UNIVERSIDADE SALVADOR - UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO EM REDES DE COMPUTADORES
MESTRADO EM REDES DE COMPUTADORES

Karine Belens Pessoa

Caracterização da Qualidade de Serviço (QoS) na Implantação de Soluções VoIP

Salvador

2004

Karine Belens Pessoa

Caracterização da QoS na implantação de soluções VoIP

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

CO-ORIENTADOR: Prof. Msc. Sérgio de Figueirêdo Brito

Salvador
2004

TERMO DE APROVAÇÃO

Karine Belens Pessoa

Caracterização da QoS na implantação de soluções VoIP

Esta dissertação foi julgada adequada para a obtenção do título de MESTRE e aprovada em sua forma final pelo Orientador e pelo Curso de Pós-Graduação.

Orientador

Joberto Sérgio Barbosa Martins

Doutor em Informática, *Université Pierre et Marie Curie*, França

Universidade Salvador

Co-orientador

Sérgio de Figueirêdo Brito

Mestre em Engenharia Elétrica, Universidade Federal da Paraíba

Universidade Salvador

Coordenador do Curso

José Augusto Suruagy Monteiro

Doutor em Informática, University of California at Los Angeles, Estados Unidos

Universidade Salvador

Banca Examinadora

Prof. Dr. José Augusto Suruagy Monteiro, UNIFACS

Prof. Dr. Joaquim Celestino Junior, UECE

A Gil

AGRADECIMENTOS

São tantos e tão especiais...

Gostaria de agradecer a Deus e aos espíritos de luz por esta oportunidade de crescimento e por tornar possível a realização de um trabalho que representa um projeto de vida.

Queria agradecer à Coordenação do Mestrado, representada pelo Prof. Suruagy, assim como a todos os meus outros professores por todo o conhecimento compartilhado.

Ao prof. Joberto pela dedicação, paciência e incentivo ao longo de alguns anos de orientação.

Ao prof. Sérgio pelo carinho e auxílio em todos os momentos em que precisei.

A Portnoi e a Rafael pela enorme generosidade e ajuda em muitos momentos de dificuldades.

Aos colegas de Mestrado, em especial a Paulo e Alessandra, pelo companheirismo nas madrugadas de estudo.

A meus pais, Noewal e Ana, pelos exemplos de coragem e força e por despertarem em mim o amor pelo estudo e pela busca do conhecimento.

A Gilmar por despertar em mim o mais nobre dos sentimentos: o amor.

A Kelly pela amizade durante muitos anos.

A meus irmãos, Rogério e Milena, pela oportunidade de convivência e a meus cunhados, Ana e Lucio, pela amizade de todas as horas.

A Iamara pela amizade generosa de tantos anos e a todas as outras pessoas amigas pelos momentos felizes que me proporcionaram.

Enfim, queria agradecer à vida....

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

CELP	*Low Delay Code Excited Linear Prediction*
CIR	*Commited Information Rate*
CRC	*Cyclic Redundancy Check*
CSTA	*Call Siqualling Transport Addresses*
DARPA	*Defense Advanced Research Projects Agency*
DOS	*Denial Of Service*
DS	*Differentiated Service*
DSP	*Digital Signal Processor*
DSCP	*Differentiated Service Code Point*
DTMF	*Dual Tone Multiple Frequency*
FEC	*Forward Error Correction*
FIFO	*First In First Out*
FTP	*File Transfer Protocol*
GOS	*Grade of Service*
IP	*Internet Protocol*
ITG	*Internet Telephony Gateway*
LAN	*Local Area Network*
LDP	*Label Distribution Protocol*
LSP	*Label Switched Path*
LSR	*Label Switching Router*
MG	*Media Gateway*
MGCP	*Media Gateway Control Protocol*
MOS	*Mean Opinion Score*
MPLS	*MultiProtocol Label Switching*
NS	*Network Simulator*
NSF	*National Science Foundation*
PSTN	*Public Switched Telephone Network*
PVC	*Permanent Virtual Circuit*
QOS	*Quality of Service*

RAS	*Registration, Admission and Status*
RED	*Random Early Detection*
RIO	*Red In Out*
RFC	*Request for Comments*
RSVP	*Resource Reservation Protocol*
RTP	*Real Time Transfer Protocol*
RTT	*Round Trip Time*
SCPT	*Stream Control Transmission Protocol*
SDP	*Session Description Protocol*
SGCP	*Simple Gateway Control Protocol*
SLA	*Service Level Agreement*
SIP	*Session Initiation Protocol*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
TSAP	*Transport Service Access Point*
OTCL	*Object Tool Command Language*
UDP	*User Datagram Protocol*
VOIP	*Voice over IP*
VPN	*Virtual Private Network*
WAN	*Wide Area Network*
WRR	*Weighted Round Robin*

RESUMO

Existem grandes motivações para trafegar voz em redes IP (Minoli, 1998), entretanto este protocolo possui algumas limitações. Toda a sua estrutura visa a simplicidade e , como conseqüência, possui limitações que impedem que mídias sensíveis ao atraso, como por exemplo a voz, possam ser transportadas com qualidade. Para tanto, torna-se necessário a implantação de estratégias e mecanismos de QoS que garantam a qualidade de uma aplicação VoIP.

Este trabalho foi desenvolvido com o objetivo de analisar o desempenho de uma aplicação VoIP e sugerir parâmetros de tráfego que permitam que os requisitos de QoS possam ser atendidos. Além disto, foram analisadas as características de aplicações VoIP, assim como as estratégias, mecanismos e critérios de QoS necessários à execução eficiente das mesmas.

Como ferramenta de simulação, foi utilizado o pacote NS-2 que viabilizou a realização de simulações, cujos resultados serviram de apoio para a elaboração das conclusões.

Os resultados mostrados referem-se a valores de parâmetros de configuração que garantem que os parâmetros de QoS possam ser alcançados e as conclusões evidenciam o relacionamento existente entre os parâmetros de configuração e os de QoS.

Os resultados indicam o relacionamento entre os parâmetros de configuração da estratégia *Diffserv* e os parâmetros de QoS como percentual de perdas, atraso e *jitter*. Além disso, são fornecidas informações sobre como configurar os parâmetros, de forma a obter a qualidade exigida por uma aplicação VoIP.

ABSTRACT

There are a lot of reasons to send voice over IP, but this protocol has some limitations. It was designed based on simplicity and in consequence has some limitations that prejudice quality among certain kind of time delay sensitive transmissions, such as voice. In solving such problems it becomes necessary to implement QoS strategies and mechanisms that could guarantee the quality needed by VoIP applications.

This work was developed aiming at measuring of the performance of VoIP applications and suggesting values traffic parameters that fulfill QoS requirements. Besides, the characteristics of VoIP applications where analyzed, as weel as QoS strategies, mechanisms and criterias bearing in mind efficient ways of executing them all.

The NS-2 package was used as a simulation tool which results were used to draw conclusions. The results here in are related to traffic parameters that guarantee that the QoS requirements could be satisfied and the conclusions emphasize close the relation of the simulation and QoS parameters.

1 INTRODUÇÃO

1.1 CONTEXTO

A telefonia é uma tecnologia antiga e eficiente. É também uma forma de comunicação que atinge grande parte da humanidade. A rede telefônica, desta forma, possui uma base instalada muito estável e confiável. Entretanto, é inegável que as redes de comutação de pacotes, inicialmente projetadas para a transmissão de dados, vêm sofrendo um grande aumento no número de usuários nos dias atuais. Neste contexto, fica evidenciada a importância da Internet, cuja tecnologia se baseia no protocolo IP (*Internet Protocol*).

TCP/IP (Comer, 1998) é uma tecnologia de rede simples popularizada entre computadores com o sistema operacional Unix e a Internet. Atualmente está presente na maioria dos sistemas operacionais de rede e é utilizado por muitas empresas em uma variedade de aplicações.

O cenário atual é caracterizado por um número, cada vez maior, de computadores utilizando o TCP/IP para suas comunicações em redes privadas e na Internet.

Com o desenvolvimento da micro-eletrônica, com circuitos mais rápidos e o aumento da capacidade de processamento das máquinas, além de enlaces de alta velocidade, cada vez mais se pensa em utilizar a estrutura de uma rede de computadores para transmitir mídias contínuas, como voz e vídeo.

Um dos valores mais importantes da Telefonia na LAN é que ela permite economia no custo de chamadas a longas distâncias, transmissões de dados e vídeo-conferência. Outro valor importante é o aumento da produtividade que ela oferece devido à integração de aplicações de voz e dados.

A estrutura básica da tecnologia VoIP é relativamente barata, uma vez que ela utiliza boa parte da estrutura da rede de dados para trafegar voz. Além disto, existe a possibilidade de economia para as redes telefônicas particulares de corporações de médio a grande porte, já que pode-se evitar os custos de ligações interurbanas e internacionais. O máximo que pode ocorrer é a necessidade de ligação com um provedor de acesso, que geralmente corresponde a uma tarifa local.

Existe uma grande base de máquinas IP instaladas e isso é outro fator motivador da solução VoIP, uma vez que facilita a difusão desta tecnologia como tecnologia de suporte para as diversas redes.

Voz sobre IP é um conceito relativamente simples, basta transformar a voz em sinal digital e então empacotar os dados em uma aplicação IP dentro de uma rede de computadores que utilize o IP como protocolo de nível de rede. Esta simplicidade é que permite transmitir dados e voz dentro de uma mesma rede, fazendo com que aplicações de voz em tempo real sobre o protocolo IP tornem-se uma realidade em rede privadas.

A figura 1 mostra o esquema de encapsulamento da voz em pacotes IP.

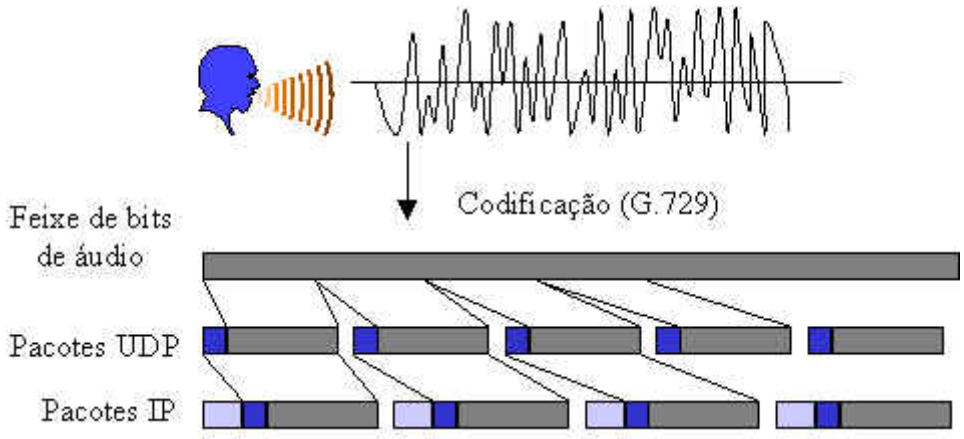

Figura 1. Encapusulamento de voz em pacotes IP (Martins, 1999)

A tecnologia VoIP permite a verdadeira integração de dados e voz e provê a utilização mais fácil, flexível e com uma boa relação custo x benefício. Várias novas aplicações podem ser utilizadas, tais como, telefonia e fax sobre IP, videoconferência, comércio eletrônico, vídeo sobre IP, educação a distancia, aplicações colaborativas, aplicações multimídia, além de distribuição de chamadas automática, resposta de voz interativa, etc. Isso é um contraste frente ao paradigma de integração de voz/dados, telefonia computadorizada, na qual o tráfego de voz é mantido separado do tráfego de dados, onde os serviços de telefonia são melhorados através do suporte do computador.

Apesar de toda a simplicidade, redes IP apresentam problemas graves quando se trata de transmitir mídias sensíveis ao atraso como a voz. O protocolo IP *best effort* (Comer, 1998) presta um serviço sem conexão, o que implica na inexistência de garantias referentes a tempo para os pacotes transmitidos. Não há garantias, por exemplo, de um atraso constante, o que pode tornar uma aplicação de voz em tempo real, como por exemplo uma conversa telefônica, um serviço de baixa qualidade com a voz entrecortada e muitas vezes ininteligível.

Visando tratar de questões relativas à adequação da tecnologia de redes para transmissão de tráfego sensível a determinados parâmetros, surgiu o estudo da Qualidade de Serviço (QoS) em redes (Huston e Ferguson, 1998). A obtenção de qualidade de serviço adequada é então, um requisito de operação de rede e suas componentes, para viabilizar a operação com qualidade de uma aplicação VoIP.

Este trabalho apresenta os aspectos necessários para a obtenção de qualidade de serviço para aplicações VoIP executadas em uma rede *Diffserv* (Blake *et al.*,1998). Para isto, faz uma

descrição das aplicações VoIP, definindo critérios de QoS para as mesmas. Em seguida, propõe um modelo de simulação para, através da ferramenta de simulação NS, obter um conjunto de resultados e para finalmente, a partir da análise dos resultados, sugerir especificações de parâmetros que possam ser aplicados de forma prática em redes corporativas, com o objetivo de contribuir para o aumento da qualidade das mesmas.

1.2 OBJETIVOS

O propósito desta dissertação é definir parâmetros e mecanismos de QoS que possam garantir a qualidade do uso de uma aplicação VoIP que rode em uma rede IP *Diffserv*.

Para tanto, objetiva-se caracterizar uma aplicação VoIP, assim como os critérios necessários à sua execução com qualidade. Além disso, busca-se definir parâmetros de tráfego que podem ser configurados de forma adequada para garantir o atendimento dos critérios de Qualidade de Serviço (QoS) exigidos pela aplicação em questão.

1.3 ESTRUTURA DA DISSERTAÇÃO

O segundo capítulo apresenta as principais características de aplicações VoIP, dando ênfase aos equipamentos e protocolos típicos.

O terceiro capítulo discorrerá sobre os critérios básicos que definem QoS e os requisitos de QoS que redes de pacotes devem ter para garantir uma boa qualidade na transmissão de voz.

O quarto capítulo descreverá a estratégia *Diffserv* e a forma como esta estratégia pode ser usada para obtenção de qualidade de serviço em aplicações VoIP .

