RObust Header*

Compression), com suporte a *RTP, UDP e ESP* e projetado para enlaces pouco confiáveis, com alta latência (BORMANN et al. 2001).

Uma das muitas vantagens de aplicações de *VoIP* é o atrativo financeiro que trazem com a redução de banda. Por exemplo, uma chamada convencional, por *TDM* ou *PCM*, consome banda fixa de 64Kbps, enquanto que uma chamada *VoIP* com qualidade equivalente pode consumir cerca de 12Kbps. Dois tipos de compressão são usados em pacotes de voz para redução da banda consumida:

- Compressão de carga útil (*payload compression*), através de CODECs de maior complexidade, como G.729 ou G.723. Outros tipos de compressão de carga não são úteis para a voz, já que a compressão é feita anteriormente pelo CODEC;
- Compressão de cabeçalho, utilizando funções como *Compressed RTP* (ou *cRTP*).

A compressão da voz com G.729 faz com que uma chamada *TDM* de 64Kbps seja reduzida a 8Kbps. As amostras de voz são encapsuladas no *VoIP* usando pacotes *RTP* com 20 *bytes* de carga por pacote (por padrão). A utilização do *RTP header compression* (*cRTP*) permite a compressão dos 40 *bytes* de cabeçalho *IP/UDP/RTP* dos pacotes de *VoIP* para valores entre 2 e 5 *bytes* por pacote.

A compressão de cabeçalho é utilizada para melhorar a relação entre o tamanho do cabeçalho e a carga útil, fazendo com que a banda seja consumida mais com tráfego das aplicações que com

controle. A compressão pode ser feita tanto para cabeçalho *TCP* (JACOBSON, 1990), como para cabeçalho *RTP* (SCHULZRINNE, 1996).

Maior eficiência com a compressão pode ser obtida, ainda, se utilizados os mecanismos padronizados para compressão de cabeçalhos de protocolos de enlace, tais como *PPP* e *PPP Multilink* (ENGAN, 1999), além do *Frame Relay*.

A utilização de mecanismos de compressão em enlaces de alta velocidade não é recomendada, pois o tempo necessário ao processamento da compressão pode influenciar no desempenho final. Além disso, a compressão demanda recursos de processamento dos equipamentos, o que, em casos de alto tráfego, pode gerar impactos negativos na *performance*.

5.7.2 Fragmentação e intercalação

As redes *IP* podem transportar pacotes de tamanhos diversos. O atraso fim-a-fim desses pacotes é proporcional ao tempo de serialização que, por sua vez, varia conforme o tamanho dos pacotes. Pacotes grandes em algum momento serão enfileirados na interface e escalonados para transmissão. Isso pode acarretar em maior atraso na transmissão dos pacotes pequenos, que precisarão esperar a transmissão.

Depois de escalonados os pacotes, os mecanismos de filas nada mais podem fazer para evitar ou postergar sua transmissão. As aplicações sensíveis ao atraso e ao *jitter* são as mais prejudicadas nesse contexto.

A solução encontrada foi fragmentar os pacotes grandes em pacotes menores, de forma a produzir maior uniformidade de comportamento no que diz respeito ao consumo de recursos para a transmissão. Além disso, os fragmentos dos pacotes maiores são intercalados com os pacotes pequenos, diminuindo o atraso. As aplicações de tempo real poderão ter, dessa forma, o desempenho esperado.

O tamanho do fragmento depende do atraso máximo que se pretende produzir, o que, por sua vez, depende da velocidade do enlace (tabela 5-6). Os mecanismos existentes para a fragmentação são o *LFI – Link Fragmentation and Interleaving* para o *PPP* (*Multilink*) e o *FRF.12* para o *Frame Relay* (SZIGETI, 2004) . Quanto maior a velocidade do enlace, menor o tempo de serialização e, por conseguinte, maior poderá ser o fragmento que viabilizará a manutenção do atraso máximo. Por esse motivo, a fragmentação é desnecessária para enlaces de alta velocidade.

Tabela 5-6: Tempo de serialização de pacotes

Tamanho do Pacote / Velocidade	1 *Byte*	64 *Bytes*	128 *Bytes*	256 *Bytes*	512 *Bytes*	1024 *Bytes*	1500 *Bytes*
56 kbps	143 µs	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
64 kbps	125 µs	8 ms	16 ms	32 ms	64 ms	128 ms	187 ms
128 kbps	62.5 µs	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
256 kbps	31 µs	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
512 kbps	15.5 µs	1 ms	2 ms	4 ms	8 ms	16 ms	23 ms
768 kbps	10 µs	640 µs	1.28 ms	2.56 ms	5.1 ms	10.2 ms	15 ms
1536 kbps	5 µs	320 µs	640 µs	1.28 ms	2.56 ms	5.12 ms	7.5 ms

5.8 Aplicação dos mecanismos

Após o levantamento e caracterização das aplicações, os mecanismos de *QoS* podem ser implementados nos equipamentos de rede, seguindo a abordagem hierárquica das funções da *DiffServ*, a fim de que se mantenha escalabilidade e granularidade no projeto. Assim, funções de classificação, marcação e condicionamento do tráfego devem ser implementadas nos equipamentos mais próximos possíveis da origem do tráfego (figura 5-20). Os roteadores do núcleo da rede, internos ao domínio *DS*, deverão identificar a classe do pacote pela marcação feita na borda e acionar a política correspondente para encaminhamento ao próximo nó (*PHB*) (figura 5-21).

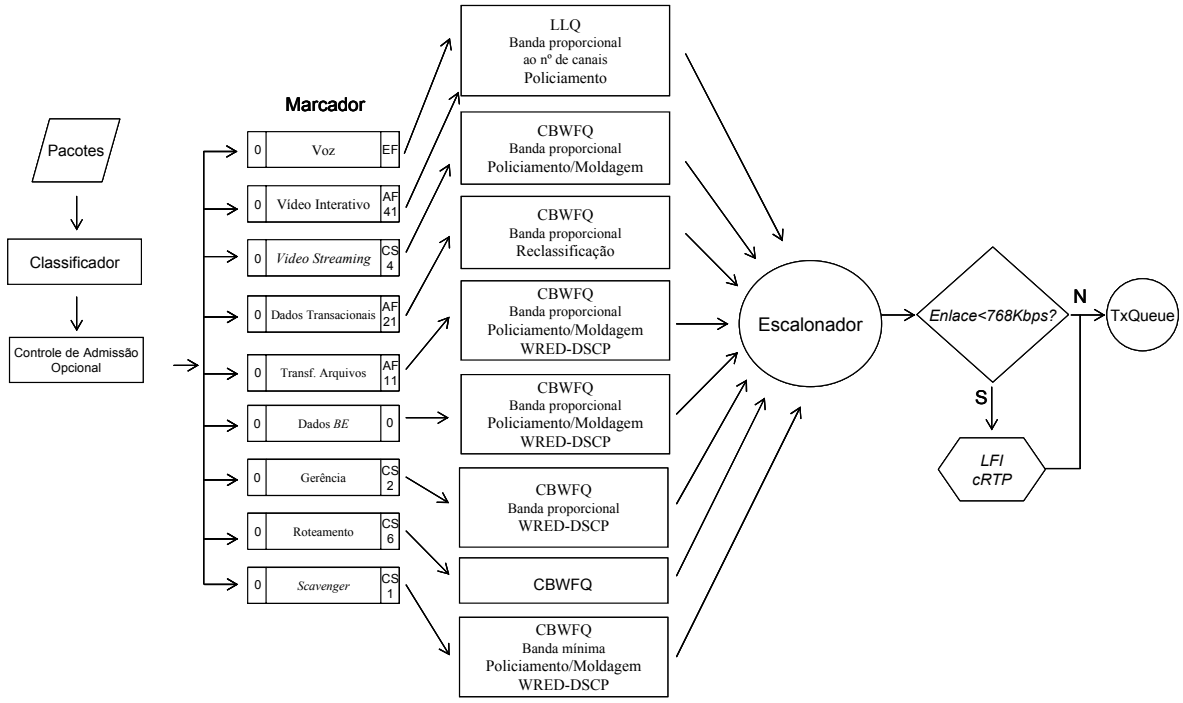


Figura 5-20: Implementação dos mecanismos na borda

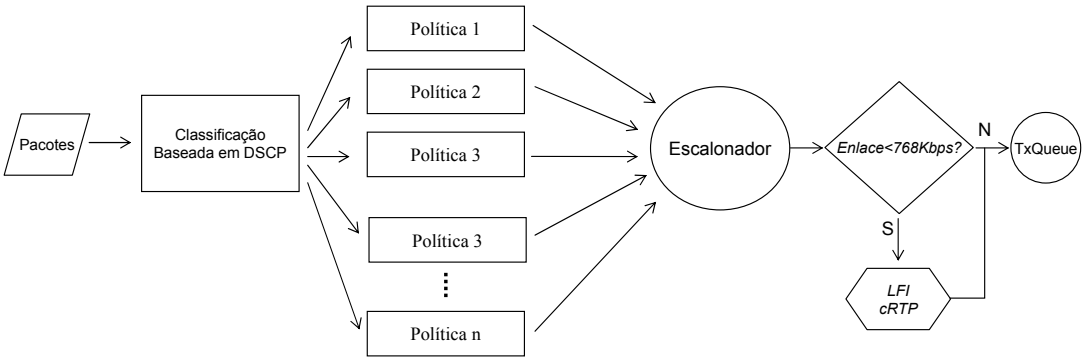


Figura 5-21: *PHB – Per-hop Behavior*

A figura 5-22, a seguir, ilustra, em blocos, as etapas do projeto de entrega da solução de *QoS* fim-a-fim.

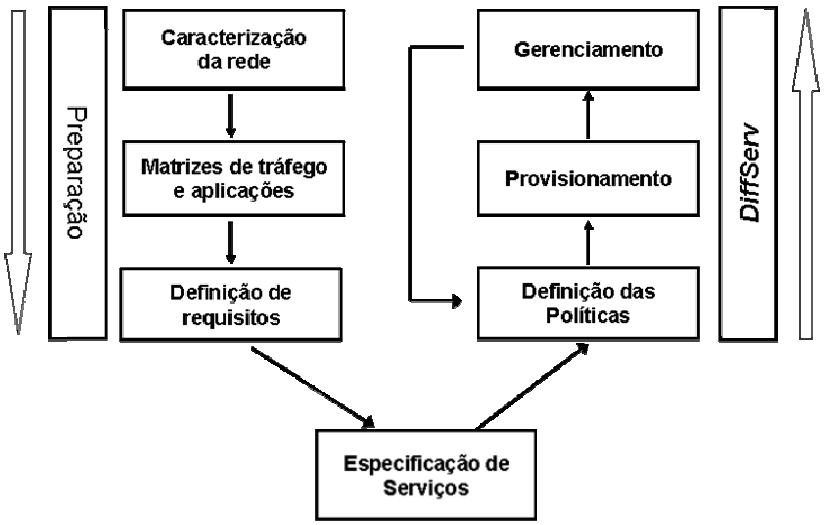


Figura 5-22: Etapas do projeto de *QoS*

A figura 5-23 procura ilustrar, em camadas, os componentes do modelo, mostrando a seqüência seguida pelos fluxos gerados pelas aplicações em uma rede baseada em políticas através da adoção da arquitetura *DiffServ* como padrão.

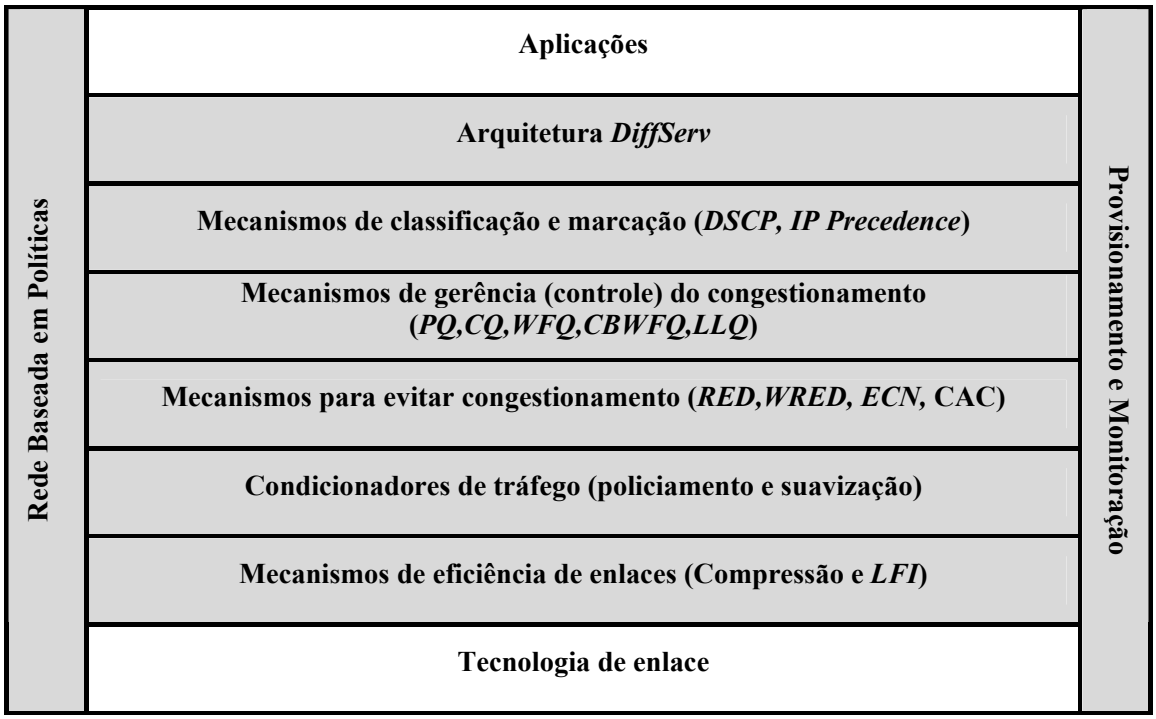


Figura 5-23: Componentes e mecanismos do modelo

6 GERENCIAMENTO DE REDES BASEADAS EM POLÍTICAS

As tarefas relacionadas ao gerenciamento de redes, por conta do grande crescimento das infra-estruturas e dos serviços suportados, têm se tornado, a cada dia, mais complexas. Se antes se exigia apenas a gestão das funções básicas do gerenciamento (falhas, desempenho, configuração, contabilização e segurança), hoje há a necessidade da gerência dos serviços disponibilizados. Para que os requisitos de funcionamento dos serviços sejam atendidos, o projeto de *QoS* é fundamental. Desta forma, passar a gerenciar a rede sob o foco das políticas de *QoS* implementadas permite a supervisão dos parâmetros de atendimento exigidos pelas aplicações e, por conseqüência, o bom funcionamento da rede.

As redes de serviço modernas, com projeto de *QoS* incorporado, são tipicamente reconhecidas como redes baseadas em políticas (*PBN – Policy Based Networks*). Tendo em vista o conjunto de problemas e requisitos para o funcionamento dessas redes a contento e a necessidade de integrar recursos que permitam a implantação, modificação e supervisão das políticas de *QoS*, permitindo uma gestão homogênea dos diferentes equipamentos, de diferentes fabricantes, veio a necessidade da criação de modelos que permitissem essa gestão integrada, a chamada gerência de redes baseadas em políticas (*PBNM – Policy Based Network Management*) (KOSIUR, 2001).

Os sistemas de gerência tradicionais estão normalmente relacionados à gestão de equipamentos em baixo nível e, até pouco tempo atrás, a gestão de políticas (regras de alto nível) não estava padronizada. O esforço de padronização da gerência por políticas parte principalmente de organizações como o *IETF* e o *DMTF* resultando propostas como *COPS* (DURHAM , 2000) e *SNMP for Configuration (*MACFADEN , 2003), *CIM* e *PCIM* (MOORE, 2001).

Os sistemas *PBNM* propõem que a configuração da rede seja feita de uma forma "automática" com base em regras de alto nível (SLOMAN , 1994). Por exemplo, o sistema de gestão deverá ser capaz de, para uma determinada situação, oferecer facilidades para reconfiguração do sistema na sua totalidade, se necessário, sem que o gestor da rede tenha que se preocupar com os detalhes de configuração dos diferentes equipamentos que constituem a rede.

Gerenciar e manter redes que se comprometem a garantir qualidade de serviço fim-a-fim para aplicações não é uma tarefa simples. Além de garantir os *SLAs*, relativos aos parâmetros de projeto, as tarefas de configuração podem ser complexas. Assim, podem demandar muito tempo até que sejam completadas, provocando períodos de inatividade.

Além da complexidade inerente aos relacionamentos entre os diferentes elementos da rede, os sistemas *PBNM* devem oferecer suporte para tolerância a falhas, pois a segurança/confiança na integridade transacional no nível do protocolo é insuficiente. Isso, em resumo, quer dizer que são necessários mecanismos que assegurem que a política definida foi aplicada na forma como foi especificada e em todos os equipamentos participantes, sendo capaz de retroceder à configuração anterior, caso não seja possível a configuração.

Estes desenvolvimentos conduziram à definição de uma arquitetura para um *framework* de políticas, composto por quatro entidades funcionais (figura 6-1) (KOSIUR, 2001): a *Policy Management Tool – Policy Console*, o *Policy Repository*, o *Policy Decision Point (PDP)* e os *Policy Enforcement Points (PEPs)*. Este modelo descreve as componentes chave, mas não faz

referência a detalhes de implementação como, por exemplo, distribuição, plataforma ou linguagem.

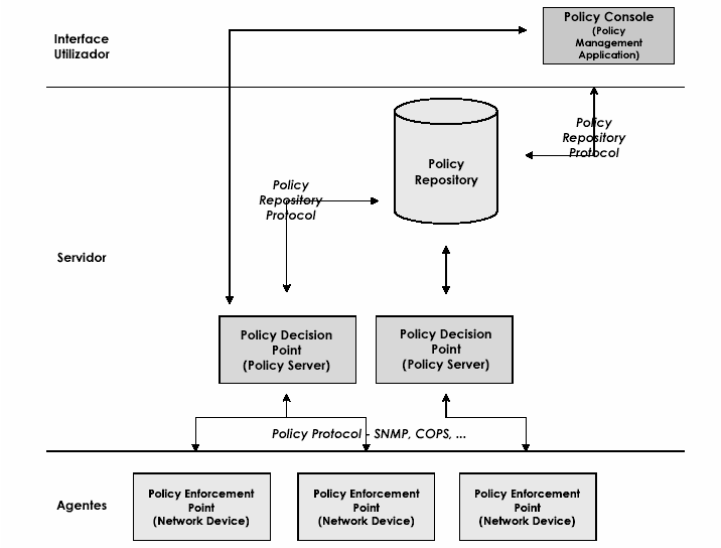


Figura 6-1: Gestão de redes baseada em políticas: modelo conceitual.

Fonte: Kosiur, 2001

O *PDP* é a entidade responsável pela verificação de quando e como as políticas podem ser aplicadas. Valida as decisões com base em medições de tráfego, análise da contabilidade, perfis de utilizadores, detecção de eventos e trata também da determinação e validação de aplicabilidade de regras de acordo com recursos específicos e funções de adaptação dos dispositivos.

O termo "política" neste contexto tem o seguinte significado: uma ou mais regras que descrevem as ações que devem ser realizadas quando determinadas condições se verificam. Semanticamente pode ser expressa como (WESTERINEN , 2001):

if (policyCondition) then (policyAction)

O *PEP*, por sua vez, é o elemento onde as decisões são aplicadas quando as condições devolvem o valor lógico "verdade". É o responsável pela execução das ações, podendo realizar operações adicionais como a verificação e a validação de condições. Como exemplos de *PEP*'s podem ser citados os roteadores, *switches*, *firewalls*, *proxies*, genericamente qualquer entidade passível de ser gerida.

O repositório de políticas (*Policy Repository*) é o local onde toda a informação relacionada com políticas é armazenada. A informação aqui guardada descreve, entre outras, usuários autorizados, aplicações, computadores e serviços, bem como os seus relacionamentos.

Um protocolo apropriado (*COPS*, *SNMP* etc.) é utilizado para a transferência de informação de políticas entre *PDPs* e entre *PDPs* e *PEPs*. O *COPS* (*Common Open Policy Service*) (DURHAM, 2000) foi desenvolvido especificamente para essa finalidade e o *SNMP* (*Simple Network Management Protocol*) é o protocolo mais utilizado na gerência de redes.

6.1 *COPS*

O *COPS* é um protocolo do tipo pergunta/resposta que suporta dois modelos para controle de políticas (figura 6-2): o modelo *Outsourcing* e o modelo *Configuration*, também conhecido como modelo *Provisioning* (*COPS-PR*).

No modelo *Outsourcing*, os eventos ocorridos no *PEP* requerem decisões imediatas que deverão ser tomadas pelo *PDP* – o *PEP* delega-lhe a responsabilidade decisória. No modelo *Provisioning* (CHAN, 2001), este relacionamento de um para um, entre eventos no *PEP*, eventos e decisões no *PDP*, não se verifica. A transferência de informação, nesse caso, pode ser feita em bloco. Por exemplo, pode ser descarregada a totalidade da informação de configuração de *QoS* de um roteador, ou por partes, como a atualização gradativa, ao longo do tempo, de um filtro para marcação *DiffServ*.

Os recursos de rede são freqüentemente configurados de acordo com parâmetros cuja variação é relativamente pequena. Enquanto o modelo *Outsourcing* está intimamente ligado à dinâmica (tempo real) do *PEP*, o modelo *Provisioning* é essencialmente dependente do *PDP*, sem grandes preocupações no que diz respeito a respostas imediatas. Os dois modelos apresentam desta forma conceitos de funcionamento diferentes um do outro.

A tolerância a falhas no *COPS* é assegurada através da verificação contante da comunicação *PEP – PDP*, por intermédio do envio de mensagens *keepalive* (DURHAM, 2000). Quando é detectada uma falha de conexão, devido à ocorrência de *timeout*, o *PEP* deve tentar novamente conectar ao *PDP* ou, no caso de impossibilidade, tentar a conexão com um *PDP* alternativo, de acordo com os seus dados de configuração. Note-se que, no *COPS*, a responsabilidade de iniciar a conexão persistente é do *PEP*.

Se o *PEP* estiver conectado a um *PDP* alternativo e o *PDP* principal retornar à operação normal, será de responsabilidade do *PDP* alternativo o redirecionamento do *PEP* para o *PDP* principal.

Não sendo possível a conexão a qualquer *PDP*, o *PEP* fica responsável pelas decisões. Entretanto, logo que a conexão seja reestabelecida, o *PEP* deverá informar ao *PDP* todas as alterações efetuadas na ausência de comunicação para que o *PDP* solicite a ressincronização de estados/comportamentos entre *PEP*'s.

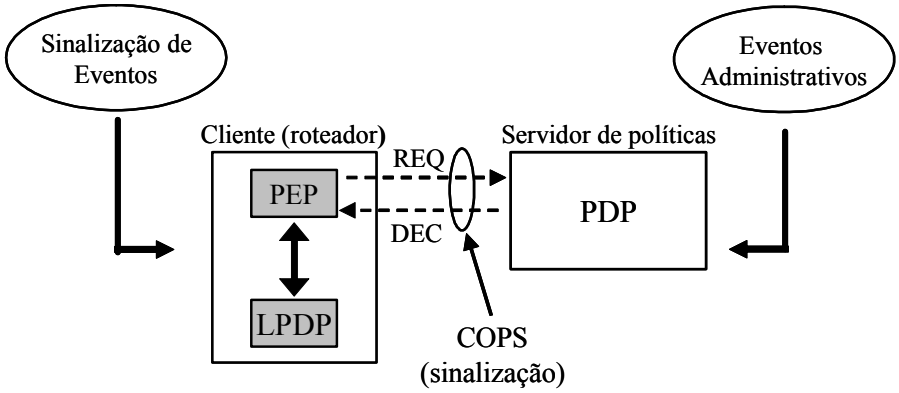


Figura 6-2: *COPS* provê conexão entre *PEPs* e *PDPs*.

Fonte: ARMITAGE, 2000 (adaptado)

6.2 *SNMP*

O *SNMP* consiste em um protocolo que permite verificar e modificar a informação de gerência de um elemento de rede remoto (agente), bem como transportar notificações geradas por esses elementos (*traps*) (STALLINGS, 1999). Para que o *SNMP* seja utilizado como protocolo de transporte de políticas entre *PDP* e *PEP*'s, é necessário que se garanta o controle transacional no nível das políticas, mapeando cada política em comandos do próprio protocolo e ações no agente.

Nesse ponto, o tamanho de mensagem do protocolo *SNMP* torna-se um inconveniente, pois são permitidas apenas pequenas atualizações, podendo a instalação completa da política demandar

um tempo demasiado, provocando, inclusive, inconsistência da própria política. Por exemplo, as primeiras atualizações podem já não estar de acordo com as posteriores. Outra possível conseqüência é que o tamanho da mensagem pode não comportar uma linha completa de uma tabela de uma *MIB*. Com isso, o preenchimento de uma linha na totalidade, em uma tabela, pode necessitar de mais de uma mensagem, o que pode acarretar problemas de consistência na própria linha, por defasamento temporal da informação.

Com o objetivo de normatizar a distribuição de políticas com *SNMP*, o grupo de trabalho *Configuration Management with SNMP (snmpconf)* do *IETF* definiu uma *Management Information Base* (MIB) para políticas, a *Policy Based Management MIB* (WALDBUSSER, 2003), que contém um conjunto de objetos relacionados com políticas, podendo ser gerenciados.

6.3 Políticas e regras

A política constitui em uma ou mais regras que descrevem as ações que devem ser tomadas quando determinadas condições se verificam. As políticas podem ser simples ou podem resultar da composição de duas ou mais regras ou mesmo da composição de várias políticas (política de políticas). As regras são os elementos mais simples (atômicos) que constituem as políticas (OLIVEIRA, 2003). Regras simples são constituídas por duas expressões lógicas binárias. A primeira define o domínio de aplicabilidade da regra e a segunda o domínio de aceitabilidade da regra. Regras compostas são regras resultantes da composição de regras simples ou regras compostas.

As operações que podem ser utilizadas na composição de regras são a conjunção *(and)*, disjunção *(or)* e negação *(not)*. A definição de políticas simples e compostas é idêntica à definição de regras simples e compostas. Assim, uma política composta é o resultado da composição de políticas simples ou compostas (figura 6-3).

If (direction is out)
If (protocol is UDP) THEN (guarantee 30% of available bandwidth)
If (protocol is TCP) THEN (guarantee 40% os available bandwidth)

} *Política Composta*

Quer em regras compostas, quer em políticas compostas, as regras ou políticas não devem entrar em conflito umas com as outras. Não deve ser possível, por exemplo, que uma política autorize o acesso a um recurso e outra, na mesma política geral, negue o acesso a esse mesmo recurso.

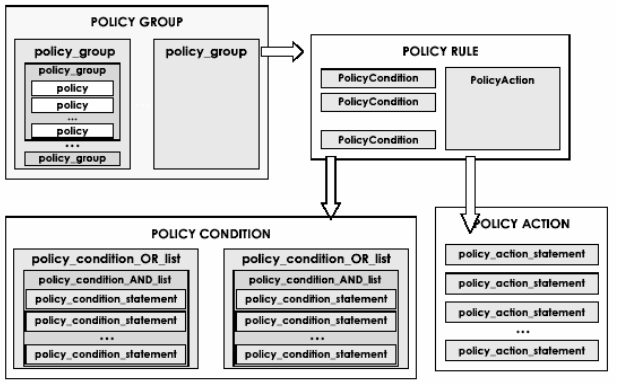


Figura 6-3: Relacionamento entre políticas, condições, ações e grupos.

6.4 Políticas e transações

A gestão de redes por políticas é uma metodologia em que a informação de configuração é obtida a partir de regras e objetivos de funcionamento da rede, sendo distribuída por vários elementos com o propósito de, no final, se conseguir um comportamento consistente da rede.

A configuração e aplicação das políticas provocam alterações de estado nos elementos da rede e é crítico que o sistema manipule as alterações de configuração de forma atômica para que na troca de estado se passe de um estado consistente para outro estado consistente. O objetivo é que a alteração de configuração nos diferentes elementos de rede em um domínio administrativo se faça como um todo, ou, na impossibilidade de tal acontecer, seja reposta a configuração anterior, mesmo que em alguns elementos a instalação da nova configuração tenha tido sucesso (figura 6-4).

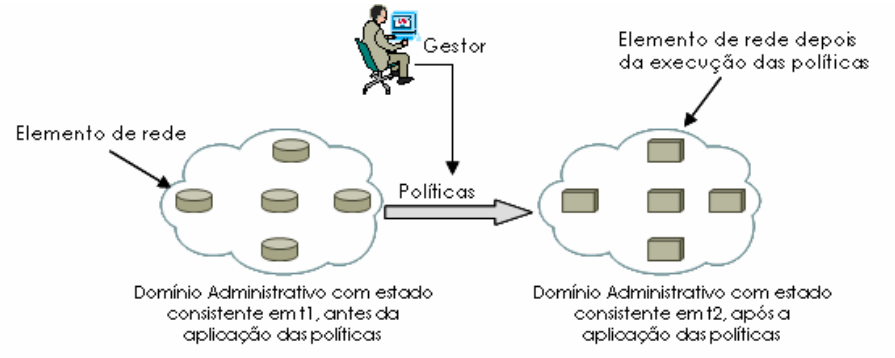


Figura 6-4: Passagem de um estado consistente a outro estado consistente em um domínio.