O quinto capítulo descreverá a ferramenta NS, abordando suas características, recursos, limitações e o modelo de simulação utilizado nas simulações. O modelo foi criado para uma rede IP *Diffserv* que executa uma aplicação VoIP.

O sexto capítulo apresentará uma descrição de simulações realizadas, desde a definição dos parâmetros até a coleta dos resultados. Será apresentada uma análise dos resultados coletados, definindo requisitos de rede necessários para garantir níveis aceitáveis de determinados parâmetros de QoS, como por exemplo atraso, *jitter*, e taxa de perdas, com o objetivo de garantir um bom desempenho das aplicações VoIP.

Na conclusão, são feitas recomendações sobre a parametrização de uma rede *Diffserv* que executa uma aplicação VoIP, focando o gerenciamento de QoS como forma de possibilitar o uso em larga escala da integração de voz e dados em redes IP.

2 CARACTERIZAÇÃO DA APLICAÇÃO VOIP

2.1 INTRODUÇÃO

A especificação TCP/IP foi desenvolvida tendo, como uma de suas premissas básicas, o requisito de poder ser utilizada com os diversos tipos de meios físicos e tecnologias existentes na época de sua criação ("IP sobre Tudo" - Anos 70).

O cenário atual das redes IP mudou. Hoje, o cenário de utilização das redes IP deseja que "qualquer aplicação" possa ser executada com qualidade sobre o IP. Ou seja, a situação do IP atualmente é no sentido do "Tudo sobre IP" mantendo a premissa básica de projeto do "IP sobre Tudo" dos anos 70.

Atualmente, o protocolo IP tem uma base instalada muito grande e a tendência é que ele suporte s novas aplicações em rede, como por exemplo:

- Telefonia e Fax sobre IP (VoIP - *Voice over IP*);
- Comércio Eletrônico (*E_commerce*);
- Vídeo sobre IP;
- Educação a Distância (EAD) (*Distance Learning*);
- Vídeo-Conferência;
- Aplicações Colaborativas e de Grupo;
- Aplicações Multimídia;
- Aplicações Tempo Real;

- Outras.

Genericamente, a maioria das aplicações citadas podem ser caracterizadas como multimídia, uma vez que, envolvem a transferência de múltiplos tipos de mídia (dados, voz, vídeo, gráficos, etc) com requisitos específicos para a sua operação com qualidade.

O protocolo IP foi projetado inicialmente para ser um protocolo simples, visando atender o cenário imaginado na época para sua implantação em termos de rede. Este paradigma de concepção, denominado melhor esforço (*best effort*), impõe algumas restrições técnicas ao IP e, por conseqüência, restringe as aplicações suportadas às aplicações com poucos requisitos de operação, como por exemplo as aplicações de dados que podem perder pacotes e não são sensíveis ao atraso.

As redes baseadas no IP *best effort* não oferecem garantia de qualidade de serviço às aplicações que utilizam seus recursos, logo torna-se necessária a identificação das eventuais limitações do IP e procedimentos necessários para adequá-lo à nova realidade das redes.

Objetivando o atendimento de pré-requisitos que garantam a qualidade exigida por algumas aplicações, como por exemplo, as aplicações VoIP. O IETF definiu alternativas técnicas como o IP *Intserv* (White, 1997) e o IP *Diffserv*. Estas alternativas superaram as deficiências do protocolo IP *best effort* e viabilizaram o transporte de mídias como vídeo e áudio em redes IP.

Um grande número de elementos estão envolvidos na elaboração de uma chamada VoIP com qualidade de serviço aceitável. Estes elementos incluem o codec de voz, empacotamento, perda do pacote, atraso, variação do atraso e a arquitetura da rede que visa fornecer QoS. Outros fatores envolvidos em uma bem sucedida chamada VoIP incluem o protocolo de sinalização do estabelecimento da chamada e questões de segurança (Goode, 2002).

Muitos fatores determinam a qualidade da voz, incluindo a escolha do *codec*, controle de eco, perda do pacote, atraso, variação do atraso (*jitter*) e o projeto da rede. A perda do pacote provoca falhas na voz. Alguns algoritmos de *codec* podem corrigir algumas perdas de pacotes de voz. Caracteristicamente, apenas um único pacote pode ser perdido durante um período curto para que a correção do *codec* seja eficaz (Goode, 2002). Se o atraso fim-a-fim se tornar muito longo, a conversa começa a perder a qualidade. Um buffer no aparelho receptor sempre equilibra o *jitter*. Se a variação do atraso excede o tamanho do buffer haverá uma perda de pacotes em qualquer ponto da linha de transmissão prejudicando a qualidade da mensagem recebida.

Uma aplicação VoIP pode codificar a mídia de voz com padrões H.323 como o ITU-T G.711 ou G.723.1 e envia os pacotes através do protocolo RTP encapsulado dentro do UDP. Como a mídia de áudio é sensível a atrasos, ela deve ter a preferência na alocação de banda de rede na transmissão e utiliza-se c*odec*s de baixa taxa de bits para reduzir a necessidade de banda.

Quanto aos requisitos de retardo fim-a-fim para o áudio, este deve ser mínimo, não devendo ultrapassar 200 ms, a fim de não permitir uma comunicação de áudio sem sincronismo entre os participantes. A rede também deverá garantir um *jitter* menor ou igual a 40 ms, a fim de que a utilização de áreas de armazenamento para compensá-lo deva ser mínima para não adicionar componentes de retardo, ao atrasar o ponto de reprodução de pacotes. O percentual de perdas não deve ultrapassar 10 %.

As estratégias, mecanismos e parâmetros de QoS relacionados a uma aplicação VoIP serão discutidos no capítulo 3.

2.2 DESAFIOS DO USO DAS REDES IP PARA TRANSMISSÃO DE VOZ

Vários são os motivos que causam os problemas relacionados com a transmissão de voz, principalmente, em redes IP *best effort*. Abaixo estão alguns exemplos:

- O primeiro desafio é de caráter técnico e diz respeito ao paradigma inicialmente previsto para o protocolo que enfatizava a simplicidade de concepção. Por exemplo, o IP *best effort* não tem nenhuma garantia de vazão constante para uma aplicação em particular. Além disso, uma aplicação não pode obter do IP nenhuma garantia de entrega da próprios pacotes que eventualmente são descartados ou perdidos sem que nenhum tipo de correção ou ação seja tomada. Não existe igualmente nenhuma garantia de tempo de entrega para os pacotes;

- O Protocolo IP *best effort* oferece um serviço do tipo datagrama (sem conexão), e portanto não aloca um caminho fixo para um pacote de dados trafegar na rede, o que não garante requisitos temporais para os pacotes transmitidos. Uma grande diferença entre uma aplicação de dados e um aplicação de voz é que uma aplicação de voz é sensível ao atraso. Em uma rede IP não é possível garantir um atraso constante o que pode tornar uma aplicação de voz em tempo real, como por exemplo uma ligação telefônica, um serviço de baixa qualidade com a voz entrecortada e muitas vezes ininteligível;

- Problemas de desempenho na rede, pois o TCP/IP não foi projetado para suportar aplicações de tempo real, não tendo como oferecer qualidade de serviço. Se a rede base para o transporte desta aplicação for a Internet, certamente não deve ser utilizada para fins profissionais, pois o TCP/IP não oferece garantias de QoS. Sem a especificação da qualidade de serviço, a qualidade da voz fica dependente do tráfego de dados existente no momento da conversa, o que pode comprometer seriamente o resultado almejado;

- Outra dificuldade que aparece na transmissão de mídias que exigem sincronismo, como a voz, é que o serviço prestado por grande parte das redes IP é do tipo "melhor esforço" (*best effort*). Assim, todos os pacotes são tratados de forma igual, sem nenhuma discriminação entre os diversos tipos de tráfegos, nem atribuição de prioridades aos pacotes. O escalonamento nas filas dos roteadores é feito baseado em filas FIFO, onde, se houver espaço nos *buffers* dos roteadores, o pacote é armazenado para transmissão e, caso contrário, ele é descartado;

- Outro fator crítico para a transmissão de voz é o atraso. Este parâmetro pode ser agrupado em dois tipos: fixo e variável. A variação de atraso também recebe a denominação de *jitter*. Os dois tipos possuem naturezas e causas diferentes. Os atrasos grandes causam desconforto na conversação e os variáveis atrapalham a sincronização na transmissão da voz.

A forma de abordar as "deficiências" do IP consiste então em propor novos protocolos, algoritmos e mecanismos que garantam a transmissão dos pacotes de voz com um bom nível de qualidade de Serviço, suprindo as deficiências tecnológicas intrínsecas ao protocolo e viabilizando o suporte efetivo de qualquer tipo de aplicação sobre redes IP.

Nos próximos capítulos, serão abordados os algoritmos e mecanismos de QoS que viabilizarão a minimização dos problemas citados anteriormente.

2.3 FUNCIONAMENTO DAS APLICAÇÕES VOIP

Embora o VoIP envolva a transmissão de voz digitalizada em pacotes, o telefone pode ser tanto analógico quanto digital. A voz pode ser digitalizada e codificada tanto antes quanto durante o empacotamento. A solução VoIP necessita de equipamentos especiais, os *gatekeepers* e os

gateways VoIP (Keagy, 2000), responsáveis respectivamente pela implementação das funções de PABX e roteamento de pacotes. As características, recursos e funcionalidade destes equipamentos serão discutidos no tópico 2.5.

Para explicar o funcionamento de uma aplicação VoIP, serão usados dois exemplos que ilustram possíveis soluções VoIP.

A figura 2 (Goode, 2002) mostra um esquema no qual o PABX está conectado a um *gateway* VoIP e também a uma central telefônica de uma companhia local. O *gateway* VoIP permite que as chamadas telefônicas sejam completadas através da companhia telefônica como no passado. O esquema pode usar a rede para fazer todas as chamadas entre os sites conectados e o *gateway* VoIP ou optar por dividir o tráfego entre a rede IP e a PSTN buscado em uma rota de menos custo usando algoritmos configurados no PABX. As chamadas VoIP não estão restritas aos telefones servidos diretamente pela rede IP. Refere-se às chamadas VoIP para telefones servidos pelo PSTN como chamados “fora da rede”. As chamadas fora da rede podem ser roteadas através da rede IP para um *gateway* VoIP próximo ao telefone de destino.

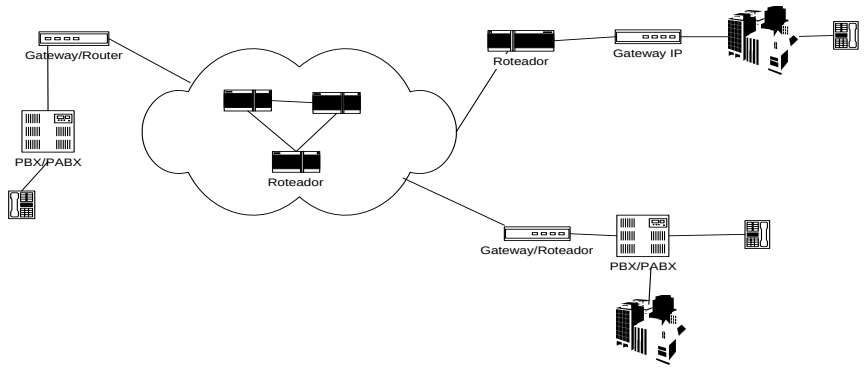

Figura 2. Esquema VoIP

Uma implementação alternativa do VoIP usa telefones IP e não confia em um PABX padrão. A figura 3 (Goode, 2002) mostra um diagrama simplificado de um sistema telefônico IP conectado com uma rede IP de longa distância. Os telefones IP são conectados a uma LAN. As chamadas de voz podem ser feitas localmente através da LAN. Os telefones IP incluem *codecs* que digitalizam e codificam (e também decodificam) a voz. Os telefones IP também empacotam e desempacotam a voz codificada. Chamadas entre locais diferentes podem ser feitas através da rede IP de longa distância. Os servidores Proxy realizam o registro do telefone IP e coordenam a sinalização de chamada, principalmente entre os lugares. As conexões com PSTN podem ser feitas através do *gateway* VoIP.

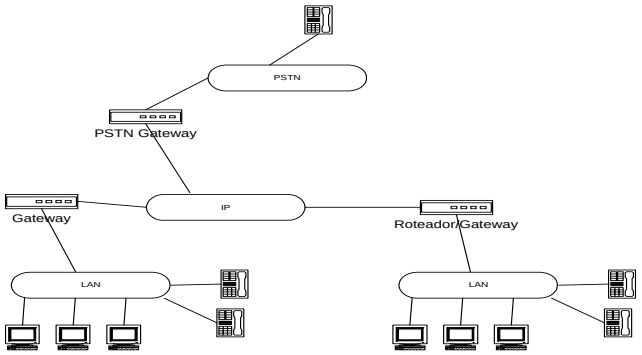

Figura 3. Diagrama de um sistema telefônico IP conectado a uma rede IP

2.4 PROTOCOLOS DE SINALIZAÇÃO

Outro elemento envolvido em uma chamada VoIP é o protocolo de sinalização.

O protocolo de sinalização é utilizado nas aplicações VoIP para configurar as chamadas através da informação dos parâmetros necessários à manutenção da qualidade de serviço necessária.

Há muitos protocolos de sinalização envolvidos em uma chamada VoIP. Este trabalho falará sobre o H.323, o SIP (*Session Initiation Protocol), o* MGCP (*Media Gateway Control Protocol*) e sobre o Megaco/H.248.

O H.323 e o SIP são protocolos de controle de chamada *peer-to-peer*, enquanto o MGCP e o MEGACO são protocolos de controle de chamadas *master-slave* O MGCP é baseado no modelo de telefonia PSTN. O H. 323 e o MEGACO são projetados para acomodar vídeo conferência e também telefonia básica, porém eles ainda são baseados em um paradigma de orientação a conexão, apesar dos seus usos para sistemas de comunicação de pacotes. Os *gateways* que usam MGCP supõem que a maior parte da inteligência reside em um controlador de *gateway* separado. O SIP abriga terminais inteligentes relacionados não apenas a sessões de voz, mas também a outras aplicações.