Considerando a associação política – transação, isto é, uma política como uma transação, em que todas as operações (regras – condição/ação) têm que ser executadas ou nenhuma operação é executada, no caso de haver uma operação que falhe, toda a instrução de configuração deve ser abortada. Por esse motivo, os sistemas de gestão de redes por políticas devem implementar as propriedades ACID (suporte às propriedades Atomicidade, Consistência, Isolamento e Durabilidade). Assim, as aplicações devem ter a capacidade de monitorizar as operações realizadas nos elementos de rede, o início e o fim da aplicação da regra/transação. Se uma operação a ser executada em um objeto colocar em risco alguma dessas propriedades, o sistema deve ter a capacidade de retornar ao estado anterior (OLIVEIRA et al., 2003).

O gerenciamento por políticas é uma metodologia em que a informação de configuração é especificada segundo regras e objetivos que, depois de distribuídos pelos diferentes elementos da rede, irão assegurar um comportamento consistente em um domínio administrativo, acordante à política de *QoS* e assegurando a *SLA* contratada.

6.5 Medição de *QoS*

São várias as formas de implementar *QoS* em uma rede *IP*. Existe, entretanto, uma questão crucial que vem após a implantação do projeto que trata de apurar a efetividade da solução implantada para o ambiente analisado. Medir *QoS* pode ter diversas finalidades, dentre elas:

- avaliar a diferença nos níveis de serviços entre transações com *QoS* habilitado e transações sem *QoS* habilitado;
- determinar se o desempenho da transação se mantém constante independente da carga existente na rede. Ou seja, verificar se há, e de quanto é, a variação no desempenho de uma transação, derivando uma métrica de constância no ambiente de *QoS*;
- verificar a capacidade da rede em consumir os recursos até o seu nível máximo de reserva, além de determinar a eficácia do esquema de reserva no caso de ambientes de *QoS* baseados na reserva de recursos.

As medições, especificamente centradas nos parâmetros de *QoS* da rede, têm, basicamente, dois objetivos fundamentais: para provedores, determinar quão bem pode-se entregar serviços que correspondem às necessidades particulares de um cliente ou aplicação. De fato, isto é a medição

do nível de consistência entre o ambiente que oferece *QoS* e as necessidades específicas do cliente; para o cliente, medir o sucesso ou falha do provedor em satisfazer os parâmetros de *QoS* acordados. Portanto, nesse caso, o cliente está motivado a executar medições que correspondem às métricas de um ambiente com *QoS* em particular.

Um usuário de uma rede remota pode também estar interessado em medir a transitoriedade de um ambiente de *QoS*, de modo a verificar se os níveis de serviço acordados para uma transação são mantidos quando pacotes pertencentes a essa transação atravessam a rede de diversos provedores e se os mecanismos de *QoS* utilizados em cada rede são compatíveis.

O administrador da rede também está interessado em medir parâmetros do ambiente com *QoS*, mas por razões diferentes do usuário. O administrador está interessado em medir:

- o nível de recursos alocados para o ambiente com *QoS* e seu impacto sobre a alocação de recursos no ambiente sem *QoS*; e
- o desempenho que o ambiente com *QoS* está entregando ao usuário, para assegurar que os níveis de serviços estão dentro dos parâmetros acordados.

Os usuários estão limitados a medir os efeitos da arquitetura de *QoS* através da medição de dados de desempenho do sistema final que gerou a transação na rede e não podem ver diretamente o comportamento dos dispositivos internos à rede. A medição no sistema final pode ser efetuada medindo o comportamento de uma transação em particular ou instalando *probes* ou sondas na rede e medindo o tempo de resposta nestas *probes*. Para medição remota da rede, o agente, um

administrador remoto ou usuário remoto, geralmente é limitado a estas técnicas de medição no sistema final: ou medição de transações ou medição através de *probes*.

Na prática, não existe um único meio de medir *QoS* e tampouco um único resultado para uma métrica de *QoS*. Por isto, o conhecimento da rede que está sendo medida e seu perfil de comportamento em condições normais tornam-se fundamentais na análise dos resultados.

A questão inicial na medição de *QoS* é determinar que agente será usado para esse fim e o que efetivamente ele irá medir. Um agente pode ser definido como uma aplicação com finalidade de monitorar ou medir componentes e parâmetros da rede. O agente utilizado deve ter privilégios de acesso aos elementos da rede suficientes para a obtenção das medidas.

A medição de desempenho de um ambiente de rede com *QoS* pode ser considerada como um caso especial de medição de desempenho das redes *IP*, e a utilização da experiência e estudos existentes nesta área são apropriados.

Ferramentas de medição devem ser baseadas no nível de entendimento da arquitetura que está sendo medida para serem consideradas como ferramentas efetivas. O grupo de trabalho *IPPM* (*IP Performance Metric*) (MAHDAVI,1999) do *IETF* procura estabelecer padrões de medição de desempenho em redes *IP*, tanto sob a perspectiva do usuário, quanto do administrador de rede.

De forma simplificada, a rede pode ser analisada como um conjunto de roteadores interligados. Em cada roteador, quando um pacote é recebido, seu cabeçalho é examinado e encaminhado para

transmissão. Se o roteador puder encaminhar o pacote, este é mapeado imediatamente para transmissão na interface de saída apropriada. Se a interface já estiver transmitindo um outro pacote, ele é enfileirado para posterior transmissão. Se o tamanho da fila exceder o tamanho máximo, o pacote pode ser descartado imediatamente, ou pode ser descartado após um tempo de espera para transmissão na fila. Enfileiramento contínuo e descarte de pacotes no roteador ocorrem quando a taxa de chegada de pacotes excede a capacidade do meio de transmissão, incluindo a capacidade da interface.

6.5.1 O efeito do tamanho da fila

A capacidade de transmissão e o tamanho da fila estão relacionados. Se a largura de banda de transmissão é de 1 Mbps e a taxa de chegada de pacotes é de 1,5 Mbps, por exemplo, a fila irá crescer a uma taxa de 0,5 Mbps. Se o tamanho da fila é de Q *bytes*, a velocidade de transmissão é de T bps, e a taxa de chegada é de A bps, o descarte de pacotes ocorre após (8*Q) / (A-T) segundos. Neste caso, cada pacote é retardado por (8*Q)/T segundos. Aumentando o comprimento de fila em uma fração da largura de banda de transmissão diminui-se a probabilidade de perda de pacotes por conta de eventuais aumentos do tráfego. Isto também aumenta a variação do tempo de transmissão dos pacotes e reduz a sensibilidade dos cronômetros de retransmissão de pacotes do *TCP*. Este incremento na variância do *RTT* (*Round Trip Time*) faz com que o *TCP* responda mais lentamente à disponibilidade dinâmica de recursos da rede. Desta forma, embora filas nos roteadores sejam um bom mecanismo, um espaço muito grande para filas tem um impacto normalmente indesejável no tempo de resposta fim-a-fim.

6.5.2 O atraso

Em redes *IP*, considerando que cada interface de um nó é um sistema de enfileiramento, o atraso está diretamente relacionado com o tamanho e a taxa de serviço da fila (ou largura de banda) de cada interface, visto que o atraso devido ao tempo de propagação do canal é constante. Uma fila grande pode evitar o descarte de pacotes, mas pode provocar mais demora em seu encaminhamento. Por outro lado, uma fila com alta taxa de serviço exige *hardware* de maior desempenho (processador, *buffer*, memória e canais de comunicação).

Outra forma de minimizar o atraso sobre determinados fluxos de tráfego, além de aumentar a taxa de serviço, é priorizá-los sobre os demais, que é a proposta da Arquitetura de Serviços Diferenciados, apresentada nos capítulos 3, 4 e 5. Desta forma, certos fluxos sofreriam menor atraso pela rede, pois seus pacotes teriam maior prioridade de envio nas filas sobre os demais, em uma tentativa de manter uma largura de banda assegurada para estes fluxos.

6.5.3 Técnicas de medição

Uma das razões para se medir o tráfego com *QoS* é poder cobrar ou contabilizar de maneira diferenciada o tráfego que recebe tratamento diferenciado na rede em comparação com o tráfego de melhor esforço. Uma outra razão importante é o fato de que uma organização que provê o serviço deve estar ciente da capacidade de seu *backbone*. Nesse caso, o intuito é verificar a diferença entre o tráfego designado para *QoS* e o de melhor esforço. São definidos dois métodos

básicos para medir o tráfego com *QoS* em uma rede: intrusivo ou ativo e não intrusivo ou passivo (FERGUSON,1999).

6.5.3.1 Medição não intrusiva

O método de medição não intrusiva mede o comportamento da rede através da observação da taxa de chegada de pacotes em um sistema final (FERGUSON ,1999). Deduções sobre o estado da rede e, portanto, sobre a efetividade de *QoS* são feitas com base nestas observações. Em geral, esta prática requer conhecimento detalhado da aplicação que está gerando o fluxo de dados que está sendo observado. Assim, a ferramenta para medição pode distinguir entre o comportamento da aplicação remota e a moderação deste comportamento imposta pelo estado da rede.

Simplesmente monitorar um dos lados de uma troca de dados arbitrária não acrescenta muita informação de valor e pode resultar em diversas interpretações dos resultados. Por esta razão, monitoração em um único lado como base para medição de desempenho da rede e, por inferência, desempenho de *QoS* não é recomendado como uma abordagem efetiva para o problema. Entretanto, existem ferramentas de monitoração baseadas em *hosts* que analisam o comportamento do fluxo *TCP* e a temporização dos pacotes dentro do fluxo.

Uma interpretação cuidadosa, comparando pacotes enviados com pacotes recebidos, pode oferecer alguma indicação sobre a extensão da distorção, se houver enfileiramento no caminho da rede para o sistema remoto. Esta interpretação também pode prover uma indicação aproximada da capacidade de dados disponível no caminho. Uma ferramenta que pode ser utilizada para

medição através do comportamento dos fluxos *TCP* é o *DBS* (*Distributed Benchmark System*). De qualquer forma, esta técnica apresenta dificuldades para a avaliação dos resultados.

6.5.3.2 Medição intrusiva

Medição intrusiva refere-se à injeção controlada de pacotes na rede e a subseqüente coleta destes pacotes. A troca de pacotes *PING* (*ICMP echo request* e *ICMP echo reply*) é um bom exemplo deste método de medição. Enviando seqüências de pacotes através deste comando em intervalos regulares, a estação de medição pode medir parâmetros tais como: alcançabilidade, tempo de resposta *RTT* (*Round Trip Time)* de transmissão para uma estação remota e expectativa de perda de pacotes no caminho de ida e volta.

A partir da execução de um determinado número de medições, considerando o comportamento da fila dentro dos equipamentos (roteadores, *switches* etc.) e combinando estas medidas com a medida de perda de pacotes e o *jitter* imposto, é possível fazer algumas inferências sobre a largura de banda disponível entre dois pontos e o nível de congestionamento. Tais medições não são, entretanto, medidas da eficácia do ambiente com *QoS*. Para medir a efetividade de uma estrutura de *QoS*, deve-se utilizar uma transação de dados típica da carga da rede e medir seu desempenho em condições controladas.

O método requer que seja medida a eficácia da taxa contínua, a taxa de retransmissão, a estabilidade do *RTT* estimado e o tempo global da transação (FERGUSON, 1999). Com a

introdução da medição de *QoS* na rede, a comparação do nível de degradação destas métricas de uma rede sem carga para uma rede plenamente carregada deve representar uma métrica de efetividade de *QoS* da rede.

O problema, em relação a esse aspecto, é que os mecanismos de *QoS* são visíveis somente quando partes da rede estão com contenção de recursos e a introdução de mecanismos de medição intrusiva no sistema acarretará aumento na condição de sobrecarga. Na prática, tentar observar o estado dinâmico de uma rede perturba este estado, e há um limite na precisão das medidas (FERGUSON, 1999). Quanto mais detalhada a informação a ser obtida, maior será o tráfego inserido e, portanto, maior o erro na medição.

Podem ser usadas, ainda, aplicações proprietárias geradoras e/ou medidoras parâmetros, capazes de injetar tráfego de forma controlada e fornecer relatórios das variáveis. Uma das ferramentas usadas nas medições feitas sobre o cenário apresentado como Estudo de Caso segue esse propósito e trata-se da aplicação Cisco *SAA* (*Service Assurance Agent*).
Neste trabalho, em condições de carga controlada na rede, foi utilizado um método intrusivo para medir a vazão, o atraso fim-afim, a variação do atraso e a taxa de perda de pacotes.

6.5.4 Métricas

As métricas para avaliação de *DiffServ* foram divididas em três grupos:

- as que descrevem o comportamento do *DiffServ* de um modo geral;
- as relativas à marcação *AF*; e

- as relativas ao comportamento da classe *EF*.

6.5.4.1 Métricas relativas à *DiffServ*

São as métricas que descrevem o comportamento das classes de serviços *EF*, *AF* e *BE* em cada interface de rede. Através deste conjunto de métricas, é possível a obtenção de informações sobre o desempenho de cada classe e desta em relação às outras. Em resumo, as seguintes métricas podem ser citadas:

- número de pacotes descartados por classe de serviço;
- número de buffers alocados por classe de serviço;
- número de pacotes enviados por classe de serviço; e
- número de pacotes que excederam a quantidade de buffers ou perfil de tráfego definido para uma classe de serviço.

6.5.4.2 Métricas relativas à marcação *AF*

Devido à existência de um marcador de pacotes para os fluxo na classe *AF*, a *DiffServ* apresenta adicionalmente métricas específicas para as filas *AF* não aplicáveis às outras duas classes: número de pacotes marcados com verde, número de pacotes marcados com amarelo, número de pacotes marcados com vermelho e número de pacotes alterados de verde para amarelo.

6.5.4.3 Métricas relativas à classe *EF*

Foram selecionadas três métricas para caracterizar o comportamento da classe de tráfego *EF* e verificar a adequação desta classe às aplicações que possuem demandas estritas de *QoS*: atraso do pacote fim-a-fim em um sentido, variação do atraso ou *IPDV-Jitter* (*Instantaneous Packet Delay Variation*) e taxa de perda de pacotes.

Para garantir medições consistentes e reprodutíveis, adotaram-se para estas métricas as definições do grupo de trabalho *IPPM* do *IETF*.

Para o *IPPM* a noção de *wire time* é fundamental. Esta noção assume que o dispositivo de medição tem um posto de observação no canal *IP*. O pacote chega em um *wire time* particular quando o primeiro *bit* aparece no ponto de observação e o pacote parte do canal em um *wire time* particular quando o último *bit* do pacote passar pelo ponto de observação. Também é definido o *host time*. Quando a medição é feita por software, este registra o *timestamp* imediatamente antes de enviar o pacote de teste e o receptor registra o *timestamp* imediatamente após a recepção do pacote. A diferença entre os dois tempos é o atraso fim-a-fim em um sentido.

6.5.4.4 Atraso fim-a-fim

O atraso fim-a-fim (*one-way delay)* é definido formalmente na RFC-2679. Esta métrica é medida do *wire time ou host-time* do pacote chegando no canal de comunicação observado pelo emissor

para o *wire time ou host time* do último *bit* do pacote observado pelo receptor. A diferença entre estes dois valores é o atraso fim-afim.

6.5.4.5 Variação do atraso ou *jitter*

A variação do atraso ou *IPDV-Jitter (Instantaneous Packet Delay Variation*) ou simplesmente *jitter* é formalmente definida pelo grupo de trabalho do *IPPM*. Ele é baseado na medição do *atraso* fim-a-fim e é definido para pares consecutivos de pacotes. A medição de um único *IPDV* requer dois pacotes. Supondo que D_i seja o atraso do i-ésimo pacote, então o *IPDV* do par de pacotes é definido com D_i-$D_{(i-1)}$. O valor do *IPDV-Jitter* é computado de acordo com a fórmula:

IPDV-Jitter = | *IPDV* | = módulo do *IPDV*

Neste trabalho, os termos variação do atraso, *jitter* e *IPDV-Jitter* são utilizados com o mesmo significado.

6.5.4.6 Taxa de perda de pacotes

A taxa de perda de pacotes em um sentido (o*ne-way packet loss*) é calculada no lado do receptor como a razão entre a quantidade de pacotes perdidos e a quantidade de pacotes transmitidos, em cada intervalo de tempo considerado. A perda de pacotes se dá em função do descarte que ocorrem nas filas dos roteadores que, quando cheias, não permitem mais a admissão e o armazenamento de pacotes para posterior envio. Protocolos como o *TCP* se recuperam desta

situação através da detecção e reenvio dos pacotes porventura perdidos. Já o protocolo *UDP*, normalmente utilizado para transmissão de áudio e vídeo, não utiliza mecanismos de retransmissão, sendo as perdas, portanto, irrecuperáveis. É útil frisar que, para os fluxos de tempo real, a retransmissão não teria utilidade.

A taxa percentual de perdas de pacotes é calculada da seguinte forma:

Taxa de perdas=(Número de pacotes perdidos / Número total de pacotes recebidos) * 100

7 PROJETO APLICADO

Ao longo desse trabalho surgiram algumas oportunidades de aplicação do modelo apresentado em cenários de redes corporativas. Um deles foi selecionado de forma a exemplificar e ilustrar a implementação do modelo em uma rede real.

7.1 Descrição do cenário

A empresa pesquisada atua no segmento do comércio de bens duráveis, sobretudo móveis e eletrodomésticos. A estrutura de rede de atendimento às cerca de 200 lojas consiste na interligação desses pontos de serviço à sede, localizada na região metropolitana de Salvador-BA. O interesse de tráfego de dados se dá somente entre lojas e matriz. Nesse ponto central, estão instalados os servidores corporativos, tanto para as aplicações comerciais, quanto para os serviços acessórios, como correio eletrônico e *Intranet*. Na rede da matriz, também estão concentrados os serviços de comunicação com parceiros, como empresas de cartão de crédito e bancos. Para tráfego de voz, ao contrário dos fluxos das aplicações, há interesse de comunicação entre todos os pontos.

A rede antiga consistia basicamente em uma *VPN* de camada 2 com *Frame Relay* (figura 7-1). Os circuitos virtuais eram iniciados em cada loja e finalizados em um concentrador na sede, compondo topologia *hub-and-spoke* simples. Para atendimento do tráfego de voz, todas as chamadas originadas nas lojas eram enviadas a roteadores instalados na matriz, que, caso o

destino da chamada fosse um ramal da sede, entregavam diretamente ao *PABX* central. Nos casos de chamadas entre lojas, os roteadores centrais comutavam a chamada, alocando o *PVC* específico (*tandem-switching*). Essa solução, já bastante consolidada e estável, segue o padrão *FRF*.11 do *Frame-Relay Forum*. Nenhuma diferenciação era feita no tratamento de dados das aplicações.

A idéia da nova rede era tornar a estrutura mais flexível e escalável, além de torná-la compatível com as novas tecnologias convergentes, centradas no *IP*. Um outro benefício esperado era o melhor desempenho da rede de lojas. Na nova solução, com enlaces ponto-a-ponto, a capacidade nominal de cada um dos enlaces poderia ser atingida, sem as limitações da rede *Frame Relay* para tratamento de tráfego não conforme (acima do *CIR*) em casos de congestionamento.

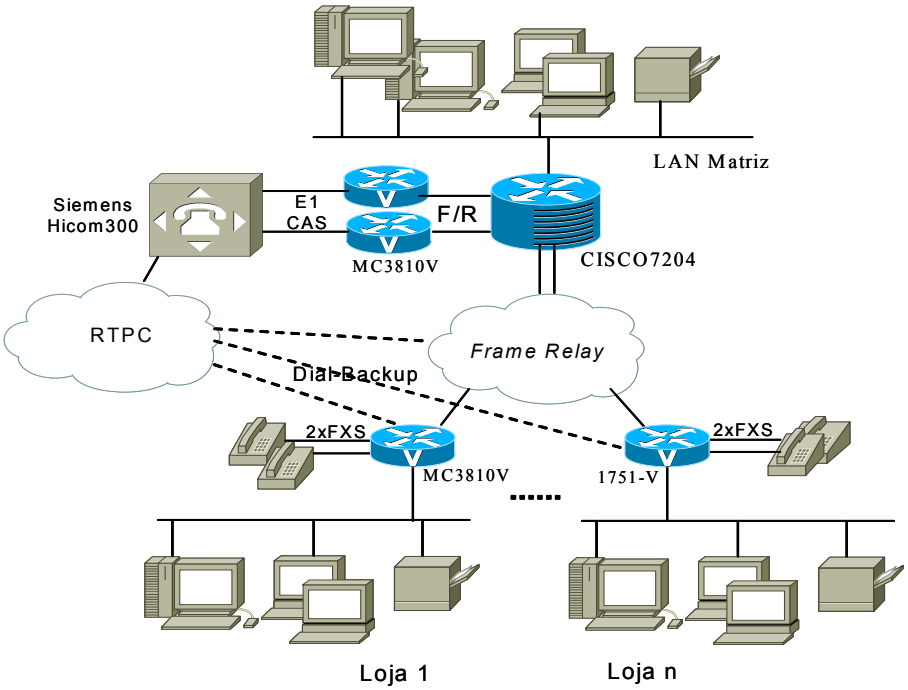


Figura 7-1: Cenário anterior – *VPN L2 - Frame Relay*

O novo projeto deveria prever, ainda, o tratamento diferenciado das aplicações usadas nas lojas, otimizando o uso da banda e alocando prioritariamente mais recursos para aplicações mais importantes. Com isso, seriam atendidos os requisitos de desempenho dessas aplicações, evitando problemas com tempo de resposta em momentos de alto tráfego.

O último grande requisito do projeto era a previsão de redundância na estrutura da sede, atendida originalmente por estrutura única, que deveria ser duplicada, de forma a garantir alta disponibilidade na comunicação com as lojas.

Assim, um novo projeto de rede teve que ser concebido, englobando soluções de integração de voz sobre *IP* (*VoIP*), qualidade de serviço com tratamento diferenciado para as aplicações críticas e alta disponibilidade.

Algumas premissas foram consideradas para a elaboração do projeto:

- A migração de tecnologia não deveria causar interrupção de serviço;
- a qualidade dos serviços de voz deveria, no mínimo, manter os níveis anteriores;
- a nova solução traria ganhos de *performance* para as aplicações de dados relacionadas ao negócio;
- os roteadores das lojas deveriam suportar as novas funções, sem necessidade de troca de *hardware*.

A partir desses pontos, foi proposta a nova arquitetura da rede (figura 7-2), contemplando os seguintes itens:

- Solução de voz sobre *IP*, em substituição à implementação anterior de voz sobre *Frame Relay*;
- qualidade de serviço para as aplicações de dados, inexistente na rede anterior.

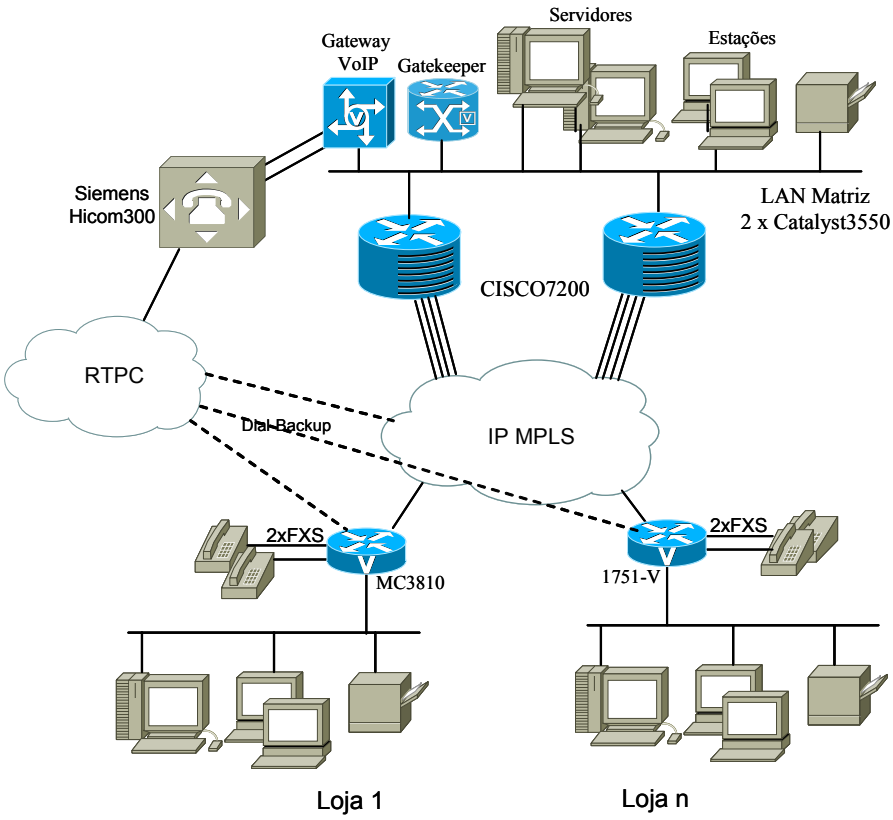


Figura 7-2: Nova topologia (diagrama simplificado)

Completando o cenário, deveria ser fornecida solução de rede, com voz e dados, para outra empresa do grupo. Nessa rede, formada pelas 60 sedes das financeiras, são utilizadas aplicações relacionadas ao negócio distintas da rede de lojas. As duas redes compartilham as aplicações corporativas (correio eletrônico, *Intranet* etc.). A expectativa era que fosse proposta uma solução de integração a baixo custo para a rede das financeiras.

Com esse intuito, a solução considerou o baixo tráfego de aplicações observado nos levantamentos realizados. Somente uma aplicação foi classificada como crítica para o negócio. Essa aplicação implementa os módulos de Cadastro de Clientes, Análise de Perfil e Aprovação de Crédito. Foi considerada importante a aplicação de consulta aos bancos parceiros. Essas consultas são feitas através de servidores centralizados, com a utilização de interface *web* nos clientes.

As demais aplicações poderiam ser tratadas de acordo com o seu padrão normal de operação, sendo consideradas de prioridade normal e sem interferência direta no negócio. Outra observação quanto ao tráfego de dados é que não há interesse de tráfego (troca de dados) entre lojas. Todos servidores e estações das lojas somente se conectam a servidores da sede. A solução de voz deveria prever dois ramais por loja, com capacidade de transportar voz e fax sobre *IP*. Nesse caso, deveriam ser previstas chamadas entre todos os pontos.

Diante dessas considerações e expectativas, o projeto foi elaborado, tendo como ponto de partida a definição de que a solução seria baseada na arquitetura de serviços diferenciados – *DiffServ*.

7.2 Levantamento das aplicações

Essa fase consistiu no estudo da rede e no levantamento das aplicações. De fundamental importância, serviu para a construção dos perfis de tráfego. Além disso, permitiu a classificação gerencial das aplicações, de acordo com a sua importância para o negócio.

Foram usadas duas estratégias. A primeira foi buscar informações a respeito das aplicações críticas junto à equipe de suporte técnico, quando foram identificadas as seguintes aplicações de dados:

- Sistema de automação de lojas;
- sistema de autorização de transações financeiras (cartões de débito e crédito) (TEF);
- sistema de gerência *on-line* (painel).

O acesso à *Intranet* e ao correio eletrônico, feito através de interface *webmail*, foi classificado como de média prioridade. As demais aplicações foram consideradas sem relevância para o negócio.

Na tentativa de obter um perfil de tráfego de cada uma dessas aplicações, foram feitas medições. Para isso, utilizou-se de ferramenta de análise de rede (*Sniffer*), capaz de identificar os fluxos, determinar o percentual de banda utilizado por cada um desses fluxos de dados e mensurar tempo de resposta para aplicações *TCP* (função *ART – Application Response Time*) e ferramentas de medição de fluxos (*netflow*) no roteador central.

A figura 7-3, a seguir, mostra a topologia utilizada nessa etapa.

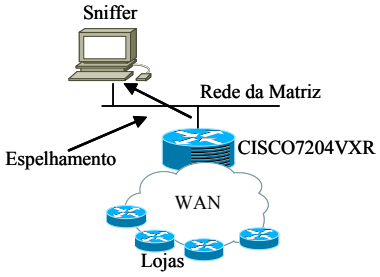


Figura 7-3: Captura de tráfego na matriz

Com o espelhamento da interface de rede local do roteador central, pode-se observar todo o tráfego nos sentidos lojas-matriz e matriz-lojas. A partir desses resultados, o tráfego da rede de lojas foi classificado, conforme a tabela 7-1.

Tabela 7-1: Levantamento das Aplicações

Item	Aplicação	Descrição	Classificação
1	TEF	Comunicação entre servidores de lojas e servidores TEF na matriz para transações financeiras: autorizações de cartão de crédito e débito.	Crítica
Detalhes:	A aplicação funciona sobre *TCP*, sendo a conexão aberta pelo servidor da loja. O servidor TEF tem conexões sempre ativas (X.25) com as administradoras e autorizadoras de cartão.		
2	SAG	Sistema de automação comercial – consulta a estoques centrais, cadastro de clientes, aprovação de crédito etc.	Crítica
Detalhes:	A aplicação funciona sobre *TCP*, sendo a conexão aberta pelo servidor da loja.		
3	Painel	Atualização do sistema central que disponibiliza o volume de vendas de cada loja	Muito importante
Detalhes:	A aplicação funciona sobre *TCP*, sendo a conexão aberta pelo servidor da loja que, a cada 5 minutos, atualiza a base de dados da matriz.		
4	*Intranet*	Acesso ao portal interno para busca de informações de Recursos Humanos, lista de ramais, novidades dos fabricantes parceiros etc.	Importante
Detalhes:	Trata-se de aplicação *web* tradicional. O *pool* de servidores está instalado na matriz e servem na porta *TCP* 80.		
5	Mensagens	Acesso a correio eletrônico, utilizando interface *webmail*	Média
Detalhes:	Os clientes somente acessam o servidor através do navegador, através de interface *webmail* (*TCP* porta 80).		