2.4.1 O PROTOCOLO H.323

O conjunto de protocolos H. 323 da recomendação ITU-T evoluiu da vídeo telefonia padrão. Em 1996, quando os primeiros projetos de telefonia IP começaram a ser patenteados, foi desenvolvido um padrão de controle de chamada VoIP com o objetivo de permitir que usuários e fornecedores de serviços tivessem opção de produtos interoperantes.

O Grupo de Atividade em *Voic Over IP*, do Consórcio de Telecomunicações Multimídia Internacional (IMTC), recomendou o H.323, o qual havia sido desenvolvido para a comunicação multimídia através de redes de dados em pacotes. Estas redes de pacotes podiam incluir LANs ou WANs. O IMTC tinha a visão de que o VoIP era um caso especial de vídeo sobre IP. Embora nem todos os pioneiros do VoIP concordassem que o vídeo sobre IP rapidamente se tornaria popular, o conjunto de protocolo H.323 tornou-se o primeiro padrão para implementações em

VoIP. As versões 2 – 4 incluem modificações para tornar o H.323 mais aceitável para as necessidades do VoIP (Goode, 2002).

As entidades H.323 podem ser integradas em computadores pessoais ou roteadores ou implementados em dispositivos em posição isolada. Para o VoIP, as entidades H.323 importantes são terminais, *gateways* e *gatekeepers*. Um *gateway* H.323 fornece tradução de protocolo e transcodificação de mídia entre um terminal H.323 e um terminal não H.323. Por exemplo, um *gateway* VoIP fornece a tradução da transmissão, formata e sinaliza procedimentos entre uma rede telefônica de comutação de circuitos e uma rede de pacote. Além disso, o *gateway* VoIP pode fazer a transcodificação e compressão de uma fala e, freqüentemente, é capaz de gerar e detectar sinais DTMF.

Os elementos de um terminal VoIP H.323 incluem o seguinte:

- Uma Unidade de Controle de Sistemas que fornece uma sinalização para operação adequada do terminal H.323 que fornece um controle de chamada usando o H.225 e o H.245;
- H.225 formata o áudio transmitido e controla os fluxos de mensagens, restabelece os fluxos de áudio das mensagens que tenham sido recebidas das interface na rede e faz a numeração da seqüência, a detecção e correção de erros quando apropriado;
- Um controle de áudio transcodifica a fala;
- O canal de controle H.245 carrega fim-a-fim as mensagens de controle, administrando as operações das entidades H.323 (o *host* H.323, o *gateway* H. 323 ou o *gatekeeper* H. 323);
- A função chave do Canal de Controle H.245 é a troca de habilidades. Outras funções do H.245 incluem a abertura e o fechamento de canais lógicos e mensagens de controle de

fluxo. O terminal estabelece um Canal de Controle H.245 para cada chamada na qual, os terminais participam. Este Canal de Controle Lógico H.323 permanece aberto durante toda a duração da chamada.

Como exemplo de como o H.245 é usado, será discutido como ele abriga a sinalização de telefonia simples.

Os curtos tons DTMF transmitidos por *codecs* de baixa taxa de bit (e.x. G.729 e G.723.1) podem ser distorcidos a um ponto que o usuário pode ter problemas no acesso de sistemas automatizados baseados em DTMF, tais como *Voicemail*, menu baseados em sistemas ACD e sistemas bancários automatizados. O H.323v2 oferece uma solução para envio de tons DTMS fora da banda ao invés de utilizá-los da mesma maneira que a fala. Ele é chamado de *relay* DTMF. Se um *relay* DTMF estiver disponível, um *gateway* H.323 detecta os sinais DTMF, cancela o fluxo de voz DTMF antes que ele seja enviado através do RTP, e envia uma indicação de *User Input* H.245 fornecendo o valor do dígito DTMF (0-9, a-d, * ou #) e uma avaliação da duração do tom para um terminal remoto. O *gateway* só enviará os sinais DTMF usando um H.245 se o procedimento de troca de habilidade do H.245 resultar no conhecimento de ponto distante capaz de receber canais DTMF na indicação do *input* do usuário. O padrão H. 245 especifica duas indicações para transportar DTMF na indicação de input do usuário: a indicação alfanumérica e a indicação de sinal. O H.323v2 adiciona apoio para estes dois métodos. A indicação de sinal inclui a duração do dígito e informação opcional tal como a *time stamp*, que pode ser usada por um receptor para sincronizar o sinal DTMF com o fluxo RTP.

A sinalização de chamada pode envolver muitas mensagens passando de um lado para outro entre as entidades H. 323. Para reduzir o tempo de setup da chamada para chamadas diretas tais como

VoIP, o H.323v2 apresenta um procedimento de estabelecimento de chamada alternativa chamado “Conexão Rápida” ou procedimento H. 245 para estabelecer canais em uma chamada.

A conexão rápida encurta o tempo de setup de chamada ponto-a-ponto, reduzindo o número de mensagens trocadas. Após uma troca de mensagem, os terminais podem iniciar uma conversa. Isto é conseguido pela inclusão de um elemento *faststart* na mensagem de setup. O elemento *faststart* descreve uma seqüência em ordem de preferência dos canais de mídia que o terminal propôs usar, incluindo todos os parâmetros necessários para abrir e começar a transferir mídia dos canais imediatamente. O terminal chamado pode concordar em usar o procedimento de Conexão Rápida, enviando uma mensagem Q.931, contendo um elemento de *faststart* selecionado dentre os canais lógicos abertos oferecidos pelo terminal. Uma vez aceitos, os canais são considerados abertos como se os procedimentos H.245 usuais tivessem sido seguidos.

O ponto fim chamado pode começar a transmitir mídia imediatamente após enviar uma mensagem Q. 931 com a aceitação da chamada rápida e o ponto fim que chamou pode começar a transmitir mídia tão logo receba esta mensagem.

O padrão H.235 oferece itens de segurança incluindo autenticação, integridade, privacidade e não rejeição. A função de autenticação certifica que os terminais participantes da conferência são realmente quem eles dizem que são. A função de integridade fornece um meio de garantir que os dados dentro de um pacote sejam de fato uma representação fiel dos dados originais. A privacidade é fornecida por mecanismos de criptografia e decriptografia que escondem os dados dos usuários, de maneira que se eles forem interceptados não poderão ser escutados. A não rejeição é um meio de proteção contra alguém que negue falsamente que eles tenham participado

de uma conferência. O H.323v2 especifica funções para cada uma destas características de segurança. O H. 235 especifica o uso adequado destas funções.

O canal RAS, usado para sinalização do *gateway* para o *gatekeeper*, não é um canal seguro. Para se certificar de uma comunicação segura, o H.235 permite que os *gateways* incluam chave de autenticação nas suas mensagens RAS. O *gatekeeper* pode usar esta chave de autenticação para autenticar a fonte das mensagens. Alguns equipamentos VoIP trazem esta característica em resposta às exigências dos provedores de serviços.

2.4.2 O PROTOCOLO SIP

SIP é um protocolo de sinalização que pode iniciar e encerrar qualquer tipo de sessão. O controle de chamada SIP usa um SDP para descrever os detalhes da chamada (ex. áudio, vídeo, tipo de *codec*, tamanho dos pacotes, etc.).

O SIP usa um URI para identificar um destino lógico, e não um endereço IP. O destino pode ser um *nickname*, um endereço de e-mail ou um número de telefone. Além de realizar uma chamada telefônica, o SIP pode avisar aos usuários de eventos tais como, “estou on-line”, “uma pessoa entrou na sala” ou “chegou um e-mail”. O SIP pode também ser usado para enviar mensagens de texto instantâneas.

Uma URI é um identificador para um recurso que gera respostas diferentes em horas diferentes, dependendo da entrada. Uma URI não depende da localização do recurso. Uma URI geralmente consiste de três partes: O protocolo para comunicação com o servidor (e.x. SIP), o nome do servidor (e.x.: www.xxx.com) e o nome do recurso. Uma URL é uma forma comum de URI.

O SIP permite uma fácil inclusão de novos serviços por terceiros.

O SIP pode ser usado para criar novos serviços, além de reproduzir serviços telefônicos tradicionais. Mensagem instantânea é um exemplo de um tipo de serviço que pode usar o SIP. Existem alguns sistemas de mensagens instantâneas que permitem que usuários criem listas de amigos. Mensagens podem mostrar que alguém está falando ao telefone ou foi a um encontro importante ou saiu para almoço ou está disponível para conversar. Os membros desta lista de amigos podem usar esta mensagem para escolher um momento adequado para fazer uma chamada telefônica, evitando assim fazê-lo em um momento inoportuno. Alguns provedores de software de mensagem instantânea converteram os seus sistema para SIP.

Usando um modelo cliente-servidor, o SIP define as entidades lógicas que podem ser implementadas separadamente ou juntas no mesmo produto. Os clientes enviam pedidos SIP, considerando que os servidores aceitem pedidos SIP, eles executam os métodos e respondem.

A especificações SIP definem seus métodos de pedidos:

- REGISTER permite tanto ao usuário como a uma terceira parte registrar informação de contato com um servidor SIP;
- INVITE inicia a seqüência de sinalização de chamada;
- ACK e CANCEL auxilia a sessão de setup;
- BYE termina uma sessão;
- OPTIONS questiona um servidor sobre sua capacidade.

O protocolo SIP é estruturado em quatro camadas e tem seis categorias de respostas. Algumas das entidades funcionais SIP importantes estão relacionadas abaixo:

- agente usuário desempenha tanto a função de cliente, o qual inicia pedido SIP, como de um servidor, o qual contacta o usuário quando um pedido SIP é recebido e dá uma resposta;
- um proxy SIP age tanto como um cliente SIP como um servidor SIP, fazendo os pedidos SIP de interesse de outros clientes SIP. Um servidor proxy SIP pode ser tanto *statefull* quanto *stateless*. Um servidor proxy tem que ser *statefull* para fornecer TCP, ou para fornecer uma gama de serviços. Contudo, um servidor proxy *stateless* adapta-se melhor, isto é, fornece maior volume de chamadas;
- *Register* é um servidor SIP que recebe, autentica, e aceita pedidos de registros de cliente SIP. Ele pode estar instalado em um servidor *proxy* SIP;
- o servidor local guarda a informação do usuário em um banco de dados e ajuda a determinar onde (para qual endereço IP) enviar um pedido. Ele também pode estar instalado em um servidor *proxy* SIP;
- o servidor de reendereçamento é *stateless*. Ele responde a um pedido SIP com um endereço, onde o originador do pedido pode contactar a entidade desejada diretamente. Ele não aceita chamadas ou inicia seus próprios pedidos.

O SIP permite aos servidores *proxy* tomar decisões complexas sobre para onde enviar o INVITE. Por exemplo, alguém poderia estar viajando e ter deixado as suas chamadas para serem enviadas para um escritório da empresa em Salvador. Um servidor *proxy* pode enviar um INVITE para alguns locais ao mesmo tempo, portanto a chamada poderia ser roteada simultaneamente para o servidor de *Voicemail* da pessoa, em São Paulo ou para o seu escritório em Salvador.

Se a pessoa responder à chamada em Washington, o pedido do INVITE pode conter informação para ser usada pelo servidor proxy de destino para determinar que os aparelhos de destino

toquem.. Pode-se programar um telefone SIP para requerer um serviço “siga-me” apenas para os locais de trabalho. Por outro lado, a pessoa pode programar o seu telefone SIP para enviar as chamadas para o seu celular, mas apenas uma lista privilegiada de acesso teria as chamadas direcionadas para a casa dele.

O SIP facilita a mobilidade porque a mesma pessoa pode usar diferentes terminais com o mesmo endereço e os mesmos serviços. O SIP promete ser usado por muitos programadores para desenvolver novos serviços. Muitos destes novos serviços podem ser oferecidos na Internet. Existe, entretanto, algumas complicações relacionadas ao uso de uma sinalização *peer-to-peer* aberta e o protocolo de controle com o SIP. Uma delas é a segurança.

O SIP em uma rede compartilhada levanta algumas preocupações quanto à segurança que devem ser resolvidas antes de ser largamente adotado. As soluções SIP deparam-se com assuntos sobre preservação da confiabilidade e integridade dos pedidos SIP, prevenção de ataques ou mensagens equivocadas, assegurando a privacidade dos participantes em uma sessão e prevenindo ataques DoS (*Denial of Service*).

As mensagens SIP podem conter informações delicadas dos remetentes, incluindo quem comunica com quem e por quanto tempo e, talvez, os seus endereços de e-mail ou o endereço IP usado nas chamadas. Tanto os indivíduos como as empresas podem desejar que este tipo de informação permaneça privada.

A primeira solução que vem à mente é a criptografia SIP usando porta 5061 ao invés do 5060. A criptografia de todo o pedido ou resposta nos enlaces entre as entidades SIP pode prevenir os farejadores de pacotes e outros intrometidos de descobrir quem está chamando quem. Contudo, os pedidos e as respostas do SIP não podem ser inteiramente criptografados fim-a-fim porque os

campos de mensagem tais como os campos de pedidos URI, rota e via, precisam ser visíveis para os *proxys* para que os pedidos SIP possam ser roteados adequadamente. Além disso, a criptografia SIP é limitada para incluir a carga útil do SDP. As entidades da rede serão incapazes de determinar o *codec* usado, o tamanho do pacote ou a quantidade de largura de banda necessária para o fluxo de RTP. De fato, quando se usa uma criptografia SIP, as entidades podem não ser capazes nem mesmo de determinar se a chamada é apenas de voz ou inclui vídeo. Os pedidos e respostas SIP podem ser protegidos por mecanismos de segurança da rede, como por exemplo, o IPSec. O IPSec é um protocolo de segurança de rede mais adequado para redes privadas virtuais (VPNs).

Para que o SIP funcione com segurança, os servidores *proxy* têm de ser uma parte da relação de confiança na rede. TLS (Dierks e Allen, 1998) é um protocolo da camada de transporte como o TCP ou UDP e qualquer um deles pode ser especificado como protocolo de transporte no campo do cabeçalho VIA ou um SIP-URI. O TLS é adequado para arquiteturas nas quais os *hosts* estão unidos por uma cadeia de confiança (Alguém confia no servidor proxy nice.com, o qual por sua vez confia no servidor proxy global.com no qual o outro confia). Se uma relação de confiança como esta não for estabelecida na rede SIP, existe a possibilidade de que servidores proxys mentirosos possam modificar a sinalização (e.x. adicionando cabeçalhos VIA).

Há algumas maneiras de proteger um *host* contra um ataque DOS. O agente usuário pode desautorizar pedidos que não usem uma associação de segurança persistente estabelecida, usando o TLS ou IPSec para o proxys. Esta solução é também adequada para proxys que tenham confiança mútua.

Domínios administrativos podem usar o TLS e / ou o IPSec para agregar o tráfego através de túneis e canais seguros de e para "*hosts* de fortaleza" os quais são capazes de absorver ataques DOS, assegurando que os SIP, associados a eles no domínio administrativo, não fiquem sobrecarregados com mensagens falsificadas. Esta solução parece adequada para provedores de serviços que carreguem grandes volumes de tráfego SIP.

2.4.3 ARQUITETURAS MASTER/SLAVE

Os protocolos Megaco e MGCP são os protocolos de sinalização de chamadas *master-slave*. Este tipo de arquitetura acredita que a função de processamento da chamada pode ser separada da função *gateway* do VoIP. Pode-se definir uma nova entidade, que seria um "agente de chamada", visando controlar os *gateways* e realizar o processamento da chamada. O produto físico que implementa a função do agente chamador não precisa estar localizado perto do *gateway* e pode controlar muitos *gateways*. Esta arquitetura simplifica o produto *gateway* VoIP, permitindo que o *gateway* seja alocado em casas e pequenos escritórios a um baixo custo.

Considerando como exemplo o diagrama de uma rede de circuitos conectados. Os interruptores podem enviar o tráfego telefônico diretamente de um para o outro, mas comunicam as informações de sinalização de chamada entre eles usando uma rede de sinalização em pacotes SS7 separada. Observe que, embora o protocolo SS7 seja de pacotes conectados, ele não tem relação alguma com o IP.

Alguns projetistas de rede afirmam que a telefonia IP deve substituir a PSTN, mas que as funções essenciais da PSTN continuarão a funcionar através de um longo período de mudança. Isto leva a dois tipos de *gateways*. Os *gateways* de mídia aceitam voz ou tráfego de "mídia" do circuito, conecta e empacota a voz a ser transmitida através da rede IP. Gateways de sinalização conectam

as redes de sinalização (e.x. SS7) e redes IP, de maneira que os agentes chamadores conectados à rede IP possam se comunicar com os interruptores de circuito conectados à rede de sinalização.

O MG permite conexões entre redes diferentes, fornecendo funções de conversão e/ou transcodificação de mídia. Por exemplo, um MG pode receber pacotes de uma rede IP, desempacotá-los, decodificá-los e passar o fluxo de mídia para uma rede de comutação de circuitos. Embora um MG possa fazer adaptação de mídia, em alguns casos, um MG pode atuar como um interruptor ligando duas terminações de recursos do mesmo tipo.

Por esta razão, outras funções que um MG pode realizar incluem uma ponte de conferência com todas as interfaces de pacote, uma entidade de resposta de voz interativa ou um sistema de reconhecimento de voz.

Um MG também fornece função de notificação de evento, administração e alocação de recursos, e também funções de sistema, tais como o estabelecimento e manutenção de uma associação com o Agente Chamador.

Uma função SG consiste em, na extremidade da rede de dados, traduzir sinais de controle de chamada entre a rede de dados empacotados e a rede de telefonia de circuito conectado. Um *gateway* SS7-IP empregaria uma função SG. Por outro lado, o MG pode empregar também uma função SG para processar sinalização de telefonia tradicional associada a um tronco ou terminação de linhas no MG, tal como o canal D de uma linha ISDN BRI ou um tronco PRI.

O agente chamador, o qual é freqüentemente denominado de “controlador do *gateway* de mídia”, tem de se comunicar com o *gateway* de mídia para controlar as suas ações.

Alguns protocolos têm sido desenvolvidos para este tipo de comunicação, incluindo o protocolo SGCP (Arango e Huietema, 1998), protocolo de controle de dispositivo (IPDC) e o Megaco /

H.248. O SGCP é o protocolo de sinalização Mestre Escravo (*Master Slave*) baseado em ASCII original para o VoIP. O MGCP surgiu no ano seguinte, combinando características do SGCP e do IPDC com mais habilidades. O Megaco é um protocolo semelhante que o IETF desenvolveu com mais habilidades ainda.

Tanto o SCGP como o MGCP são projetados como protocolos de sistema distribuído que dão ao usuário a aparência de um sistema VoIP único. Eles são protocolos *stateless* no sentido de que a seqüência de transação entre o MG e o agente da chamada pode ser realizada sem qualquer lembrança de transações anteriores.

Tanto o MGCP como Megaco fornecem as seguintes funções de *gateways* de mídia:

- Cria, modifica e apaga conexões usando qualquer combinação de rede de trânsito, incluindo Frame Relay, ATM, TDM, *Ethernet* ou analógico. As conexões podem ser estabelecidas para transmissão de pacotes de áudio através de alguns tipos de redes de transporte:
    - Redes IP usando RTP e/ou UDP;
    - Redes ATM usando AAL2 ou uma outra camada de adaptação;
- Cria uma conexão interna, tal como a TDM. Isto é usado para conexões que terminem em um *gateway*, mas que são imediatamente redirecionadas através da rede telefônica;
- Detecta ou gera eventos nos terminais ou conexões. Por exemplo, um *gateway* pode detectar dígitos digitados ou gerar um tom de retorno de chamada em uma conexão. Um agente chamador usará o MGCP para enviar “pedidos de notificação”, os quais incluem uma lista de “eventos” que os *gateways* de mídia detectam. O protocolo usa a lista de “Eventos Pedidos”, o “Mapa de Dígitos” e a lista de “Detecção de Eventos” no manuseio destes eventos.

Quando um evento é detectado, o *gateway* de mídia realiza alguma ação, como especificado pelo agente chamador, tal como, informar o evento ou aplicar um outro ton na conexão, além de juntar dígitos de acordo um mapa de dígitos recebido do agente chamador e enviar um conjunto completo de dígitos digitados pelo agente chamador, viabilizar serviços, como por exemplo, conferência e informar estatísticas sobre as chamadas.

O mecanismo de dígitos permite aos agentes chamados servir um grande número de *gateways* residenciais e de pequenas empresas. Ele pode ser usado para coletar, não apenas números de telefones de destino, mas também para acessar códigos, números de cartão de crédito, etc. A exigência de coletar dígitos de acordo com um mapa de dígitos está relacionada com a eficiência da comunicação entre o MG e o agente chamador em um sistema distribuído.

Se o *gateway* fosse enviar para o agente chamador cada dígito digitado, separadamente, tão logo eles fossem digitados, haveria uma grande quantidade de interações desnecessárias. Por esta razão, o *gateway* deve armazenar os dígitos em um *buffer* e enviar um conjunto completo para o agente chamador. O *gateway* precisa saber quantos dígitos acumular antes da transmissão. Por exemplo, o dígito “O” sozinho poderia ser usado para conectar ao operador local, quatro dígitos “xxxx” poderia ser um número de uma extensão local, “xxxxxxx” poderia ser um número de um plano de digitação particular de uma companhia, e 9011 + até 15 dígitos, poderia ser um número internacional. O sistema VoIP distribuído pode usar o MGCP para enviar para o *gateway* um mapa de dígitos que corresponda ao plano de dígitos. Os mapas de dígitos simplesmente definem uma maneira para que o *gateway* combine seqüências de dígitos digitados de acordo com algumas regras definidas no plano.

Exceto por algumas diferenças na terminologia, o protocolo Megaco fornece ao agente chamador mais flexibilidade com relação ao tipo de transporte e controle sobre o *gateway* de mídia, assim como alguns recursos para aplicações como vídeo-conferência. Tanto o MGCP como o Megaco fornecem um procedimento para o agente chamador enviar um pacote de propriedades, sinais ou eventos para o *gateway* para o uso em linhas e troncos ligados ao *gateway*. Os conteúdos do pacote não são uma parte de nenhum protocolo, por isso o executador pode definir ou mudar os pacotes sem qualquer mudança no protocolo. O Megaco tem uma maneira pré-definida para o agente chamador e *gateway* negociarem a versão a ser usada, porém o MGCP não tem um mecanismo de controle de versão, tendo de confiar no processo de negociação do vendedor da patente.

Nos aspectos de segurança e qualidade do serviço, o Megaco é mais flexível do que o MGCP. Enquanto o MGCP fornece apenas o IPSec, o Megaco fornece também um cabeçalho de autenticação. Ao menos, os protocolos fornecem autenticação do endereço da fonte. Enquanto que o MGCP fornece apenas o UDP para mensagens de sinalização, o Megaco fornece UDP, TCP, ATM e SCPT. O Megaco tem também um melhor mecanismo de fluxo e melhor mecanismo de alocação de recursos.

O MGCP ou o Megaco podem ser usados para uma Arquitetura VoIP Master Slave, principalmente quando o objetivo é controlar muitos *gateways* de telefonia IP de baixo custo.

Para comunicação entre agentes chamadores, ou para controle de grupos de troncos, o SIP pode ser mais adequado. Enquanto o MGCP e o Megaco têm comandos específicos para controle de chamadas VoIP, o SIP permite que uma primitiva simples possa ser usada para fornecer diferentes serviços. Consequentemente, o SIP oferece a promessa de fornecer uma larga gama de

serviços através da telefonia básica, incluindo mensagem instantânea, comércio eletrônico na web com capacitação para voz. O SIP facilita o desenvolvimento de novas aplicações por terceiros independentes. Alguns projetistas usam o MGCP ou o Megaco para controlar os *gateways*, porém usam o SIP na camada de aplicação.

2.5 EQUIPAMENTOS DE UMA SOLUÇÃO *VoIP*

Dentro do contexto de telefonia IP, existem equipamentos específicos da telefonia IP que são os *gatekeepers* ou servidores de telefonia (funcionalidade de um PABX) e os *gateways* IP ou ITGs. Os roteadores e PABXs estão envolvidos na operação de telefonia IP e possuem a função de roteamento de pacotes e encaminhamento de chamadas conforme já é conhecido no ambiente de rede e telefonia convencional.

2.5.1 *GATEWAY* IP

O *gateway* IP é o equipamento que faz a interface da rede telefônica pública convencional (PSTN) ou de uma rede telefônica privada com a rede IP. Desta forma, o *gateway* IP digitaliza a voz, faz compactação e roteia pacotes de voz através de uma rede IP, assim como recebe pacotes de voz, descompacta, decodifica e encaminha na rede telefônica.

Além de funções de roteamento de pacotes de voz na rede IP, um *gateway* IP possui funções relacionadas a aspectos de uma chamada telefônica, como por exemplo a configuração da chamada (*call setup*), sinalização PSTN, mapeamento de números telefônicos em endereços IP, supressão do silêncio e encapsulamento IP. A figura 4 mostra um esquema que ilustra a funcionalidade de um *gateway* IP.

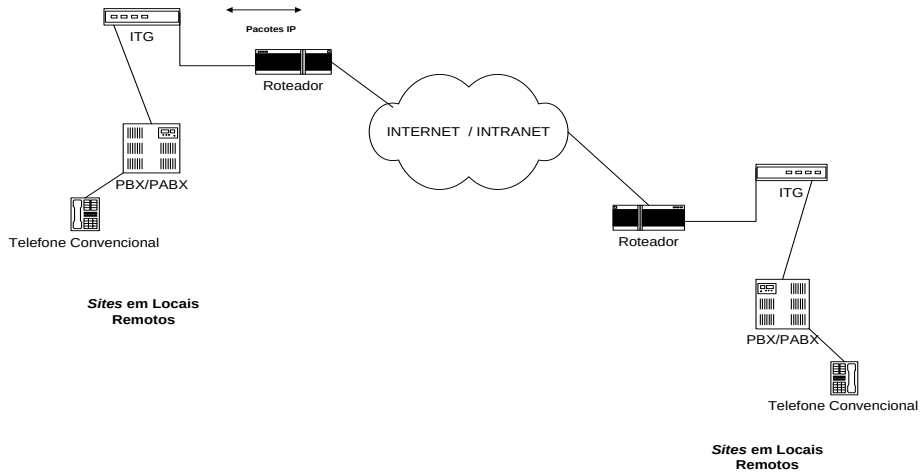

Figura 4. Funcionalidade do gateway IP

As interfaces de um *gateway* IP com a rede pública são as interfaces com equipamentos telefônicos e fax, interfaces com PABX e interfaces com estações que emulam um telefone para conexão a um PABX ou central telefônica.

As interfaces com a rede IP são as interfaces de rede (*Ethernet*, ATM, E1/T1, etc) e a conexão com os roteadores da rede IP para roteamento. Além destas, existem interfaces digitais para conexão com redes como por exemplo interfaces RDSI, E1 e outras mais.

Gateways provêem interconexão com tecnologias que não são H.323, como por exemplo vídeo conferência ou aplicações telefônicas baseadas em recomendações H.320. Um exemplo de

gateway H.323 poderia ser um roteador Cisco com interfaces de voz. Um telefone da rede pública poderia ser conectado através de um *gateway* e funcionar aparentemente na rede H.323 como um elemento final da rede, podendo fazer ligações para a rede pública através do *gateway* e também receber chamadas. Os *gateways* gerenciam a conversão da sinalização de chamadas e a conversão da sinalização do meio quando um *gateway* H.323 é conectado a uma rede de outro padrão.

2.5.2 GATEKEEPERS

O *gatekeeper* é um componente essencial de qualquer infra-estrutura telefônica em redes de dados. Permite a gerência da solução de VoIP. Basicamente, consiste em implementar a funcionalidade dos PABXs sobre as redes IP. Desta forma, o *gatekeeper* pode ser considerado um equipamento com função administrativa que funciona como um servidor que implementa funções da telefonia convencional em uma rede IP.