6	Acesso remoto	Acesso aos microcomputadores das lojas para suporte e manutenção remota, com utilização do software *VNC*	Nenhuma
Detalhes:	Essa aplicação é utilizada pelo pessoal de *help-desk* e suporte em informática para correção de problemas nos microcomputadores das lojas. O módulo servidor é instalado em cada um dos microcomputadores enviados para as lojas. Serve na porta *TCP* 5800.		
7	*Backup*	Rotina diária de *backup* das lojas em *storage* da matriz.	Importante
Detalhes:	Utiliza rotinas automatizadas com software *Veritas* para transferência de arquivos.		
8	Demais aplicações	Acesso a *Internet*, correio eletrônico externo, *messaging* etc.	Indesejáveis
Detalhes:	O tráfego *Internet* passa por um servidor *Proxy*, instalado na matriz. Esse servidor responde, mediante autenticação, na porta *TCP* 3128.		

Para o projeto, foram contabilizadas, adicionalmente, outras aplicações (tabela 7-2).

Tabela 7-2: Aplicações complementares

Item	Aplicação	Descrição	Importância
9	*VoIP* e sinalização de chamadas	Solução de telefonia interna para comunicação entre lojas e lojas-matriz.	Tempo real
10	Gerência, Suporte e consultas *DNS*	Tráfego de gerência de redes, resolução de nomes, roteamento e suporte remoto.	Crítica

Mapeando as aplicações levantadas em classes de serviço *DiffServ*, obteve-se, por conseqüência, os valores de *DSCP* utilizados (tabela 7-3). Devido à pequena quantidade de canais de voz por loja, tipicamente 2, optou-se por concentrar o tráfego de voz e o tráfego de sinalização em uma mesma classe, diminuindo o número de classes de serviço. Além disso, como os enlaces são, normalmente, de baixa velocidade, procurou-se agrupar aplicações em classes com requisitos comuns, a fim de diminuir o número de classes.

Tabela 7-3: Classificação nos roteadores de borda e *switches* de camada 3 da matriz

Aplicação	**Classe**	**Sigla**	**DSCP**
VoIP	Tempo Real	RT	*EF*
TEF	Crítico	C	*AF41*
SAG			
Painel	Semi-crítico	SC	*AF31*
Intranet			
Mensagens			
Acesso remoto	Não interativo	N	*AF21*
Backup			
Internet	Não cooperativo	NC	*AF11*
Suporte, administração e gerência de redes	Suporte	S	*AF33/CS6/CS7*
Demais aplicações	*best effort*	N	*BE - 0*

Para os roteadores internos ao domínio *DiffServ*, que, nesse caso, são os roteadores centrais da sede e os roteadores da rede do provedor, apenas quatro classes de serviço foram definidas, conforme apresentado na tabela 7-4.

Tabela 7-4: Classificação no *backbone*

Aplicação	**Classe**	**Sigla**	**DSCP**
VoIP	Tempo Real	RT	*EF*
TEF e SAG	*Business*	BU	*AF41, AF31, AF33, AF21, CS6 e CS7*
Painel			
Intranet			
Correio eletrônico			
Serviços de *logon, DNS, g*erência e administração de rede			
Internet	Não cooperativo	NC	*AF11*
Demais	*best effort*	BE	*0*

7.3 Modalidades de acessos

No projeto da rede corporativa, foram consideradas as seguintes formas de acesso:

7.3.1 Acesso dedicado

Os usuários são conectados através de roteador de acesso (*CE – Customer Edge*) (figura 7-4), que por sua vez está conectado a um roteador de fronteira da rede *MPLS* do provedor (*ELSR - Edge Label Switching Router*). Neste tipo de acesso são utilizados enlaces dedicados com protocolos nível 2 *PPP*. Esses enlaces possuem velocidades de 128Kbps e 256Kbps.

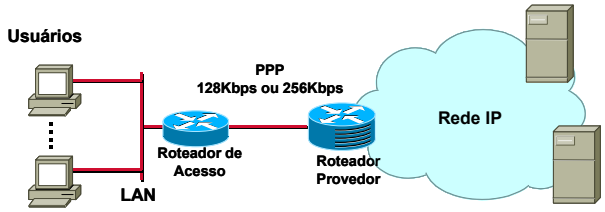


Figura 7-4: Acesso dedicado

7.3.2 Acesso direto

Os usuários são conectados em interface *LAN*, diretamente em roteadores de distribuição (figura 7-5). Neste tipo de acesso são utilizados enlaces *LAN* de 10, 100 ou 1000Mbps; ou *WLAN bridge*, de 11 ou 54Mbps. A função de distribuição, nesse caso, é feita pelos *switches* de camada 3, instalados na matriz.

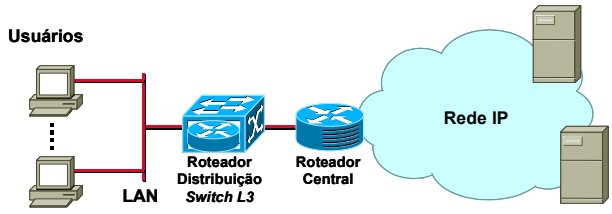


Figura 7-5: Acesso direto via *LAN* ou *WLAN*

7.4 Taxas de acesso

Foram definidas as taxas suportadas para o acesso *IP* Dedicado. A tabela 7-5 apresenta as taxas definidas.

Tabela 7-5: Velocidade dos acessos

Camada 2	**Velocidades de Acesso**	**Localização**
PPP	128Kbps	Lojas
PPP	256Kbps	Lojas
Ethernet	10Mbps	Sede
WLAN (802.11b/g)	11Mbps / 54Mbps	Sede
FastEthernet	100Mbps	Sede
GigabitEthernet	1000Mbps	Sede

7.5 Política de *QoS*

O trabalho de implementação das classes de serviços é composto das seguintes definições:

- Arquitetura de *QoS*, que fixa regras de como aplicar os mecanismos nos diversos pontos de contenção da rede, para implementação dos condicionadores de tráfego *TC (Traffic Conditioning*) e *PHB* (*Per Hop Behavior*). A combinação das funções de *QoS* permite condicionar o tráfego das classes de serviços de forma a obter os índices de *performance* requeridos;
- A definição das velocidades no interior da rede e a divisão dessa banda pelas classes de serviços são o ponto essencial da implementação.

7.6 Descrição da arquitetura

A arquitetura de *QoS* será baseada na arquitetura de Serviços Diferenciados (*DiffServ*). A arquitetura *DiffServ* propõe uma forma de implantar *QoS* em redes *IP* com a capacidade de ampliação requerida por grandes redes.

7.7 Implementação das classes de serviço

A implementação da arquitetura de *QoS* divide-se em duas partes:

- Classes de serviços e condicionamento do tráfego (*TC*) nos acessos;
- Classes de serviços e *PHB* do núcleo da rede *IP*.

A separação deve-se às definições da arquitetura *DiffServ* que especifica a necessidade de:

- Aplicar condicionamento de tráfego (*TC*) na borda do domínio, adequando o tráfego às condições fixadas pelas políticas de *QoS*;
- Reservar banda e priorizar as aplicações no *backbone* através da aplicação de *PHBs*, de forma a garantir as políticas estabelecidas.

Os processos *TC* são implementados com mecanismos de policiamento, suavização, descartes e gerência de filas/escalonamento. Os *PHB* são implementados com mecanismos de gerência de filas/escalonamento e descartes.

7.7.1 Política *TCIn*

Na política *TCIn*, aplicada na matriz, são consideradas 6 classes. A classe suporte não é considerada nessa política, pois o tráfego é originado no roteador e não pelo usuário da *LAN*.

7.7.2 Política *TCOut*

Na política *TCOut*, aplicada nas lojas, são implementadas 7 filas, uma para cada classe. A seguir são apresentadas as premissas para a política *TCOut*:

Classe Tempo Real (*VoIP*)

- 13Kbps para cada canal de voz (*cRTP*)
- Em enlaces superiores a 512Kbps, 29Kbps para cada canal de voz (sem *cRTP*)

Classe Crítico

- 16Kbps para enlaces de 128Kbps
- 32Kbps para enlaces de 256Kbps
- Tráfego excedente remarcado para semi-crítico

Classe Semi-Crítico

- 24Kbps ou 20% do enlace

– Tráfego excedente remarcado para normal

Classe Não Interativo

– 18Kbps ou 15% do enlace

Classe Normal (*best effort*)

– Mínimo de 18Kbps ou 15% do enlace

Classe de Suporte

– Máximo de 16Kbps em enlaces de 128Kbps ou superiores

Classe Não Cooperativo (*scavenger*)

– Classe utilizada para alocar aplicações não importantes para a empresa e tráfegos maliciosos

– Esta classe será considerada menos prioritária que *best effort* com reserva de 5% da banda

7.7.3 Política *PHB*

Na política *PHB*, aplicada nos roteadores internos (*backbone*), são consideradas 4 classes:

Classe Tempo Real (*VoIP*)

– Pacotes com marcação *EF*

– Até 30% da banda

– Descartes com *Tail Drop*

Classe *Business*

– Pacotes marcados com *AF41, AF31, AF33, AF21, CS6 e CS7*

– Reserva de até 45% da banda

– Descartes com *WRED*

Classe Normal

– Pacotes marcados com *CS0*, *DSCP0* e *IPP0*

– Reserva de até 15% da banda

– Descartes com *WRED*

Classe Não Cooperativo

– Pacotes marcados com *AF*11

– Até 5% da banda

– Descartes com *WRED*

7.8 Seleção dos mecanismos de *QoS* para as políticas

A seguir são descritos os mecanismos implementados nas políticas de *QoS*.

7.8.1 Marcação do tráfego

A marcação possibilita a diferenciação dos pacotes *IP* nas classes e nos grupos de aplicações no seu acesso. Foi utilizada marcação baseada em *DSCP*. Essa etapa buscou também considerar a possibilidade de portabilidade das marcações com *DSCP* internas ao domínio *DS* (*DiffServ*) para *MPLS EXP* na rede do provedor. Desta forma, se considerou a agregação das classes no *backbone* em, no máximo, 8 classes, visto que o campo *DSCP* do pacote *IP* não pode ser acessado em uma rede *MPLS* e o campo *EXP* do *MPLS* suporta apenas 8 classes de serviço.

O padrão de marcação definido acomoda as classes de serviços e os grupos de aplicações dessas classes. A marcação *DSCP* permitirá oferecer mais opções para futuras revisões da política de *QoS*. No entanto, as marcações internas do *backbone* estão limitadas aos oito valores disponíveis no *MPLS EXP*.

A tabela 7-6 apresenta a associação entre as classes *DiffServ* e *MPLS EXP* ou *IP Precedence*. Essa associação visou facilitar o mapeamento na rede *MPLS* do provedor e a compatibilidade com equipamentos sem suporte a marcação com *DSCP*.

Tabela 7-6: Correspondência entre os valores de *DSCP* e *MPLS EXP* / *IPP*

IPP/MPLS EXP	**DSCP**	**Classe**	**Tráfego**
5	*EF*	Tempo real	Voz
4	*AF41*	Crítico	Automação comercial
3	*AF31, AF33*	Semi-crítico	Painel, *Intranet,* correio-eletrônico, suporte
2	*AF21*	Normal	*Backup* e acesso remoto
1	*AF11*	Não cooperativo	*Internet*
0	*CS0*	*Best Effort*	Demais

7.8.2 Demais mecanismos: policiamento e suavização, CAC, filas e descartes

Policiamento e suavização

O policiamento controla o volume de tráfego das aplicações e classes nas interfaces de entrada e saída. Algumas das razões para o seu emprego são relacionadas a seguir:

- A banda consumida por classe deve ser controlada para que não exceda o contratado ou acordado para a classe;
- A fila prioritária empregada nos roteadores para o tráfego tempo real interativo deve ter seu uso limitado a volumes de tráfego que não comprometam o desempenho dessas aplicações e das aplicações das demais classes de serviços;
- Aplicações não-adaptativas, baseadas no protocolo *UDP*, não reagem a notificações de congestionamento e tendem a ampliá-lo ou prejudicar as aplicações que reagem (*TCP*); para evitar essas situações as suas taxas devem ser limitadas. Isso pode ser feito através do

policiamento propriamente dito (*traffic policing*) ou da suavização (*traffic shaping)*. O primeiro é baseado no descarte do tráfego que excede os limites especificados. O segundo restringe a emissão de tráfego a uma taxa média ou máxima especificadas, retendo os excessos nos *buffers* de saída das interfaces, até conseguirem vazão ou serem descartados. O *traffic policing* pode ser aplicado na saída e entrada das interfaces. O *traffic shaping* somente atua nas interfaces de saída.

CAC – controle de admissão

Nesse projeto o controle de admissão é feito localmente (nos roteadores) pela própria limitação de portas de voz. Os roteadores padrão das lojas têm duas interfaces de voz disponíveis e os gateways da matriz dispõem de 60 canais. Não é feito redirecionamento das chamadas em caso de exaustão dos recursos.

Filas

A gerência de filas e escalonamento implementa a reserva de banda entre as classes de serviços nas interfaces de saída dos roteadores. A alocação de banda mínima por classe é fundamental na garantia dos índices exigidos de banda*,* atraso e *jitter* requisitados. Ao alocar banda para uma classe, está sendo reservada uma fila que terá o direito de competir pela fila de saída do roteador (*TX-queue*) na taxa especificada pela banda. Para aplicações com requisitos rigorosos de atraso e *jitter*, como as da classe tempo real, é recomendada a alocação de banda na fila prioritária dos

roteadores (*LLQ*). A fila prioritária é uma fila de *software* que tem prioridade na alocação da *TX-queue*, quando a banda alocada para essa fila não é ultrapassada.

Para as classes de dados, será utilizado o escalonador *CBWFQ*.
Vale ressaltar que a alocação de banda por classe é conservativa, ou seja, a banda alocada não utilizada por uma classe poderá ser utilizada pelas demais classes se houver necessidade.

Descartes

A função de descartes atua nas situações de congestionamento quando as filas de saída dos roteadores esgotam os seus recursos. Nessas situações, perdas de pacotes ocorrem e as aplicações reagem de formas diferentes a essas perdas.

Essa função pode implementar duas estratégias para tratar perdas de pacotes: *tail drop* ou *RED* (*random drop*). O *RED* é empregado para aplicações que reagem à perda de pacotes reduzindo a sua taxa de emissão de pacotes, adaptando-se assim às situações de congestionamento. O *random drop* possibilita antecipar as situações extremas de congestionamento e prioriza as aplicações ao descartar pacotes nas filas.

A estratégia *tail drop* é indicada para aplicações não-adaptativas como as da classe Tempo Real. Essas aplicações não reagem a congestionamento e, portanto, não poderiam tirar vantagem do *random drop*.