O *gatekeeper* permite o aumento da flexibilidade e recursos para usuários empresariais e provedores de serviço, de forma a implementar novas aplicações e trazer grande capacidade e confiabilidade a redes de dados e voz sobre IP. Nas empresas, o *gatekeeper* traz facilidades de gerenciamento com a convergência entre dados e voz.

A figura 5 mostra as conexões de um *gatekeeper*.

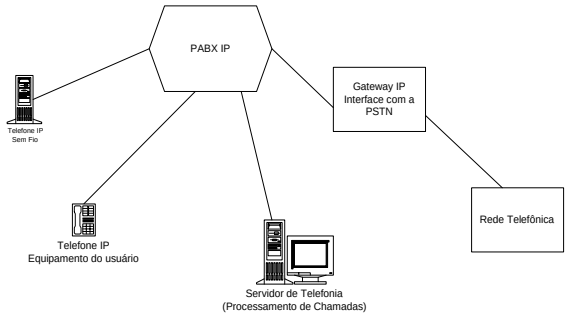

Figura 5. Conexões de um Gatekeeper

Fisicamente, o *gatekeeper* é um PC (servidor NT, por exemplo) com interface de rede. Ele utiliza o padrão H.323 para comunicações multimídia em tempo real através de redes de pacotes e padrões de sinalização como o Q.931, H.245 e H.225 (Keagy, 2000).

O módulo de processamento de chamadas possui funções semelhantes àquelas realizadas por um PABX, como por exemplo *hold*, transferências, *forward*, *display* de mensagens, *call waiting*, conferência, compressão, distribuição de chamada e interface de correio de voz.

O servidor *gatekeeper* é gerenciável através de uma interface *web* e possui uma arquitetura tolerante a falhas.

O *gatekeeper* é desenvolvido em duas camadas. A camada mais interna, o núcleo, é composto de funcionalidades de empilhamento e Unidades de Controle Multiponto (MCU). A camada externa, chamada de "Camada de Interface", consiste de API´s que provêem interfaces bem definidas para muitos serviços agregados, que existem nas redes hoje em dia. Isso garante máxima flexibilidade e escalabilidade do *gatekeeper* e novas aplicações podem ser desenvolvidas e integradas a qualquer momento.

O *gatekeeper* H.323 realiza controle de acesso e endereça funções de tradução. Alguns *gatekeepers* podem se comunicar uns com os outros para coordenar os seus serviços de controle. Redes com *gateways* VoIP devem, mas não são obrigados a, ter *gatekeeper* para traduzir endereços E.164 que chegam em endereços de transporte (e.x endereços IP e número de porta). O Gatekeeper é logicamente separado das outras entidades H. 323, mas fisicamente ele pode coexistir com um terminal, *gateway* ou um proxy H.323.

Quando presente em uma rede VoIP, o *gatekeeper* fornece as seguintes funções:

- Tradução do endereço

O *gatekeeper* traduz endereços (e.x: números de telefone E.164) para endereços de transporte, usando uma mesa de tradução que é atualizada usando mensagens de registros e outros meios.

- Mapeamento de endereços

O mapeamento de endereços consiste na conversão de endereços de rede e identificadores TSAP. Por exemplo, um *gatekeeper* pode receber uma chamada para fulano@empresa.com ou para +1-408-555-1212 e converter para um endereço de rede (normalmente IP) e um número de porta UDP ou TCP. *Gateways* H.323 estão habilitados a resolver apelidos em endereços de rede e identificadores TSAP que são obtidos quando a estação se registra com eles. Por convenção, qualquer apelido deve conter o identificador do dispositivo e o nome do domínio DNS onde o *gatekeeper* reside para que apelidos H.323 possam ser resolvidos, mesmo em domínios DNS grandes, onde existem muitos *gatekeepers* (para zonas H.323 diferentes) no mesmo domínio DNS.

- Controle de acesso

O *gatekeeper* autoriza o acesso à rede usando mensagens H.225. Os critérios de acesso podem incluir autorização de chamada, largura de banda ou outras políticas.

- Controle da Largura de Banda

O *gatekeeper* controla quanto de largura de banda um terminal utiliza.

- Administração de Zona

Um terminal pode registrar-se com apenas um *gatekeeper* de cada vez. O *gatekeeper* fornece as funções acima para terminais e *gateways* que tenham sido registrados com ele.

- Controle de admissões

O *gatekeepe*r autoriza terminais, *gateways* e MCUs a fazer chamadas na rede através de um canal RAS. O *gatekeeper* confirma a admissão de chamadas através de mensagens ACF ou rejeita através de chamadas ARJ. Controla também a largura de banda, respondendo as solicitações de largura de banda (BRQ) através de mensagens de confirmação (BCF) ou rejeição (BRJ).

- Roteamento de Chamadas

Os *gatekeepers* provêem um mecanismo centralizado para gerenciar planos de discagem e roteamento de chamadas em uma rede VoIP. Cada *gateway* IP deve manter informações de roteamento de chamadas para todos os destinos, mesmo que um método não H.323 de roteamento de chamadas seja implementado. Além disto, provê políticas baseadas em alocação de recursos, como por exemplo, uma política pode instruir um *gatekeeper* a restringir chamadas baseadas em um destino específico, privilégios de uso ou horário do dia.

- Controle do canal de sinalização de chamadas

O canal RAS carrega mensagens usadas no processo de registro do terminal do *gatekeeper*, associando um pseudônimo do terminal ao o seu endereço TCP/IP e número de porta, para ser usado por sinalização de chamada. O canal RAS é também usado para transmissão de acesso, troca de largura de banda e para liberar mensagens entre um terminal e o seu *gatekeeper*.

O Canal de Sinalização de Chamada carrega a mensagem de controle de chamada H.225.0 usando o TCP, tornando-se um canal confiável. Os terminais H.323 e os *gatekeepers* usam mensagem Q.931 (com TCP) para sinalização de chamada. Em redes que não possuem *gatekeeper*, os pontos fins enviam mensagens de sinalização de chamada diretamente para o terminal chamado usando o CSTA.

Se a rede tem um *gatekeeper*, o terminal que chama e envia uma mensagem de acesso inicial para o *gatekeeper* usando o canal de transporte de endereço RAS do *gatekeeper*. Na troca inicial de mensagens de acesso, o *gatekeeper* diz ao ponto fim originador da chamada ou para iniciar a mensagem de sinalização de chamada diretamente para o outro terminal ou para direcioná-las através do *gatekeeper*.

A participação em sinalização de controle de chamada é opcional. A sinalização de chamada pode ser direcionada de duas maneiras. Na sinalização direta, o ponto final envia mensagem de sinalização de chamada diretamente entre os terminais ou *gateways*. A sinalização de chamada pode ser direcionada pelo *gatekeeper*, o qual direciona as mensagens de sinalização de um terminal para outro.

Na sinalização de chamada direta do terminal, o *gatekeeper* participa no acesso da chamada, porém tem pouco conhecimento direto das conexões. Devido a este envolvimento limitado, um único *gatekeeper* pode procurar um grande número de chamadas, mas o *gatekeeper* tem uma habilidade limitada de realizar funções de administração de serviço. O *gatekeeper* não pode determinar as taxas de conclusão de chamada, e, se ele tiver que realizar a gravação de detalhe da chamada, ele vai depender dos terminais para a informação da duração da chamada.

O método de sinalização da chamada roteada pelo *gatekeeper* resulta em mais carga desde que ele tem de processar as mensagens Q.931. O *gatekeeper* pode fechar o canal de sinalização de chamada depois que a ligação da chamada for completada. Contudo, se o *gatekeeper* continuar envolvido na chamada, como por exemplo, para fazer gravações da chamada ou para fornecer serviços suplementares, ele manterá o canal aberto durante a duração da chamada.

Tanto o H. 225 como o H.245 usam o TCP para estabelecer uma conexão de transporte confiável entre os pontos fins, os *gateways* e os *gatekeepers*. No caso da sinalização de chamada roteada pelo *gatekeeper*, as conexões TCP são mantidas na duração da chamada. Embora normalmente confiável, o fracasso de uma conexão TCP pode resultar em uma interrupção no meio da chamada mesmo que a conexão TCP não exija um uso nesta hora. Por exemplo, supondo que a sinalização de chamada roteada pelo *gatekeeper* seja usada e a conexão TCP do *gateway* para o *gatekeeper* seja interrompida devido a um *timeout* ou a um fracasso na troca de mensagem *keepalive* durante a interrupção de um link. As chamadas podem cair mesmo que o fluxo de voz RTP não tenha sido afetado pelo evento na rede que fez com que a conexão da TCP com o *gatekeeper* falhasse.

Os *gatekeepers* trabalham com API´s que suportam serviços avançados, tais como:

- Configuração da chamada (*Call Setup);*
- Controle de Autenticação e Autorização de Usuário: Manipula todas as solicitações de autorização dos usuários, interagindo com a base de dados de perfis de usuários para facilitar o serviço de autenticação, além de gerenciar a banda, provendo diferentes níveis de serviço;
- Administração de interface: Através da console de administração, permite que usuários, com privilégio, possam configurar, modificar e deletar as configurações de usuários, seus perfis e informações. Pode ter interface com um servidor Web para permitir administração remota. Poderá, também, apresentar alarmes, *displays* do *status* da rede e estatísticas de utilização;
- Gerenciamento da Rede: A interface de gerenciamento da rede poderá fornecer as seguintes informações ao agente SNMP:
    - Número de terminais registrados;

    - Número de terminais ativos;
    - Número de chamadas ativas;
    - Banda alocada;
    - Banda em uso;
    - Banda reservada para alocação;
    - Recursos do *gateway* alocados;
    - Recursos do *gateway* disponíveis;
    - Recursos de MCU disponíveis;
    - Recursos de MCU alocados;
    - Status de operação;
- Segurança
    - Criptografia;
    - Autenticação;
- Recursos adicionais
    - Seleção de nível de QoS;
    - Chamada em espera;
    - Chamada suspensa;
    - Transferência de chamada;
    - Redirecionamento de chamadas;

    - Discagem abreviada;
    - Linhas prioritárias;
- Serviços e recursos de mídia (MCU)
    - Unicast – manda a mídia para um terminal;
    - Multicast – Manda para cada destino diretamente;
    - *Híbrido:* Um pouco de cada um dos anteriores;
    - Serviço de alarme;
    - Serviços de voice mail;
    - Serviço de resposta de voz interativa;
- Serviços de diretórios
    - Opera com servidores DNS na rede;
- Módulo de tarifação (*billing)*
    - Transforma informações do status das chamadas para o sistema de tarifação em rede;

3 CRITÉRIOS DE QUALIDADE DE SERVIÇO PARA REDES DE VOZ

3.1 INTRODUÇÃO

As redes IP *best effort* não oferecem nenhum tipo de QoS, sendo esta a principal limitação imposta para a execução eficiente das aplicações VoIP.

Redes de pacotes devem ser adaptadas para suportar os requisitos de QoS necessários a uma transmissão de voz com qualidade. Sem otimização, estas redes introduzem atrasos variados e informações de tempo real são perdidas devido a uma variedade de razões. Estes efeitos degradam a qualidade da voz para chamadas realizadas através da rede.

A qualidade de serviço nas redes IP é um aspecto operacional fundamental para o desempenho fim-a-fim das aplicações VoIP. Assim sendo, é importante o entendimento dos seus princípios, parâmetros, mecanismos, algoritmos e protocolos desenvolvidos e utilizados para a obtenção da QoS desejada.

QoS é definido pelo ITU-T na recomendação I.350 como: "Qualidade de Serviço é o efeito coletivo de performance que determina o grau de satisfação do usuário deste serviço específico".

Segundo (Martins, 1999), Qualidade de Serviço (QoS) é um requisito da(s) aplicação(ões) para a qual exige-se que determinados parâmetros (atrasos, vazão, perdas, ...) estejam dentro de limites bem definidos (valor mínimo, valor máximo).

Há vários caminhos para obter QoS em redes IP. Antes de discutir as opções de QoS, deve-se considerar se o QoS é realmente necessário. Alguns engenheiros de rede afirmam que a maneira de oferecer um bom desempenho para uma rede IP provisionar recursos, o que elimnaria a necessidade de utilizar complicados protocolos de QoS. Se nenhum link em uma rede IP está

mais do que 30% ocupado, mesmo em condições de pico de tráfego, então os pacotes devem fluir sem qualquer fila de atraso e protocolos elaborados com a finalidade de dar prioridade para um tipo de pacote não são necessários (Goode, 2002).

O projetista da rede deve considerar a capacidade dos componentes do roteador para enviar pequenos pacotes de voz, assim como a largura da banda dos enlaces que interligam os roteadores na determinação da ocupação da rede. Se a ocupação for baixa, então o desempenho deve ser bom. Essencialmente, o debate é sobre se uma excessiva capacidade da rede (incluindo largura de banda dos links e roteadores) é menos custosa do que a implementação do QoS.

O desenvolvimento do QoS tem continuado devido à percepção de alguns engenheiros de rede de que o tráfego em tempo real (assim como o de outras aplicações) pode, algumas vezes, requerer tratamento prioritário para alcançar bom desempenho. Em alguns casos, adquirir enlaces pode ser caro e a banda larga difícil de ser obtida. Por conseguinte, QoS pode ser desejável no acesso de links mesmo se o núcleo da rede estiver levemente carregado. Acesso sem fio aos links são especialmente caros, portanto o QoS é importante para chamadas telefônicas de celulares IP.

A QoS é garantida pela rede, suas componentes e equipamentos utilizados. Do ponto de vista dos programas de aplicação, a QoS é tipicamente expressa e solicitada em termos de uma "Solicitação de Serviço" ou "Contrato de Serviço". A solicitação de QoS da aplicação é denominada tipicamente de SLA (*Service Level Agreement)* (Martins, 1999).

A SLA deve definir claramente quais requisitos devem ser garantidos para que a(s) aplicação(ões) possam executar com qualidade. Um exemplo típico de SLA para uma aplicação de voz sobre IP com algumas centenas de canais de voz simultâneos numa rede IP WAN poderia ser:

- Perda $\leq$ 10 %;

- Atraso ≤ 200 mseg;
- Jitter ≤ 40 mseg.

Uma vez que a rede garanta esta SLA, tem-se como resultado que a aplicação VoIP em questão poderá executar garantindo a qualidade de voz prevista para os seus usuários se comunicando simultaneamente através da rede IP.

Os parâmetros de QoS necessários para execução de uma aplicação VoIP com qualidade serão discutidos no tópico 3.4.

As considerações que seguem buscam definir técnicas de QoS que viabilizem o alcance dos requisitos de QoS que redes de pacotes de voz devem ter para garantir uma boa qualidade na voz.

3.2 QoS - Alternativas Técnicas

As alternativas técnicas básicas para o provimento da qualidade de serviço em redes IP são as seguintes:

- IntServ - *Integrated Services Architecture* com o RSVP;
- DiffServ - *Differentiated Services Framework;*
- MPLS - *MultiProtocol Label Switching.*

Em seguida, resume-se os princípios, os mecanismos específicos e a estratégia destas alternativas técnicas.

3.2.1 Int Serv - Integrated Services Architecture com o RSVP

A alternativa técnica IntServ (White, 1997) foi definida pelo IETF e corresponde a um conjunto de recomendações (RFCs) visando a implantação de uma infra-estrutura robusta para a Internet

que possa suportar o transporte de áudio, vídeo e dados em tempo real além do tráfego de dados transportado na infra-estrutura atual (Martins, 1999).

O conjunto de recomendações proposto é denominado de "Arquitetura de Serviços Integrados" (*Integrated Services Architecture*) e visa obter uma garantia de qualidade de serviço para as aplicações.

O conceito de InterServ consiste em reservar recursos para cada fluxo através da rede. O RSVP (Braden et al., 1997) foi inicialmente utilizado para ser o protocolo de reserva. Quando uma aplicação requer um QoS específico para o seu fluxo de dados, o RSVP pode ser usado para entregar o pedido para cada roteador e para manter a condição do roteador de fornecer o serviço requisitado. O RSVP transmite dois tipos de fluxos de voz de acordo com as normas do IntServ. As especificação do tráfego (Tspec) descreve o fluxo, e a especificação do serviço requisitado (Rspec) descreve o serviço requisitado supondo que o fluxo seja transmitido de acordo ao Tspec. Melhoramentos do IntServ permitem uma escolha de Serviço Garantido ou Serviço de Carga Controlada.

O Serviço Garantido envolve um policiamento do tráfego por um modelo de *Token Bucket* para controlar o tráfego médio. O pico de tráfego é limitado por uma taxa parâmetro de pico “p” e um intervalo “T” de maneira que não mais do que P * T bytes sejam transmitidos em qualquer intervalo T. O tamanho do pacote é restrito à faixa [m, M], de maneira que os pacotes menores sejam considerados como sendo do tamanho de m, e os pacotes maiores do que M estarão violando o contrato. É determinada uma largura de banda exigida, e uma largura de banda suficiente é reservada para cada enlace para satisfazer todas as exigências do fluxo.

Se cada nó e enlace podem aceitar o serviço pedido, o fluxo deve ocorrer sem perda porque o tamanho da fila reservada para o fluxo pode ser configurado no parâmetro de tamanho do *Token Bucket*. Esse serviço é projetado para aplicações interativas em tempo real. Para usá-lo eficientemente é necessário um dimensionamento do atraso fim-a-fim realista e preciso, além de uma largura de banda adequada ao fluxo.

O Serviço de Carga Controlada usa o mesmo Tspec do Serviço Garantido. Contudo, não foi definido um Rspec. Os fluxos que usarem esse serviço devem experimentar o mesmo desempenho que eles teriam em uma rede *best effort* levemente carregada. O Serviço de Carga Controlada seria adequado para o controle de acesso de chamada e evitaria os atrasos e as perdas de pacote que prejudicam o tráfego em tempo real quando a rede está congestionada.

Há algumas razões para não usar o IntServ com RSVP para telefonia IP. Embora o IntServ junto com o RSVP funcionaria em uma rede privada para pequenas quantidades de tráfego, o grande número de chamadas de voz que os fornecedores de telefonia IP realizam nas suas redes, sobrecarregariam um sistema IntServ RSVP. Primeiro, a largura de banda exigida para voz é pequena, e o controle de tráfego RSVP seria uma parte significativa do tráfego total. Segundo, o código do roteador do RSVP não foi projetado para administrar muitos milhares de conexões simultâneas por roteador.

Deve ser observado, contudo, que o RSVP é um protocolo de sinalização, e ele tem sido proposto para uso em contextos outros que não o IntServ. Por exemplo, RSVP-TE é o protocolo de roteamento baseado no confinamento para estabelecer um LSP combinado com uma largura de banda e linhas especificadas em uma rede MPLS. O RSVP tem sido proposto também como o

mecanismo de controle de aceitação de grau de chamada para o VoIP em redes de serviços diferenciados (Goode, 2002).

3.2.2 DIFFSERV - DIFFERENTIATED SERVICES

A qualidade de serviço na solução *DiffServ* é garantida através de mecanismos de priorização de pacotes na rede. A solução *DiffServ* não utiliza nenhum tipo de mecanismo de reserva de recursos. Nesta solução os pacotes são classificados, marcados e processados segundo o seu rótulo DSCP.

A idéia básica da solução *DiffServ* é reduzir o nível de processamento necessário nos roteadores para fluxos de dados (*streams*). Isto é realizado com a definição de poucas "Classes de Serviço" numa estrutura comum de rede. Os inúmeros fluxos de tráfego (Pacotes IP) gerados pelas aplicações são agregados a poucas classes de serviço em função da qualidade de serviço especificada para o fluxo. Esta tarefa é tipicamente realizada nos roteadores de entrada do *backbone* (*Edge routers*) e, desta forma, o processamento nos roteadores intermediários (*Core*) fica mais simplificado e independente dos fluxos individuais das aplicações (Martins, 1999).

O funcionamento da estratégia *DiffServ* será detalhado no capítulo 4, pois esta foi a alternativa utilizada na rede simulada.

As alternativas IntServ e *DiffServ* não são concorrentes ou mutualmente exclusivas. Na realidade, estas são soluções complementares que podem ser utilizadas conjuntamente. Uma alternativa de uso conjunto das duas soluções seria a utilização do *DiffServ* no *backbone* de roteadores (*core*), na medida em que é uma solução mais "leve" e o IntServ/ RSVP nas redes de acesso, na medida em que provê um bom controle com granularidade dos requisitos de QoS das aplicações (Martins, 1999).

3.2.3 MPLS - MULTIPROTOCOL LABEL SWITCHING

MPLS capacita redes IP a oferecer serviços diferenciados que viabilizam a qualidade de serviço necessária à execução de aplicações VoIP.

Para a operação efetiva do MPLS faz-se necessário a distribuição de rótulos (MPLS *Labels*) entre os roteadores e a gerência dos mesmos. O protocolo LDP (Thomas *et al.*,1999) é um protocolo de sinalização desenvolvido com esta finalidade.

A função de roteamento é executada pelos roteadores de fronteira (LSRs) que são conectados através de LSPs.

No que toca a operação, o MPLS utiliza os seus rótulos basicamente para indicar o próximo roteador (*Next hop*) para onde o pacote deve ser encaminhado (*Forwarding*). Os roteadores de fronteira (um LSR) anexa pequenos rótulos aos pacotes. Cada LSR, em conjunto com um LSP, troca o rótulo e encaminha o pacote para o próximo LSR. O último LSR remove o rótulo e trata o pacote como um pacote IP normal.

Políticas de QoS podem ser determinadas. O campo EXP do rótulo MPLS, por exemplo, pode ser configurado para que cada roteador saiba dar aos pacotes de voz a mais alta prioridade e a maior quantidade de largura de banda possível pode ser configurada para pacotes de voz. Quando a banda não estiver sendo necessária para o tráfego de voz, a mesma é liberada para outras classes de baixa prioridade.

3.3 QOS - MECANISMOS

As alternativas técnicas discutidas são implementadas através da utilização de diversos tipos de mecanismos, a saber:

- Protocolos de sinalização;

- Algoritmos de prioridade;
- Algoritmos de escalonamento;
- Algoritmos de controle de filas;
- Algoritmos de congestionamento.

Em seguida, será discutido a funcionalidade e aplicabilidade de cada um destes mecanismos e identificamos implementações dos mesmos que são utilizadas em roteadores, *hosts* e outros equipamentos visando a garantia de qualidade de serviço.

3.3.1 PROTOCOLOS DE SINALIZAÇÃO

Como já citado no capítulo 2, a finalidade de um protocolo de sinalização (*Signalling Protocol*) (Chwan e Irwin, 1998) no contexto da qualidade de serviço em redes IP pode ser entendida como descrita a seguir.

O protocolo de sinalização é utilizado pelas aplicações (*hosts*) para informar ou solicitar à rede sua necessidade de QoS. Além disso, os protocolos de sinalização permitem também que os equipamentos de rede (Roteadores, etc) possam trocar informações no sentido de cooperarem visando a garantia da qualidade de serviço aceita pela rede.

Maiores detalhes sobre estes protocolos podem ser obtidos no capítulo 2.

3.3.2 ALGORITMOS DE PRIORIDADES

Os algoritmos de prioridade (*Priority Algorithms*) são um outro mecanismo utilizado pelos equipamentos de rede para a garantia da qualidade de serviço. Neste contexto, a prioridade pode ser entendida como um mecanismo que provê diferentes tempos de espera para o processamento da informação.

Utilizando estes algoritmos, o tráfego de voz pode ser priorizado através da configuração de um valor de mais alta prioridade em relação aos demais tráfegos, o que implicaria na liberação de recursos compartilhados como largura de banda e consequentemente beneficiaria o tráfego de voz no que diz respeito ao atraso, *jitter* e percentual de perdas.

Segue alguns exemplos de algoritmos utilizados:

- *IP Precedence*

IP Precedence utiliza os três bits de precedência no cabeçalho do IPV4, campo TOS *(Type Of Service*), para especificar uma classe de serviço para cada pacote.

Os bits *IP Precedence* são utilizados para marcar um pacote com a classe de serviço desejada quando o pacote entra na rede. Os roteadores procuram os bits *IP Precedence* e aplicam ao pacote as ações referentes à classe de serviço associada ao pacote. A especificação IP define valores específicos para serem usados nesse campo de acordo com os vários tipos de tráfego. Este esquema utilizado para alocação de recursos é baseado na importância de diferentes fluxos de tráfego.

Os mecanismos básicos para o processamento utilizando precedência em um roteador são alocação preferencial de recurso, incluindo o serviço de fila ordenada por precedência, controle de congestionamento baseado na precedência e seleção das características de prioridade da camada de enlace. O roteador também seleciona o *IP Precedence* para roteamento, gerenciamento e controle de tráfego por ele gerado.

Um roteador que suporte *IP Precedence* deve usar a indicação da precedência em qualquer ponto cujo processamento está relacionado com a alocação de recursos finitos como *buffers*.

- *Priority Queuing*

Priority Queuing (PQ) é um esquema rígido de priorização de tráfego: Se uma pacote A possui uma prioridade mais alta que o pacote B, então o pacote A sempre será enviado pela interface antes do pacote B.

PQ é um algoritmo de prioridade que permite atribuir níveis de prioridade diferentes para tráfegos com importâncias diferentes.

Em PQ, cada pacote é colocado em uma das quatro filas - alta, média, normal ou baixa – baseada no nível de prioridade. Pacotes que não são classificados por prioridade, são colocados na fila normal. Dessa forma, durante uma transmissão, o algoritmo dá preferência aos pacotes da fila de alta prioridade sobre os pacotes da fila de baixa prioridade.

3.3.3 ALGORITMOS DE ESCALONAMENTO

No contexto VoIP discutido, o mecanismo de escalonamento, tipicamente presente em equipamentos roteadores, procura garantir que fluxos (*streams*) diferentes de pacotes obtenham os recursos que lhes foram alocados (banda e processamento).

Seguem alguns mecanismos de escalonamento utilizados:

- WRR - *Weighted Round Robin;*
- GPS - *Generalized Processor Sharing;*
- CBQ - *Class Based Queuing;*
- WFQ - *Weighted Fair Queuing.*

O funcionamento do algoritmo WRR será detalhado no capítulo 4 por se tratar do algoritmo utilizado na rede simulada.

3.3.4 ALGORITMOS DE CONGESTIONAMENTO

Os mecanismos de controle de congestionamento são também importantes para a garantia da qualidade de serviço de uma aplicação VoIP. A idéia básica destes mecanismos é a inibição dos fluxos de pacotes durante o período de congestionamento de forma que os geradores de fluxos de pacotes IP reduzam a sua carga sobre a rede. Com menos pacotes sendo entregues à rede tem-se uma tendência de redução no nível de congestionamento.

Seguem alguns exemplos de algoritmos lidando com o congestionamento de filas de pacotes IP:

- RED - *Random Early Detection* (Floyd e Jacobson, 1993);
- WRED - *Weighted Random Early Detection;*
- ECN - *Explicit Congestion Notification.*

O funcionamento do algoritmo RED será detalhado no capítulo 4 por se tratar do algoritmo utilizado na rede simulada.

3.4 QoS – PARÂMETROS PARA APLICAÇÕES VoIP

Os seguintes parâmetros definem QoS em uma rede de pacotes para aplicações VoIP (Huston e Ferguson, 1998) :

- Taxa de Perdas;
- Atraso;
- Variação do Atraso (*jitter*);
- Largura de Banda.

Para suportar pacotes de aplicações telefônicas e aplicações de tempo real, uma rede de pacotes deve prover alta confiança, baixo atraso, pequena variação do atraso e largura de banda suficiente. É importante observar que redes de pacotes não precisam prover tal desempenho para todas as aplicações da rede.

Os próximos tópicos examinam cada um dos parâmetros de QoS e considera problemas e soluções associados com os requisitos de QoS para pacotes que armazenam dados de uma aplicação telefônica.

3.4.1 TAXA DE PERDAS

Protocolos de tempo real possuem muito requisitos de confiança, devido ao fato de que a retransmissão não é uma opção. Qualquer informação perdida que é retransmitida chega muito tarde para ser útil. Aplicações telefônicas são especialmente sensíveis a esta condição. Se em uma conversa telefônica não se consegue ouvir nenhuma palavra, a conversa pode ficar comprometida.