A classe “crítico” se caracteriza por aplicações de dados de baixo consumo de banda e que requerem baixos índices de atraso. O atraso nesta classe poderá ser controlado através do ajuste da quantidade de pacotes na fila (profundidade). Portanto, a estratégia de descartes definida para a classe foi a *tail drop*, a fim de evitar a acomodação de grandes rajadas de tráfego e, assim, diminuir o atraso na classe, resultante dessas rajadas.

Para as demais classes, foi definida a implementação do mecanismo *Weighted Random Early Detection* (*WRED*), que regula, inclusive, as prioridades na perda de pacotes entre os níveis de precedência.

Marcação, policiamento, gerência de filas e descartes não são as únicas funções que implementam os *PHB* ou condicionadores de tráfego na *DiffServ*, mas são as fundamentais. Outras funções acessórias serão empregadas na implementação de *QoS* nos enlaces, como a fragmentação, compressão e controle de admissão.

Os mecanismos descritos serão aplicados às configurações de *TC* e *PHB* na borda da Rede *IP* e no *backbone* em concordância com o provedor. A descrição resumida de como aplicá-los pode ser encontrada a seguir.

7.9 Política aplicada nos roteadores das lojas

Aplicação no sentido Loja-*Backbone*. No sentido *Backbone*-Loja, a mesma política é usada, porém, sem a necessidade de classificar e marcar o tráfego (tabela 7-7).

Tabela 7-7: Política de saída nos roteadores das lojas

		Tempo Real	Crítico	Semi-crítico	Suporte	Normal	Não cooperativo	*Best Effot*
TC_Out	Marcação *DSCP*	*EF*	*AF41*	*AF31*	*AF33*	*AF21*	*AF11*	0
	Policiamento	Banda máxima necessária ao número de canais e sinalização	Tráfego excedente remarcado para AF31	Tráfego excedente remarcado para *BE* (0)	não	não	não	Não
	Fila	*LLQ*	*CBWFQ*	*CBWFQ*	*CBWFQ*	*CBWFQ*	*CBWFQ*	*CBWFQ*
	Descarte	*Tail Drop*	*Tail Drop*	*WRED*	*WRED*	*WRED*	*WRED*	*WRED*

7.10 Políticas aplicadas na Matriz

TC - Classificação e marcação:

Saída do tráfego do *switch* de camada 3 para os roteadores centrais (tabela 7-8).

Tabela 7-8: Política *TC* de saída nos *switches* de camada 3 da Matriz

		Tempo Real	Crítico	Semi-crítico	Suporte	Normal	Não cooperativo	*best effort*
TC_*LAN_Out*	Marcação *DSCP*	*EF*	*AF41*	*AF31*	*AF33*	*AF21*	*AF11*	0
	Classificação	não	sim	sim	sim	sim	sim	sim
	Policiamento	não	não	não	não	não	não	não
	Fila	não	não	não	não	não	não	não
	Descarte	não	não	não	não	não	não	não

Saída do tráfego dos elementos centrais da rede de voz (*gateways e gatekeepers*) para os roteadores centrais (tabela 7-9).

Tabela 7-9: Política *TC* de saída nos elementos da rede de voz da matriz

		Tempo Real
*TC_LAN*_V_*Out*	Marcação *DSCP*	*EF*
	Classificação	sim
	Policiamento	não
	Fila	não
	Descarte	não

PHB - Saída do tráfego dos roteadores centrais para a rede do provedor (tabela 7-10).

Tabela 7-10: Política *PHB* de saída dos roteadores centrais para o provedor

		Tempo Real	*Business*	Não cooperativo	*Best Effot*
*PHB*_Sede	Marcação *DSCP*	*EF*	*AF41, AF31, AF33, AF21, CS6 e CS7*	*AF11*	0
	Policiamento	30% da banda	não	não	não
	Fila	*LLQ*	*CBWFQ*	*CBWFQ*	*CBWFQ*
	Descarte	*Tail Drop*	*WRED*	*WRED*	*WRED*

As outras políticas descrevem o comportamento da rede para as conexões internas. Por exemplo, o tratamento dado aos enlaces de comunicação com a utilização de *WLAN bridges* entre prédios da sede. Além dessas, foram definidos também os padrões adotados na rede da financeira BPN, pertencente ao mesmo grupo e com sede no mesmo *campus*, quando foram usadas as mesmas estratégias de identificação e medição do tráfego e demais etapas do projeto de *QoS*.

7.11 Medições e análises

Com o objetivo de certificar a implementação de *QoS* proposta, foi elaborado um plano de testes. Esses testes consistiram em medições dos parâmetros de *QoS* anterior e posteriormente à aplicação das políticas, seguindo basicamente o seguinte roteiro:

7.11.1 Reserva de banda para as classes

Nesses testes e medições, foram utilizadas ferramentas para geração e coleta de tráfego, analisadores de protocolos e recursos de gerenciamento dos roteadores.

7.11.1.1 Sem *QoS*

Nesse ponto, a aplicação da política foi retirada das interfaces dos roteadores, conforme mostrado na figura 7-6, a seguir. Como não há política aplicada, todos os fluxos são tratados com a mesma prioridade, seguindo o comportamento padrão do *TCP/IP* (*best effort*).

Gerando tráfego nas classes, tentou-se chegar a níveis de saturação do canal. Isso foi observado através de comandos do roteador. Com a utilização do software analisador de protocolos, observou-se que os fluxos de maior consumo variaram na ocupação do enlace (figura 7-7). Nesse instante, não houve sucesso na tentativa de completar chamadas de voz entre matriz e loja.

```
conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Loja_200(config)#interface multilink 1
Loja_200(config-if)#no service-policy output TC_Out_128K
Loja_200#sh policy-map interface mul 1

sh int mul 1
Multilink1 is up, line protocol is up
  Hardware is multilink group interface
  Internet address is 200.223.217.98/30
  MTU 1500 bytes, BW 128 Kbit, DLY 100000 usec,
     reliability 255/255, txload 245/255, rxload 253/255
  Encapsulation PPP, loopback not set
  DTR is pulsed for 2 seconds on reset
  LCP Open, multilink Open
  Open: IPCP, CDPCP
  Last input 00:00:02, output never, output hang never
  Last clearing of "show interface" counters 00:01:42
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 5956
  Queueing strategy: fifo
  Output queue :40/40 (size/max)
  30 second input rate 127000 bits/sec, 17 packets/sec
  30 second output rate 123000 bits/sec, 65 packets/sec
     1742 packets input, 1638214 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     6415 packets output, 1589158 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

Figura 7-6: Desativação da Política de *QoS* da interface do ponto remoto

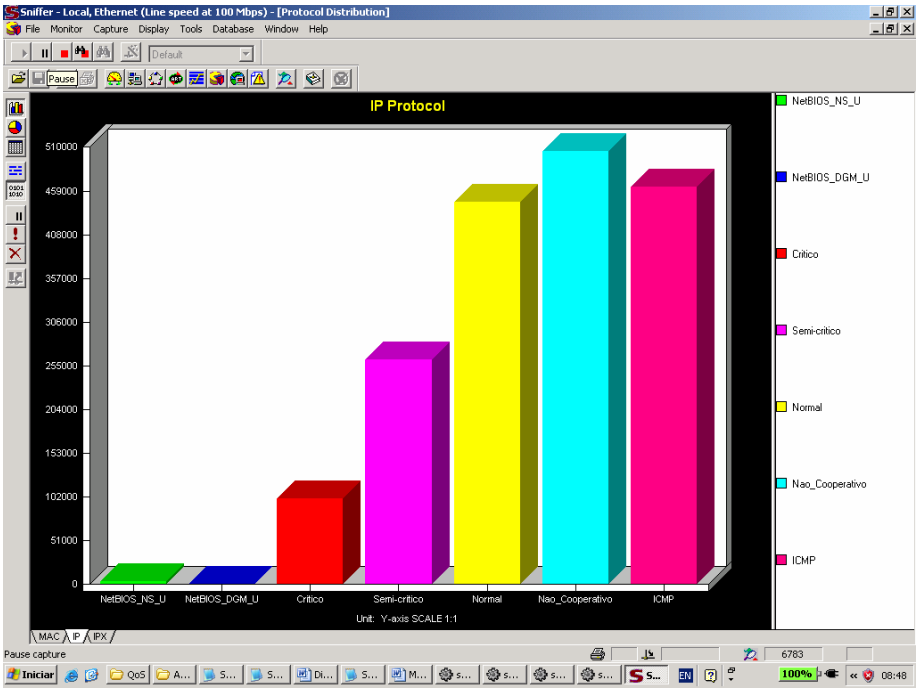


Figura 7-7: Medição de tráfego na interface sem política

7.11.1.2 Com *QoS*

A política foi aplicada às interfaces dos roteadores (figura 7-8) e o mesmo procedimento de geração de tráfego foi seguido (figura 7-9). Nesse caso, consegue-se observar claramente a reserva e ocupação da banda total de forma estratificada e de acordo com as definições de projeto.

```
Multilink1 is up, line protocol is up
  Hardware is multilink group interface
  Internet address is 200.223.217.98/30
  MTU 1500 bytes, BW 128 Kbit, DLY 100000 usec,
     reliability 255/255, txload 223/255, rxload 243/255
  Encapsulation PPP, loopback not set
  DTR is pulsed for 2 seconds on reset
  LCP Open, multilink Open
  Open: IPCP, CDPCP
  Last input 00:00:00, output never, output hang never
  Last clearing of "show interface" counters 00:26:42
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 100158
  Queueing strategy: weighted fair
  Output queue: 117/1000/64/93539/0 (size/max total/threshold/drops/interleaves)
     Conversations  7/8/32 (active/max active/max total)
     Reserved Conversations 4/4 (allocated/max allocated)
     Available Bandwidth 39 kilobits/sec
  30 second input rate 122000 bits/sec, 61 packets/sec
  30 second output rate 112000 bits/sec, 110 packets/sec
     54773 packets input, 18708847 bytes, 3 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     86588 packets output, 15066658 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

Figura 7-8: Comprovação de ativação da política

```
Loja_200#sh policy int mul 1
 Multilink1
  Service-policy output: TC_Out_128K
    Class-map: classe_voz (match-any)
      23458 packets, 1225981 bytes
      30 second offered rate 21000 bps, drop rate 0 bps
      Match: access-group name voz_acl
        23458 packets, 1225981 bytes
        30 second rate 21000 bps
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 26 (kbps) Burst 650 (Bytes)
        (pkts matched/bytes matched) 16842/881949
        (total drops/bytes drops) 0/0
     Class-map: classe_sup (match-any)
      2050 packets, 98496 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: access-group name rp_acl
        0 packets, 0 bytes
        30 second rate 0 bps
      Match: access-group name sup_acl
        2050 packets, 98496 bytes
        30 second rate 0 bps
      Queueing
        Output Queue: Conversation 41
        Bandwidth 8 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 2050/98496
        (depth/total drops/no-buffer drops) 0/0/0
      QoS Set
       ip dscp af33
          Packets marked 2050
    Class-map: classe_critico (match-any)
      16696 packets, 1602816 bytes
      30 second offered rate 14000 bps, drop rate 0 bps
      Match: access-group name critico_acl
        16696 packets, 1602816 bytes
        30 second rate 14000 bps
      Queueing
        Output Queue: Conversation 42
        Bandwidth 16 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 16696/1602816
        (depth/total drops/no-buffer drops) 7/0/0
     police:
         cir 24000 bps, bc 1500 bytes conformed 16696 packets, 1602816 bytes; actions:set-
dscp-transmit af41
       exceeded 0 packets, 0 bytes; actions: set-dscp-transmit af31
       conformed 14000 bps, exceed 0 bps
    Class-map: classe_semicritico (match-any)
      27866 packets, 4458560 bytes
      30 second offered rate 42000 bps, drop rate 0 bps
      Match: access-group name semicritico_acl
        27866 packets, 4458560 bytes
        30 second rate 42000 bps
      Queueing
        Output Queue: Conversation 43
        Bandwidth 24 (kbps)
        (pkts matched/bytes matched) 27867/4458720
        (depth/total drops/no-buffer drops) 14/0/0
         exponential weight: 9
         mean queue depth: 7
     QoS Set
       ip dscp af31
          Packets marked 27868
```

```
... Continuação

     police:
         cir 24000 bps, bc 1500 bytes
       conformed 15977 packets, 2556320 bytes; actions: set-dscp-transmit af31
       exceeded 11891 packets, 1902560 bytes; actions: set-dscp-transmit af21
       conformed 23000 bps, exceed 17000 bps
    Class-map: classe_nc (match-any)
      17349 packets, 9221397 bytes
      30 second offered rate 87000 bps, drop rate 53000 bps
      Match: access-group name nc_acl
        16778 packets, 9127232 bytes
        30 second rate 87000 bps
      QoS Set
       ip dscp default
          Packets marked 17343
      Queueing
        Output Queue: Conversation 44
        Bandwidth 8 (kbps)
        (pkts matched/bytes matched) 17264/9214561
        (depth/total drops/no-buffer drops) 21/7732/0
         exponential weight: 9
         mean queue depth: 23
    Class-map: class-default (match-any)
      79234 packets, 12167850 bytes
      30 second offered rate 109000 bps, drop rate 105000 bps
      Match: any
```

Figura 7-9: Funcionamento da política

7.11.2 Atraso e *Jitter*

Para as medições de atraso e *jitter* não se considerou necessária a utilização de fluxos de todas as classes, visto que esses parâmetros são essenciais apenas para alguns tipos de tráfego. Aqui, houve concorrência de tráfego de voz (*RTP*) com tráfego *best effort* injetado na rede.