Em função da existência de aplicações em tempo real, redes de pacotes necessitam ter atrasos mínimos e previsíveis que não ultraprassem determinados limites. Aplicações sensíveis ao atraso, como por exemplo aplicações VoIP, guardam largura de banda e processam a carga para tentar reduzir o número de pacotes perdidos.

Para redes IP, o serviço confiável do TCP não é apropriado para aplicações de tempo real porque o TCP usa a retransmissão para garantir a confiança. Isto é uma das razões que o RTP, protocolo padrão de transmissão de áudio para implementações VoIP, é baseado no protocolo UDP Retransmissões devem ser evitadas porque o áudio pode suportar pequenas perdas de pacotes , mas possui severos requisitos de temporização. Ou seja, o receptor reproduz o áudio

continuamente e portanto os pacotes devem chegar ao seu destino antes dos seus respectivos pontos de reprodução ou então seriam considerados perdidos ou atrasados. Desta forma, aplicações VoIP teriam seu desempenho prejudicado em função do atraso e da variação do atraso imposto pelas retransmissões.

A taxa de perdas de pacotes é calculada no lado do receptor como a razão entre as quantidades 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 estes sofrem nas filas dos roteadores quando estas se tornam cheias, não permitindo mais admissão e armazenamento de pacotes para posterior envio. Protocolos como o TCP se recuperam desta situação através da detecção e reenvio dos pacotes que foram perdidos. Como uma aplicação VoIP utiliza o UDP como protocolo de transporte, que não retransmite pacotes perdidos, estas perdas não são recuperadas, e nem deveriam ser, pois os requisitos críticos de temporização destas aplicações impedem a utilização de tempo adicional necessário ao processo de retransmissão. É importante ressaltar que aqueles pacotes que chegam atrasados demais no destino não são contabilizados como perdidos, apesar de não serem reproduzidos. Estima-se que a taxa de perdas em uma aplicação VoIP deva ser inferior a 10 % (ITU-T – recomendação G.114).

Um método para aumentar a confiança sem necessitar de retransmissão de pacotes é usar o FEC. Com esta técnica, pacotes importantes de uma seqüência de informação são enviados redundantemente através da rede. A idéia é considerar que se um pacote com uma parte da seqüência de dados é perdida, um outro pacote com a mesma informação normalmente chegará ao destino. Introduzir redundância aumenta a confiança da informação transferida. FEC provê aumento da confiança através de um custo maior de largura de banda, uma vez que cópias da

informação são transmitidas. Um exemplo de aplicação que utiliza esta abordagem seria uma aplicação de videoconferência.

Há dois tipos de confiança a considerar em aplicações envolvendo tráfego de voz, como por exemplo aplicações VoIP:

- Confiança em um procedimento de chamada

Em um procedimento de chamada, seqüências de áudio devem atravessar a rede com um mínimo de informação perdida. A escala de tempo para a confiança é determinada pela duração da conversa. Durante este intervalo, o aumento da confiança da rede afeta a qualidade da voz e a perda de pacotes reduz a qualidade.

Quadros são perdidos em função de erros em rajada ou erros aleatórios em bits. Erros em rajada ocorrem tipicamente quando vários quadros são colocados em um pacote simples que é perdido ou quando vários pacotes consecutivos são perdidos. Por exemplo, um roteador pode, de forma não intencional, descartar pacotes de voz consecutivos quando uma interface congestionada encher o *buffer*. Os erros aleatórios em bits podem ser detectados na entrada de um decodificador provenientes de algum problema de software na estação receptora, uma vez que erros transitando na rede são detectados pelo algoritmo CRC. Pacotes com erros de CRC são descartados antes de serem enviados ao decodificador.

- Confiança entre sessões de chamadas

Confiança entre sessões de chamadas está relacionada com grau de serviço (GOS). Grau de serviço indica a fração das chamadas que foram completadas com sucesso em um tempo padrão pré-determinado. Este é um parâmetro para avaliar a confiança em redes telefônicas.

Uma rede deve prover alta confiança para garantir um alto GoS. Em uma rede não confiável, o grau de serviço é afetado quando mensagens de configuração de chamadas e liberação de recursos são perdidas. Uma mensagem do tipo *call setup* perdida deve resultar em uma chamada mal sucedida. Uma mensagem de liberação de recurso perdida reduz os recursos disponíveis para outras chamadas. Felizmente, mensagens deste tipo não são sensíveis ao atraso como as seqüências de áudio durante uma chamada, logo, mensagens perdidas podem ser retransmitidas para garantir um alto grau de serviço. Também pode-se considerar válido usar o RTP para mensagens de liberação de recursos e configuração de chamadas.

Quando se projeta uma rede, é importante observar alguns pontos relacionados à confiança:

- Usar componentes confiáveis, como cabos, *patch panels*, roteadores e *switches*;
- Onde possível, construir redundância e capacidade de tolerância a falhas, usando técnicas como por exemplo roteamento dinâmico e *spanning tree* para *switches* de redes locais;
- Implementar um programa de gerência da rede que possa reagir rapidamente em casos de quedas da rede e degradação do serviço e responda de forma proativa a possíveis ameaças que comprometam a utilização da rede.

3.4.2 ATRASO

Em uma conversa telefônica normal, cada parte fala enquanto a outra escuta. Quando alguém pára de falar e é estabelecido um silêncio por um certo período de tempo, uma outra pessoa pode falar. Se ninguém fala, a pessoal que falava inicialmente pode continuar a falar. Quando isto é mapeado para uma conversa telefônica com muito atraso, o efeito é semelhante a um baixo tráfego que mostra verde para carros andando na interseção de todas as direções. Todas as partes começam a falar ao mesmo tempo e então todos param de falar quando eles ouvem outros

falarem. Depois de uma pausa, todos começam a falar novamente parando apenas quando eles ouvem que todos estão falando novamente. Para a maioria das pessoas, até 250 ms é aceitável para prevenir colisões de fala, sendo que um atraso de até 400 ms é o valor máximo suportável. Pessoas acostumadas a fazer ligações internacionais via satélite terão uma tolerância a atraso maior, um fato que os projetistas de redes de pacotes de voz internacional devem apreciar.

É importante minimizar o atraso em uma rede de voz não apenas para evitar colisões, mas também para minimizar reflexões do sinal de áudio (eco) indesejadas. O degrau onde os ecos são prejudiciais em uma conversa telefônica é baseado na altura do eco e no atraso de cada eco. Quando se reduz o atraso final da rede, qualquer eco presente é menos prejudicial para os usuários.

Antes de poder minimizar o atraso da rede, deve-se entender as origens mais significativas do atraso. A figura 6 mostra o sinal de áudio através da rede e indica os maiores pontos onde o sinal é atrasado. Existem atrasos fixos e variáveis na transmissão de pacotes de voz. Os atrasos fixos estão relacionados com a compressão, transmissão e descompressão e os atrasos variáveis relacionados com as filas nos roteadores.

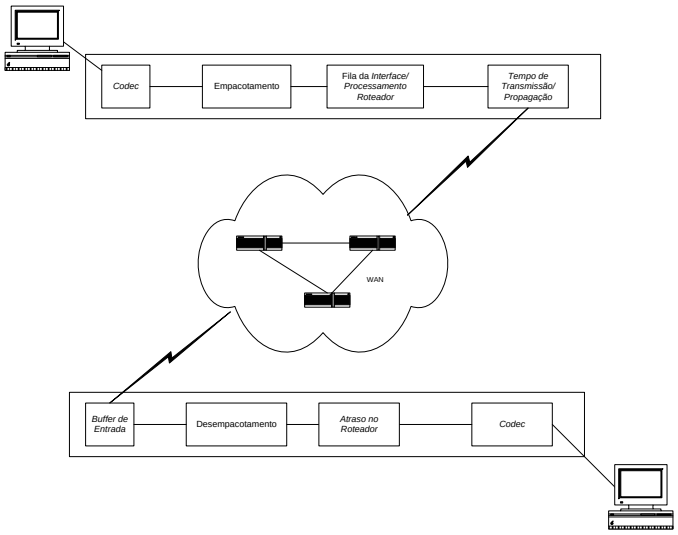

Figura 6. Origens do atraso

Estes pontos referem-se às seguintes origens de atraso:

- *Processamento do algoritmo de digitalização (Codec)*

O primeiro atraso substancial ocorre quando um codificador converte um sinal de áudio de analógico para digital comprimido. Em geral, os codificadores de áudio com menores taxas possuem alto atraso. Uma exceção é o algoritmo G.728 LD – CELP com uma taxa de 16 kbps e menos de 1 ms de atraso.

Este atraso depende da complexidade do algoritmo e do poder de processamento do processador de sinal digital (DSP). Este atraso é um parâmetro fixo que varia conforme o *Codec* escolhido.

Durante muitos anos o PSTN operou com o padrão ITU-T G.711. Contudo, em uma rede de comunicação de pacotes, e também em redes móveis sem fio, outros *codecs* poderão ser usados.

Telefones ou *gateway* envolvidos em estabelecer uma chamada estarão aptos a negociar qual *codec* usar, dentre um pequeno conjunto de *codecs* disponíveis.

Há muitos *codecs* disponíveis para digitalização de voz. O quadro 1 mostra algumas das características de uns poucos *codecs* padrão. Os *codecs* G.XXX são definidos pelo ITU. Os *codecs* ISXXX são definidos pelo ITA.

Quadro 1. Características dos vários Codecs usados em aplicações VoIP

Codec	**Algoritmo**	**Tamanho do frame/ *lookahead***	**Taxa**	**Comentários**
G.711	PCM	0.125 ms / 0	64 kb/s	Uso universal
G.722		0.125 ms / 1.5 ms	45,56 ou 64 kb/s	Banda larga
G.726	ADPCM	0.125 ms / 0	32 kb/s	Alta qualidade, baixa complexidade
G.728	LD-CELP	0.625 ms / 0	16 kb/s	Alta qualidade para conexões a cabo
G.729(A)	CS-ACELP	10 ms / 5 ms	8 kb/s	
.729(e)	CELP híbrido	10 ms / 5 ms	11.8 kb/s	Alta qualidade e alta complexidade
G.723.1(6.3)	MPC-MLQ	30 ms / 7.5 ms	6.3 kb/s	Origem da Vídeo conferência
G.723.1(5.3)	ACELP	30 ms / 7.5 ms	5.3 kb/s	Origem da Vídeo conferência
IS-127	RCELP	20ms /5 ms	4.2 kb/s	
AMR	ACELP	20 ms	Var. 4.75 – 12.2 kb	

A qualidade de uma chamada de voz através de um *codec* é freqüentemente avaliada por testes subjetivos, sob condições controladas, usando um grande número de ouvintes para determinar um MOS. Algumas características podem ser avaliadas variando as condições dos testes. Importantes características incluem o efeito do barulho do meio ambiente, o efeito da degradação do canal tal como perda do pacote quando interagindo com outras redes de transporte sem fio ou terrestre. A última característica é especialmente importante, desde que as redes VoIP terão de interagir com

redes de circuitos e redes sem fio usando *codecs* diferentes. A ordem dos *codecs* de taxa fixa listados na tabela, de um melhor para um pior desempenho é G.711, G.726, G.729, G.728, G.729, G.723.1. Resultados quantitativos são dados em (Perkins et al., 1997). Desde que a qualidade da voz é prejudicada quando usados *codecs* de baixa taxa de bit na linha de transmissão, a rede projetada deve esforçar-se para evitar *codecs* quando e onde quer que seja possível.

O melhor projeto de rede de pacote, codifica a fala próximo da pessoa que fala e a decodifica próximo do ouvinte. O encadeamento de *codecs* de fala, de baixa taxa de bit, assim como transcodificação da fala no meio da linha de transmissão, prejudica a qualidade da voz. A figura 7 (Goode, 2002) mostra os MOSs de alguns *codecs* com e sem encadeamento. Estes resultados são de (Perkins et al., 1997). Um MOS de 5 é excelente, de 4 é bom, de 3 é razoável, de 2 é ruim e de 1 é péssimo.

Observe que G.729x2 significa que uma fala codificada com um G.729 foi decodificada e depois recodificada com um G.729 antes de atingir o decodificador final. G729x3 significa que três *codecs* G.729 foram encadeados na linha de transmissão entre a pessoa que fala e o ouvinte.