7.11.2.1 Sem *QoS*

Nesse ponto, a aplicação da política foi retirada das interfaces dos roteadores, conforme mostrado anteriormente na figura 7-6.

a. Sem injeção de tráfego

Nesse caso as chamadas foram completadas com qualidade satisfatória e os índices de latência e *jitter* foram observados através da linha de comando dos roteadores e medidos através do Cisco *SAA* (*Service Assurance Agent*) (figura 7-10). A interpretação dos parâmetros fornecidos pela ferramenta está descrita na tabela 7-11.

```
RTMatriz01#sh rtr collection-statistics 1
Entry number: 1
Start Time Index: 14:57:14.229 GMT-3 Tue Mar 7 2006
Number of successful operations: 7
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
NumOfRTT: 70  RTTAvg: 39    RTTMin: 27    RTTMax: 181
RTTSum: 2777  RTTSum2: 191373
Packet Loss Values:
PacketLossSD: 0     PacketLossDS: 0
PacketOutOfSequence: 0     PacketMIA: 0  PacketLateArrival: 0
InternalError: 0    Busies: 0
Jitter Values:
NumOfJitterSamples: 63
MinOfPositivesSD: 1         MaxOfPositivesSD: 15
NumOfPositivesSD: 6         SumOfPositivesSD: 43       Sum2PositivesSD: 1245
MinOfNegativesSD: 1         MaxOfNegativesSD: 10
NumOfNegativesSD: 18        SumOfNegativesSD: 201      Sum2NegativesSD: 3329
MinOfPositivesDS: 1         MaxOfPositivesDS: 7
NumOfPositivesDS: 8         SumOfPositivesDS: 15       Sum2PositivesDS: 59
MinOfNegativesDS: 1         MaxOfNegativesDS: 3
NumOfNegativesDS: 8         SumOfNegativesDS: 14       Sum2NegativesDS: 30
Interarrival jitterout: 0  Interarrival jitterin: 0
One Way Values:
NumOfOW: 70
OWMinSD: 16   OWMaxSD: 171  OWSumSD: 2054 OWSum2SD: 141142
OWMinDS: 9    OWMaxDS: 17   OWSumDS: 723  OWSum2DS: 7559
```

Figura 7-10: *Jitter* e atraso sem concorrência do tráfego de dados

Tabela 7-11: Principais informações fornecidas pela medição de *jitter* através do *Cisco SAA*

Variável	Descrição
RTTMin, RTTMax e *RTTAvg*	Valores de *Round Trip Time* mínimo, máximo e médio
MaxOfPositivesSD e *MaxOfNegativesSD*	Valores de *jitter* máximos positivos e negativos entre os pontos de origem (*S – Source*) e destino (*D – Destination*) de medição
MaxOfPositivesDS e *MaxOfNegativesDS*	Valores de *jitter* máximos positivos e negativos entre os pontos de origem destino (*D – Destination*) e (*S – Source*) de medição

b. Com injeção de tráfego

Não havendo diferenciação, o tráfego de voz passou a concorrer com o tráfego injetado. O congestionamento provocado, além de aumentar o tempo dos pacotes nas filas de saída das interfaces, passou a provocar descartes, comprometendo consideravelmente a qualidade das chamadas. Relacionando essa constatação à observação dos parâmetros de *QoS*, pôde-se notar o aumento do atraso e do *jitter* (figura 7-11) .

7.11.2.2 Com QoS

Foram, novamente, aplicadas as políticas às interfaces.

a. Sem fragmentação e intercalação (LFI)

Mesmo com injeção de tráfego até níveis de saturação do canal, foi observado que o tráfego de voz teve banda assegurada. Além disso, pela política de filas adotada, *LLQ - Low Latency Queueing*, combinada com *CBWFQ - Class-Based Weighted Fair-Queueing*, enquanto há pacotes de voz nas filas, até o limite da reserva pré-alocada, esses pacotes são despachados. Isso mantém o atraso dentro de valores aceitáveis.

```
sh rtr collection-statistics 1
Entry number: 1
Start Time Index: 06:41:57.650 UTC Wed Mar 8 2006
Number of successful operations: 37
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 2
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
NumOfRTT: 370 RTTAvg: 204   RTTMin: 13    RTTMax: 1016
RTTSum: 75693 RTTSum2: 47460483
Packet Loss Values:
PacketLossSD: 0     PacketLossDS: 0
PacketOutOfSequence: 0     PacketMIA: 0  PacketLateArrival: 0
InternalError: 2    Busies: 0
Jitter Values:
MinOfPositivesSD: 86        MaxOfPositivesSD: 91
NumOfPositivesSD: 0         SumOfPositivesSD: 1841     Sum2PositivesSD: 157766
MinOfNegativesSD: 11        MaxOfNegativesSD: 16
NumOfNegativesSD: 96        SumOfNegativesSD: 1445     Sum2NegativesSD: 21823
MinOfPositivesDS: 1         MaxOfPositivesDS: 76
NumOfPositivesDS: 0         SumOfPositivesDS: 68       Sum2PositivesDS: 61
MinOfNegativesDS: 1         MaxOfNegativesDS: 14
NumOfNegativesDS: 25        SumOfNegativesDS: 28       Sum2NegativesDS: 40
Interarrival jitterout: 0  Interarrival jitterin: 0
One Way Values:
NumOfOW: 370
OWMinSD: 7    OWMaxSD: 1010 OWSumSD: 73128      OWSum2SD: 46444228
OWMinDS: 6    OWMaxDS: 11   OWSumDS: 2565 OWSum2DS: 17865
```

Figura 7-11: *Jitter* e atraso com concorrência do tráfego de dados

O maior problema, entretanto, eram os cortes (picotamento) nas conversações. Como o tráfego injetado gerava pacotes de até 1500 *bytes*, a cada transmissão de um desses pacotes, o tempo de serialização ultrapassava o *jitter* máximo tolerado. Isso pôde ser observado através de comandos operacionais dos roteadores.

Tempo de Serialização = Tamanho do Pacote / Velocidade da Interface

b. Com LFI

Foram aplicados os parâmetros de fragmentação e intercalação, fazendo com que um pacote levasse, no máximo, 10ms para ser transmitido. O tamanho do fragmento gerado varia de acordo com a velocidade do enlace. No caso em questão, com a velocidade do enlace sendo de 128Kbps, o fragmento pode ser calculado da seguinte forma:

Tamanho do Fragmento = Tempo de Transmissão x Velocidade do Enlace

Para o cenário analisado para os testes e medições, com enlaces de 128Kbps:

Tamanho do Fragmento = 10ms x 128Kbps = 1280 *bits* = 160 *bytes*

O funcionamento do mecanismo de *LFI* pôde ser comprovado por comandos operacionais nos roteadores (figura 7-12 e 7-13). Após essa aplicação, não se observou mais problemas de

qualidade de voz, mesmo com a ocupação máxima do enlace. Com esses testes foi constatada, portanto, a eficácia da política para o cenário apresentado.

```
sh rtr collection-statistics 1
Entry number: 1
Start Time Index: 09:03:49.625 UTC Wed Mar 8 2006
Number of successful operations: 7
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
NumOfRTT: 70  RTTAvg: 47    RTTMin: 20    RTTMax: 94
RTTSum: 3344  RTTSum2: 178846
Packet Loss Values:
PacketLossSD: 0     PacketLossDS: 0
PacketOutOfSequence: 0     PacketMIA: 0  PacketLateArrival: 0
InternalError: 0    Busies: 0
Jitter Values:
MinOfPositivesSD: 2 MaxOfPositivesSD: 26
NumOfPositivesSD: 0 SumOfPositivesSD: 406      Sum2PositivesSD: 15589
MinOfNegativesSD: 1 MaxOfNegativesSD: 16
NumOfNegativesSD: 44       SumOfNegativesSD: 332      Sum2NegativesSD: 4028
MinOfPositivesDS: 1 MaxOfPositivesDS: 16
NumOfPositivesDS: 0 SumOfPositivesDS: 191      Sum2PositivesDS: 1599
MinOfNegativesDS: 1 MaxOfNegativesDS: 14
NumOfNegativesDS: 31       SumOfNegativesDS: 157      Sum2NegativesDS: 1295
Interarrival jitterout: 0  Interarrival jitterin: 0
One Way Values:
NumOfOW: 70
OWMinSD: 9    OWMaxSD: 74   OWSumSD: 1951 OWSum2SD: 73119
OWMinDS: 9    OWMaxDS: 28   OWSumDS: 1393 OWSum2DS: 29261
One Way Values:
NumOfOW: 370
OWMinSD: 7    OWMaxSD: 1010 OWSumSD: 73128      OWSum2SD: 46444228
OWMinDS: 6    OWMaxDS: 11   OWSumDS: 2565 OWSum2DS: 17865
```

Figura 7-12: Jitter e atraso com concorrência do tráfego de dados e com *LFI* aplicado

```
debug ppp multilink fragments
Multilink fragments debugging is on
Loja_200#
Mar  8 09:06:14.924: Se0/0 MLP: I frag 8003E88F size 160 direct
Mar  8 09:06:14.928: Se0/0 MLP: I frag 4003E890 size 34 direct
Mar  8 09:06:14.936: Se0/0 MLP: I frag 8003E891 size 160 direct
Mar  8 09:06:14.940: Se0/0 MLP: I frag 4003E892 size 34 direct
Mar  8 09:06:14.944: Se0/0 MLP: O frag 0002147B size 160
Mar  8 09:06:14.944: Se0/0 MLP: O frag 4002147C size 94
Mar  8 09:06:14.948: Se0/0 MLP: I frag 8003E893 size 160 direct
Mar  8 09:06:14.960: Se0/0 MLP: O frag 8002147D size 160
Mar  8 09:06:14.960: Se0/0 MLP: O frag 0002147E size 160
Mar  8 09:06:14.960: Se0/0 MLP: I frag 0003E894 size 160 direct
Mar  8 09:06:14.968: Se0/0 MLP: I frag 0003E895 size 160 direct
Mar  8 09:06:14.980: Se0/0 MLP: O frag 0002147F size 160
Mar  8 09:06:14.980: Se0/0 MLP: O frag 40021480 size 94
Mar  8 09:06:14.980: Se0/0 MLP: I frag 0003E896 size 160 direct
Mar  8 09:06:14.992: Se0/0 MLP: I frag 0003E897 size 160 direct
Mar  8 09:06:14.996: Se0/0 MLP: O frag 80021481 size 160
Mar  8 09:06:14.996: Se0/0 MLP: O frag 00021482 size 160
Mar  8 09:06:15.000: Se0/0 MLP: I frag 0003E898 size 160 direct
Mar  8 09:06:15.012: Se0/0 MLP: I frag 0003E899 size 160 direct
Mar  8 09:06:15.016: Se0/0 MLP: O frag 00021483 size 160
Mar  8 09:06:15.016: Se0/0 MLP: O frag 40021484 size 94
Mar  8 09:06:15.020: Se0/0 MLP: I frag 0003E89A size 160 direct
Mar  8 09:06:15.032: Se0/0 MLP: I frag 0003E89B size 160 direct
Mar  8 09:06:15.032: Se0/0 MLP: O frag 80021485 size 160
```

Figura 7-13: Comprovação do funcionamento dos mecanismos de *LFI*

8 CONCLUSÕES E TRABALHOS FUTUROS

Esse trabalho procurou abordar as principais técnicas, mecanismos e arquiteturas disponíveis para a construção de um projeto de qualidade de serviço para redes corporativas. Essa diferenciação entre as *Intranets* e a *Internet* se torna necessária quando é estabelecido que o principal objetivo de um projeto de *QoS* é atender aos requisitos de tráfego próprios da rede em questão. Neste ponto, redes corporativas e *Internet* diferem sobremaneira.

Além disso, a etapa que inclui a medição, caracterização e projeção de tráfego em redes privadas pode ser considerada uma tarefa muito mais simples que na *Internet*. Esta última envolve variáveis dificilmente previsíveis (enlaces, roteadores, *PoPs*, *ISPs*), havendo, por esse motivo, grandes dificuldades na definição dos pontos de medição, coleta e análise de amostras verdadeiramente representativas do perfil de tráfego. Nas redes corporativas, de topologia bem definida, interesses de tráfego conhecidos e fluxos de dados previsíveis, essa atividade não demanda tão grandes problemas como na *Internet*.

Seguindo a caracterização, tanto na profundidade e detalhamento da pesquisa, como na implementação do estudo de caso, optou-se pela arquitetura de serviços diferenciados - *DiffServ*, recomendada para redes maiores por proporcionar maior escalabilidade e granularidade. A partir daí, foram apresentados os mecanismos de *QoS* que apoiam a implementação da *DiffServ*. Foram detalhadas as fases de classificação e marcação do tráfego, formando os agregados de fluxo, gerenciamento de filas, controle e prevenção de congestionamento e eficiência de enlaces.

Tratando-se da defesa da construção de projetos de rede baseadas em políticas, classificar e agregar os fluxos de dados constitui atividade fundamental. Essa etapa define o tratamento dado a cada um dos agregados no domínio *DiffServ*, de acordo com o *PHB* pré-definido. Depois do condicionamento do tráfego, feito pelos roteadores de borda na entrada do domínio, os próximos roteadores darão tratamento diferenciado à classe, sem que seja necessária nova análise detalhada dos pacotes, mas apenas identificando a classe de cada um, a partir da marcação feita na borda, com *DSCP*, e fazendo a associação a uma estratégia de gerenciamento associada à classe. Essa estratégia define, basicamente, o tipo e as configurações do mecanismo de fila e dos mecanismos de controle de tráfego (policiamento, suavização e descartes) associados à classe.

As dificuldades existentes para gerenciamento de redes *DiffServ* também foram estudadas. O gerenciamento de redes baseadas em políticas conta com diversos grupos de trabalho específicos, os quais propõem uma solução integrada que traduza a definição de políticas abstratas, de alto nível de complexidade e próximas da linguagem humana, em políticas de baixo nível, configuradas nos equipamentos.

Por fim, tentou-se demonstrar, através de um estudo de caso real, no qual todos ou boa parte dos mecanismos da *DiffServ* foram utilizados, a implementação da abordagem de projeto apresentada. Esses casos tiveram resultados satisfatórios, o que comprovou a eficácia da proposta de referência para projetos de *QoS* em redes corporativas. Para as classes de tempo real, composta pelas aplicações *VoIP*, resultados de qualidade foram obtidos através das medições de parâmetros de *QoS* relevantes a esse tipo de tráfego, notadamente, a taxa de serviço (banda), atraso e *jitter*, e de pesquisas de satisfação por amostragem com usuários, garantindo a manutenção da qualidade

do serviço de telefonia após a implantação da solução. Para as classes de dados críticos foram utilizadas ferramentas de supervisão do tempo de resposta das transações, garantindo que a configuração das políticas estava acordante aos requisitos dessas aplicações.

Como conseqüência e continuidade desse estudo, alguns trabalhos surgem como possibilidades:

- Construção de um modelo de gerência de classes com modificação dinâmica de parâmetros;
- Desenvolvimento de ferramenta que implemente e integre a caracterização da rede, com fornecimento de perfil de tráfego, e o modelo de *QoS* recomendado no trabalho atual. Caberá ao administrador fornecer as informações acerca da criticidade contextual das aplicações identificadas.

A pretensão é que essas pesquisas também sejam baseadas em padrões e voltadas para as redes corporativas.

REFERÊNCIAS

ALLMAN, M.; PAXSON, V. STEVENS, W. **TCP Congestion Control.** IETF RFC-2581, 1999.

ALMQUIST, P. **Type of Service in the Internet Protocol Suite**. IETF RFC-1349. 1992.

ARMITAGE, G. **Quality of Service in IP Networks**. New Riders Publishing. 2000.

BLAKE, S.; BLACK, D.; CARLSON, M.; DAVIES, E.; WANG, Z. ; WEISS, W. **– An Architecture for Differentiated Services**. IETF RFC-2475, 1998 .

BRADEN, R.; CLARK, D.;SHENKER, S. **Integrated Services in the Internet Architecture: An Overview**, IETF RFC-1633, June 1994.

BRADEN, R.; ZHANG, L.; BERSON, S.; HERZOG, S.; JAMIN, S. **Resource Reservation Protocol (RSVP) Version 1 Functional Specification**, IETF RFC- 2205, September 1997.

BORMANN, C. et. al. **RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed**. IETF RFC-3095. Julho 2001.

CASNER, S.; JACOBSON, V. **Compressing IP/UDP/RTP Headers for Low-Speed Serial Links**. IETF RFC-2508. 1999.

CHAN, K.; SELIGSON, J.; COHEN, R.; HERZOG, S.; REICHMER, F.; SMITH, A. YAVATKAR, R ***COPS*** **Usage for Policy Provisioning (COPS-PR)**, IETF RFC-3084, 2001.

COMER, Douglas E. **Interligação em Rede com TCP/IP**, vol I: princípios, protocolos e arquiteturas. 3 ed. Rio de Janeiro : Ed. Campus, 1998. 672 p.

DAVIE, B. et al. **An Expedited Forwading PHB. IETF RFC-3246**. 2002.

DEGERMARK, M; NORDGREN, B; PINK, S. **IP Header Compression**, IETF RFC-2507,1999.

DURHAM, E.; BOYLE, J.; COHEN, R.; HERZOG, S.; SASTRY, A.; RAJAN, R. **The *COPS* (Common Open Policy Service) Protocol Overview**, IETF RFC-2748, 2000

ENGAN, M ; CASNER, S; BORMANN, C. **IP Header Compression Over PPP**, IETF RFC-2509, 1999.

FERGUSON, P.; HUSTON, G. **Quality of Service: Delivering QoS on the Internet and in Corporate Networks** - Wiley Computer Publishing, 1999.

FLOYD, S.; JACOBSON, V. – **Random Early Detection Gateways for Congestion Avoidance**. IEEE/ACM Transactions on Networking, vol 1, no 4, agosto 1993.

GROSSMAN, D. **New Terminology and Clarifications for Diffserv**, IETF RFC-3260, 2002.

HEINANEN, J.; BAKER, F., WEISS, W.; WROCLAWSKI, J. **Assured Forwarding PHB Group**, IETF RFC-2597, 1999.

HEINANEN, J. **A Single Rate Three Color Marker**. IETF RFC-2697. 1999.

HEINANEN, J. **A Two Rate Three Color Marker**. IETF RFC-2697. 1999b.

ITU-T. Recomendação G.114. **One-way Transmission Time.** ITU, maio 2003.

JACOBSON, V. et. al. **An Expedited Forwarding PHB**. IETF RFC-2598. 1999;

JACOBSON, V. **Compressing TCP/IP Headers**. IETF RFC-1144. 1990.

KOSIUR, D., **Understanding Policy-Based Networking**. Wiley, 2001.

LEE, T. K.; ZUKERMAN, M. **Admission control schemes for bursty multimedia traffic. In: IEEE INFOCOM'01, Alaska, USA, apr. 2001.**

MACCABE, J. **Practical Computer Network Analisys and Design**. Morgan Kaufmann, 1998.

MACFADEN, M. **Configuring Networks and Devices With *SNMP***, IETF RFC3512. 2003.

MAHDAVI, J.; PAXSON, V. **Framework for IP Performance Metrics,** IETF RFC 2330, 1999.

MARTINS, J. **Qualidade de Serviço em redes *IP*: princípios básicos, parâmetros e mecanismos**. Setembro, 1999. Disponível em http://www.jsmnet.com . Acesso em 10 de abril de 2005.

MEDINA, A. et al. **A Two-step Statistical Approach for Inferring Network Traffic Demands**. ICIR, Berkeley, CA, USA. 2004.

MOORE, B., et al., **Policy Core Information Model Specification** – IETF RFC3060. 2001.

NEUMAN, Peter G. **Inside Denial-of-Service Attacks**. Communications of the ACM, 2000, vol. 43, nº 4.

NICHOLS, K.; BLAKE, S.; BLACK, D. **Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers**, IETF RFC-2474, 1998.

ODDOM, W.; CAVANAUGH, M. **Cisco DQOS Exam Certification Guide**. Cisco Press. 2004.

OLIVEIRA, J. et al. **Controlo Transaccional em Sistemas de Gestão de Redes por Políticas**. Universidade de Aveiro, Portugal, março 2003.

OPPENHEIMER, Priscilla. **Top-Down Network Design. Cisco Press**. 2004.

PARK, K. ***QoS* in Packet Networks.** Springer Science and Business Media, Inc. 2005.

RAMAKRISHNAN, K.; FLOYD, S.; BLACK, D. **The Addition of Explicit Congestion Notification (ECN) to *IP*.** IETF RFC-3168. Setembro 2001.

REZENDE, Jose Ferreira. **Avaliação do serviço Assegurado para a diferenciação de serviços na Internet**. Rio de Janeiro, 1999. Disponível em http://www.gta.ufrj.br.

RODRIGUES, A.; GATRELL, J.; KARAS, J.; PESCHKE, R. **TCP/*IP* Tutorial and Technical Overview**, IBM, 2001.

SCHULZRINNE, H. et al. **RTP: A Transport Protocol for Real-Time Applications**. IETF RFC-1889. 1996.

SLOMAN, M., **Policy driven management for distributed systems**. Journal of Management Information Systems, 1994. p. 333-360.

STALLINGS, W., **SNMP, SNMPv2, and RMON 1 and 2**. 3rd ed. 1999: Addison Wesley.

SZIGETI, T.; HATTING, C. **End-to-End *QoS* Network Design**. Cisco Press, 2004.

WALDBUSSER, S. et al, **Policy Based Management MIB**. 2003, IETF.

WESTERINEN, A. **Terminology for Policy-Based Management.** IETF RFC3198, 2001.

WROCLAWSKI, J. **Specification of the Controlled-Load Network Element Service.** IETF RFC-2211, 1997.

ANEXO – *SCRIPTS* DE CONFIGURAÇÃO DE *QoS* UTILIZADOS NO ESTUDO DE CASO

Lojas - Política TC_Out

a) Classificação por listas de acesso

i) Classe Tempo Real

```
ip access-list extended voz_acl
   permit tcp any eq 1720 any
   permit tcp any any eq 1720
   permit udp any range 16384 32767 any range 16384 32767
   deny ip any any
```

→ Sinalização

→ RTP

ii) Classe Suporte

```
ip access-list extended sup_acl
  permit udp any any eq snmp precedence 0
  permit udp any eq snmp any precedence 0
  permit udp any any eq snmptrap precedence 0
  permit udp any eq snmptrap any precedence 0
  permit tcp any any eq tacacs
  permit tcp any eq tacacs any
  permit tcp any any eq telnet
  permit tcp any eq telnet any
  permit udp any any eq tftp
  permit udp any eq tftp any
  permit tcp any any eq 22
  permit tcp any eq 22 any
  permit udp any any eq ntp
  permit udp any eq ntp any
  permit udp any any eq 1967
  permit udp any eq 1967 any
!
ip access-list extended rp_acl
  permit tcp any eq bgp any
  permit tcp any any eq bgp
  permit ospf any any
  permit udp any eq rip any eq rip
```

iii) Crítico

```
ip access-list extended critico_acl
  permit tcp host 10.4.200.253 host 10.0.1.100
  permit tcp host 10.4.200.253 host 10.0.1.101
  permit tcp host 10.4.200.253 host 10.0.1.102
  permit tcp host 10.4.200.253 host 10.0.1.200
  permit tcp host 10.4.200.253 host 10.0.1.201
```

→ Servidores

iv) Semi-crítico

```
ip access-list extended semicritico_acl
  permit tcp any host 10.0.2.100 eq 80
  permit tcp any host 10.0.2.101 eq 80
  permit tcp any host 10.0.2.100 eq 443
  permit tcp any host 10.0.2.101 eq 443
  permit tcp host 10.4.200.253 host 10.0.2.200
```

v) Normal

```
ip access-list extended normal_acl
  permit tcp 10.0.0.0 0.0.0.255 any eq 5800
  permit tcp 10.0.0.0 0.0.0.255 eq 5800 any
  permit tcp host 10.4.200.253 host 10.0.3.100
```

vi) Não cooperativo

```
ip access-list extended nc_acl
  permit tcp any host 10.0.5.100
  permit udp any host 10.0.5.100
  permit tcp host 10.0.5.100 any
  permit udp host 10.0.5.100 any
```

vii) *best effort*

Nenhum mecanismo de classificação. O que quer dizer que todos os pacotes não pertencentes às demais classes, pertence à classe *BE*.

b) Criação dos agregados de fluxo (classes de serviço)

i) Tempo Real (*VoIP*)

```
class-map match-any classe_voz
  match access-group name voz_acl
```

ii) Suporte

```
class-map match-any classe_sup
  match access-group name rp_acl
  match access-group name sup_acl
```

iii) Critico

```
class-map match-any classe_critico
  match access-group name critico_acl
```

iv) Semi-Critico

```
class-map match-any classe_semicritico
  match access-group name semicritico_acl
```

v) Normal

```
class-map match-any classe_normal
  match access-group name normal_acl
```

vi) Não cooperativo

```
class-map match-any classe_nc
  match access-group name nc_acl
```

vii) *best effort*

```
class-map match-any classe_nc
  match any
```

c) Criação da política

i) Política TC_Out

```
policy-map TC_Out_128K
!
! Classe Voz
  class classe_voz
    ! reserva banda de 26Kbps para de voz (2 canais)
    priority 26
    ! policia trafego de voz, descartando o excesso
    police 26000 conform-action set-dscp-transmit 46 exceed-action drop
!
! Classe Suporte
  class classe_sup
    ! reserva banda de 8 Kbps
    bandwidth 8
    set ip dscp af33
!
! Classe Critico
```

```
  ! class classe_critico
      ! reserva banda de 16Kbps
      bandwidth 16
      police  24000  conform-action  set-dscp-transmit  af41  exceed-action
  set-dscp-transmit af31
  !
  ! Classe Semi-Critico
    class classe_semicritico
      ! reserva banda de 24Kbps
      bandwidth 24
      set ip dscp af31
      police  24000  conform-action  set-dscp-transmit  af31  exceed-action
  set-dscp-transmit af21
      ! aciona random drop baseado em dscp
      random-detect dscp-based
  !
  ! Classe Normal – Nao interativo
    class classe_semicritico
      ! reserva banda de 24Kbps
      bandwidth 24
      set ip dscp af31
      ! aciona random drop baseado em dscp
      random-detect dscp-based
  !
  ! Classe Nao cooperativo - scavenger
    class classe_nc
      ! reserva banda de 8K – 5% (mínimo permitido para reserva – 8Kbps)
      bandwidth 8
      set ip dscp af11
      ! aciona random drop
      random-detect

    ! classe Best Effort
    class classe_be
      ! banda de 18K – 15% para classe, somente nao aloca valor separado
      ! para L2 overhead
      bandwidth 18
      ! aciona random drop baseado em dscp
      random-detect dscp-based
      random-detect dscp 0 10 30 1
      ! marca trafego best effort com dscp default
      set ip dscp 0
```

Aplicação da política à interface

```
interface Multilink 1
 max-reserved-bandwidth 95
 ip address 200.217.72.6 255.255.255.252
 service-policy output TC_Out_128K
 ip tcp header-compression iphc-format ──► Compressão Header TCP
 ppp multilink
 ppp multilink fragment-delay 10   ┐
 ppp multilink interleave          ├──► Fragmentação e Intercalação (LFI)
 ip rtp header-compression iphc-format ──► cRTP
```

Roteadores da Matriz

1) *Gateways* e *gatekeepers*

Política TC_Voz – somente marca os pacotes de voz com DSCP EF.

```
ip access-list extended voz_acl
   permit tcp any eq 1720 any
   permit tcp any any eq 1720
   permit udp any range 16384 32767 any range 16384 32767
   deny ip any any
!
class-map match-any classe_voz
  match access-group name voz_acl
!
policy-map TC_Voz_GW
!
! Classe Voz
  class classe_voz
    set ip dscp ef
    priority 2048000
!
interface FastEthernet 0/0
  ip address 10.255.0.1 255.255.255.0
  service-policy output TC_Voz_GW
!
```

Sinalização

RTP

2) *Switch* de camada 3

Política TC_Out_Dados_Matriz – classificação e marcação do tráfego

Utiliza as mesmas listas de acesso e mesmas classes dos roteadores das lojas.

```
policy-map TC_Dados_Matriz
!
! Classe Suporte
  class classe_sup
    set ip dscp af33
!
! Classe Critico
! class classe_critico
    set ip dscp af41
!
! Classe Semi-Critico
  class classe_semicritico
    set ip dscp af31
!
! Classe Normal – Nao interativo
! class classe_normal
    set ip dscp af21
!
! Classe Nao cooperativo - scavenger
```

```
  class classe_nc
    set ip dscp af11
   ! classe Best Effort
  class classe_be
    set ip dscp 0

!
interface Vlan100
  description *** VLAN WAN - ROTEADORES ***
  ip address 10.0.100.254 255.255.255.0
  service-policy output TC_Out_Dados_Matriz
!
```

3) Política PHB nos roteadores centrais

a) Classes

i) Tempo Real (*VoIP*)

```
class-map match-any classe_voz
  match ip dscp ef
```

ii) Business

```
class-map match-any classe_bu
  match ip dscp af41
  match ip dscp af33
  match ip dscp af31
  match ip dscp af21
```

iii) Não cooperativo (*scavenger*)

```
class-map match-any classe_nc
   match ip dscp af11
```

iv) *Default*

```
class-map match-any classe_be
   match ip dscp 0
```

b) Criação da política PHB

```
policy-map PHB_Matriz_Out
!
! Classe Voz
  class classe_classe_voz
    priority 2400
    police 2400000 conform-action transmit exceed-action drop
!
```

```
   ! Classe Business
     class classe_bu
       bandwidth  percent 45
       random-detect dscp-based
   !
   ! Classe Default
     class classe_be
       bandwidth  percent 15
       random-detect dscp-based
   !
   ! Classe Não cooperativo
     class classe_nc
       bandwidth  percent 5
       random-detect dscp-based
   !
```

Aplicação da política na interface

```
!
interface Multilink 1
  bandwidth 8192
  max-reserved-bandwidth 95
  ip address 200.217.65.2 255.255.255.252
  service-policy output PHB_Matriz_Out
  ppp multilink
!
```

OBS.: Nesse caso não há necessidade da utilização de mecanismos de eficiência de enlaces (compressão de *header RTP* ou *TCP* / fragmentação e intercalação). Compressão em enlaces de alta velocidade, além de desnecessária, pode se tornar prejudicial, levando os roteadores a ocupar excessivos ciclos de *CPU* com essa atividade. Fragmentação e intercalação não são necessárias para enlaces de velocidade igual ou superior a 2Mbps, pois o tempo de serialização é inferior ao *jitter* máximo tolerado para o tráfego de voz. Por esse motivo, para esses acessos, também não são recomendadas.