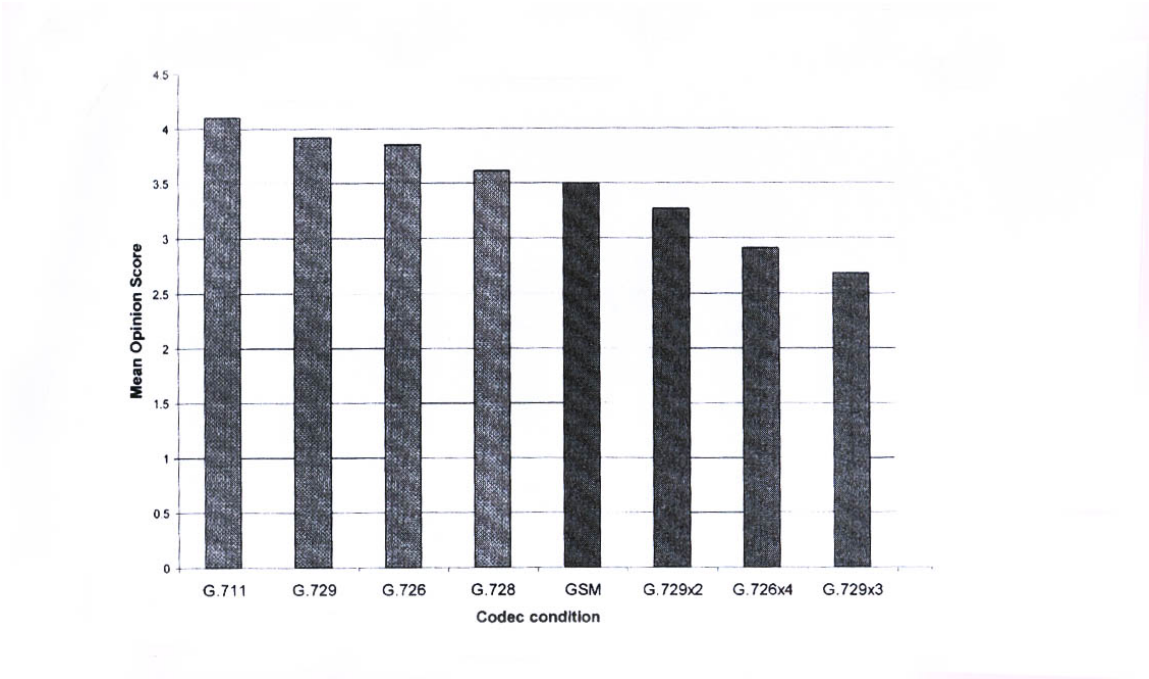

Figura 7. Efeito da concatenação de codecs no MOS

- Montagem do pacote

Aplicações típicas da Internet usam TCP/IP, enquanto que o VoIP usa RTP/UDP/IP. Embora o IP seja um protocolo de comunicação de rede sem conexão de melhor resultado, o TCP é um protocolo de transporte seguro que usa configurações e retransmissão para se certificar do recebimento do pacote. Usados conjuntamente, TCP/IP é um conjunto de protocolo de comunicações em rede de conexão orientada confiável. O TCP tem uma característica de ajuste de taxa que aumenta a taxa de transmissão quando a rede está descongestionada, mas reduz rapidamente a taxa de transmissão quando o *host* não recebe confirmação positiva do *host* destino. O TCP/IP não é adequado para comunicações em tempo real, tal como transmissão de fala, porque o aspecto de confirmação/ retransmissão levaria a excessivos atrasos. O UDP fornece serviços de entrega sem conexão não confiáveis, usando o IP para transportar mensagens entre

pontos. O RTP usado em conjunto com um UDP fornece funções de transporte na rede fim-a-fim, para utilização de transmissão de dados em tempo real, tais como áudio e vídeo, além de serviços de rede *unicast* e *multicast*.

O RTP não reserva recursos e não garante qualidade de serviço. Um protocolo associado RTCP permite o monitoramento de um link porém a maioria das utilizações do VoIP, oferece um fluxo contínuo de pacotes RTP/UDP/IP sem ter preocupação com a perda do pacote ou atraso.

Embora a transmissão possa ser barata em rotas principais em algumas partes do mundo assim com em muitas redes particulares, as facilidades de transmissão são suficientemente caras para merecer um esforço no uso eficiente da largura de banda. Este esforço começa com o uso de *codecs* de compressão da fala.

O uso de banda não larga leva a um longo atraso no empacotamento e ao uso dos mais complexos *codecs*. Um esforço de engenharia deve ser feito para que se consiga um atraso no empacotamento aceitável, um nível aceitável de complexidade do *codec* e uma exigência de capacidade de transmissão de chamada aceitável. Uma outra técnica para aumentar a eficiência da largura da banda é a supressão do silêncio. A qualidade da voz pode ser mantida pelo uso da supressão do silêncio. Se o *codec* receptor insere um ruído cuidadosamente projetado, durante cada período de silêncio. Por exemplo, o Anexo B da recomendação G.729 do ITU define pontos em que se deve medir mudanças do ruído e envia, a uma baixa taxa, informação para o receptor simulando o ruído do telefone emissor.

Os *frames* dos codificadores (que são as saídas do codificador) são colocados dentro dos pacotes RTP/UDP/IP em uma aplicação de voz sobre IP, dentro dos *frames* em aplicações de voz sobre frame relay e dentro de células em aplicações de voz sobre ATM. Quando um *frame* de um

codificador é transmitido em cada pacote , o atraso na formação do pacote não é uma origem significativa do atraso. Contudo, se vários *frames* são agrupados em um pacote simples, então o primeiro do grupo deve esperar enquanto *frames* adicionais são gerados para completar o pacote. Deve-se lembrar sempre deste fato ao aumentar o número de *frames* dos codificadores por pacotes, *frames* ou células para reduzir o consumo de largura de banda utilizada por cabeçalhos. Observe também que a estação receptora deve remover os cabeçalhos, o que representa um componente adicional de atraso.

A codificação e o empacotamento resultam em atrasos maiores do que os atrasos que os usuários conhecem em redes de comutação de circuitos. Como já citado, *codecs* de voz padrão são disponíveis para uma taxa de codificação de alcance aproximado de 64 a 5 kb/s. Geralmente, quanto mais baixa é a taxa de saída, mais complexos são os *codecs*. A montagem de um pacote envolve um esforço entre a eficiência da carga útil (carga útil / tamanho total do pacote) e o atraso no empacotamento (o tempo necessário para encher um pacote). Para IPv4, o cabeçalho do RTP/ UDP/IP é de 40 bytes. Uma carga útil de 40 bytes significaria 50% de eficiência na carga útil. A 64 kb/s ele leva apenas 5ms para acumular 40 bytes, mas a 8 kb/s ele leva 40ms para acumular 40 bytes. Um atraso no empacotamento de 40ms é significativo, e, muitos sistemas VoIP usam pacotes de 20 bytes apesar da baixa eficiência, quando usam *codecs* de baixa taxa de bit.

Para uma fala contínua, a exigência para a capacidade de transmissão da chamada BW(bandwidth), em kb/s, está relacionada ao tamanho do cabeçalho H (em bits), a taxa do *codec* R (em kb/s) e o tamanho da carga útil S do exemplo (em mili-segundos) como:

$$BW = R+H/S$$

Existem alguns algoritmos de compressão de cabeçalho que melhorarão a eficiência da carga útil (Degemark et al., 1999) , (Engan et al, 1999). O cabeçalho RTP/UDP/IP de 40 bytes pode ser comprimido para 2-7 bytes. Um cabeçalho caracteristicamente comprimido é de quatro bytes, incluindo um CHECKSUM de dois bytes. Em uma rede IP, a compressão do cabeçalho deve ser feita link por link, porque o cabeçalho deve ser restaurado antes que um roteador possa escolher uma interface disponível. Por isso, essa técnica é mais adequada para links de acesso de baixa velocidade.

Uma exigência mais baixa de largura de banda leva a um longo atraso no empacotamento e a *codecs* mais complexos. Um esforço de engenharia deve ser feito para obter um atraso aceitável no empacotamento, uma complexidade do *codec* aceitável, e uma exigência de largura de banda aceitável.

- Processamento e Encaminhamento nos roteadores

Depois que os frames dos codificadores são colocados em pacotes e estão prontos para transmissão, eles devem esperar um tempo substancial em um *buffer* de uma *interface* lógica de entrada e saída de um roteador. Enquanto pacotes esperam no *buffer*, uma política administrativa arbitra a ordem na qual os pacotes serão transmitidos. Para reduzir o atraso dos pacotes de voz, a política da fila deve mover os pacotes de voz para frente da fila da interface lógica. Os pacotes no seu trajeto passam por uma série de roteadores, que integram a rede de comutação de pacotes.

Roteadores inserem atrasos que referem-se ao atraso de enfileiramento explicado acima e o atraso de processamento decorrente da necessidade de análise dos pacotes para funções como encaminhamento e condicionamento/ classificação do tráfego.

Este é um parâmetro que afeta o atraso e que pode ser alterado através da implementação de políticas de priorização de tráfego para reduzir o atraso na fila e de algoritmos que implementam QoS, como por exemplo o *Priority Queing* e o WFQ.

- Tempo de Transmissão e Propagação do Sinal

Este atraso refere-se ao tempo de transmissão e ao tempo de propagação do sinal no meio de transmissão. Este é um parâmetro fixo, dependente do meio de transmissão utilizado pela rede.

3.4.3 MINIMIZANDO O ATRASO

Para reduzir o atraso em uma conversa telefônica via rede, deve-se considerar os seguintes pontos:

- Trabalhar para melhorar o caminho físico de transmissão da rede do fornecedor de serviço, isto é definir caminhos otimizados como preferenciais;
- Implementar política que priorize o tráfego de voz para reduzir o atraso da fila, através de algoritmos que implementam QoS, como por exemplo o *Priority Queuing*, CB-WFQ (*Class Based Weighted Fair Queuing*), WFQ (*Weighted Fair Queuing*) ou o WRR (*Weighted Round Robin)*. É importante provisionar uma quantidade de largura de banda suficiente para que a priorização do tráfego de voz seja efetivada;
- Utilizar tecnologias para reduzir o atraso de espera nas filas, como por exemplo quebrar os pacotes grandes em fragmentos e permitir que pacotes de voz sejam intercalados entre os fragmentos de um pacote grande. Isto pode ser usado em roteadores com interfaces lentas;
- Selecionar um codificador que introduz um atraso mínimo. *Hardwares* diferentes podem suportar um conjunto limitado de codificadores.

Se o tamanho do *buffer* de saída é estaticamente configurado, pode-se reduzir este tamanho para minimizar o atraso. Deve-se considerar esta opção quando a rede tem uma pequena variação do atraso, como por exemplo quando se utiliza canais E1 ou T1 com um alto atraso de transmissão.

Algumas opções para reduzir o atraso do tráfego de voz podem prejudicar a performance de outros tráfegos ou pode comprometer outros aspectos da qualidade de voz, como o *jitter* e o percentual de perdas. É importante considerar os efeitos da seleção do codificador, fragmentação do pacote e tamanho do *buffer* de saída. Não se deve considerar estes fatores isoladamente.

3.5 VARIAÇÃO DO ATRASO (*JITTER*)

Variação do atraso ou *jitter* é um fator importante que afeta os pacotes de voz. Uma conversa telefônica com atraso pode ser desagradável e a informação do áudio pode tornar-se ininteligível. O *jitter* pode fazer com que uma seqüência de áudio fique completamente indecifrável.

O espaço de tempo em que os pacotes são transmitidos para o destino, normalmente, são iguais, enquanto o intervalo de tempo em que os pacotes são recebidos pode variar. Isto ocorre porque pacotes distintos podem sofrer atrasos distintos enquanto são transmitidos na rede.

A figura 8 (Martins, 1999) ilustra o efeito do *jitter* entre a entrega de pacotes na origem e o seu processamento no destino. Observe que o *jitter* causa não somente uma entrega com periodicidade variável (*Packet-Delay Variation*) como também a entrega de pacotes fora de ordem.

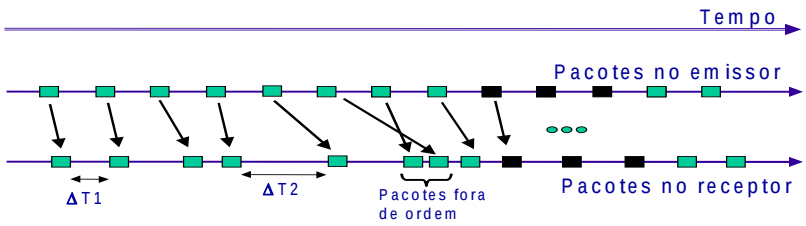

Figura 8. Efeito do jitter sobre a entrega dos pacotes

Em princípio, o problema dos pacotes fora de ordem poderia ser resolvido com o auxílio de um protocolo de transporte como o TCP (Stevens, 1994) que verifica o sequenciamento da mensagens e faz as devidas correções. Entretanto, na prática tem-se que aplicações VoIP optam por utilizar o UDP (Stevens, 1994) ao invés do TCP pela maior simplicidade e menor *overhead* deste protocolo. Nestes casos, o problema de sequenciamento deve ser resolvido por protocolos de mais alto nível normalmente incorporados à aplicação como, por exemplo, o RTP (Chwan e Irwin, 1998).

O *jitter* introduz distorção no processamento da informação na recepção e deve ter mecanismos específicos de compensação e controle que dependem da aplicação em questão. Genericamente, uma das soluções mais comuns para o problema consiste na utilização de *buffers* (técnica de "*buffering"*).

O receptor *gateway* ou telefone deve compensar a variação do atraso com buffer, a qual impõe um atraso nos pacotes iniciais e transmite os pacotes posteriores com menos atraso. Desta forma, a voz decodificada flui para o receptor a uma taxa estável. Qualquer pacote que chegue, após o enchimento do buffer, é descartado. Desde que se deseja uma baixa perda de pacote, o atraso do buffer de *jitter* é o atraso máximo que se deve esperar. Este atraso do buffer deve ser incluído no total do atraso fim-a-fim que o ouvinte experimenta durante uma conversa usando VoIP.

A qualidade da voz é extremamente dependente do tempo em que são disponibilizados os sons para serem ouvidos. Se o intervalo entre *frames* no transmissor for diferente do receptor, então o receptor não ouvirá uma representação fiel do som original. Para obter uma noção dos efeitos do *jitter*, pode-se imaginar um CD que é tocado enquanto alterna pausas, ficando o som com interrupções.

3.5.1 *BUFFERS* DE SAÍDA

Para ajudar a combater os efeitos do *jitter*, sistemas de voz usam *buffers* de saída. A idéia é utilizar um *buff*er que armazene um estoque de *frames* do codificador por algum tempo para garantir que os *frames* possam ser liberados a uma taxa constante. O número de *frames* no *buffer* reduz quando nenhum pacote de voz chega na rede e cresce quando uma rajada de pacotes chegam. Apesar do número de *frames* no *buffer* estar sempre mudando, desde que o *buffer* não fique completamente cheio nem completamente vazio, os *frames* são descarregados a uma taxa constante, o que garante uma boa qualidade do sinal de áudio.

Aumentar o tamanho do *buffer* (para acomodar mais *frames*), cria uma maior resistência para o *jitter*, mas o tempo requerido para armazenar os *frames* adicionais introduz mais atraso no sistema. Sites da Web, através de *softwares* como por exemplo o *Microsoft Media Player*, fazem com que usuários esperem 5, 10 , 15 ou mais segundos antes de ouvir um som que começará a tocar. Não há uma aplicação em tempo real, logo eles podem manter um *buffer* por um longo período de tempo. O tempo extra garante que o som seja ouvido como uma seqüência constante de áudio. Para pacotes de telefonia, o som pode ser armazenado por pouco tempo para manter as características de tempo real. Logo, os *buffers* são tão pequenos que o *jitter* torna-se uma ameaça.

A figura 9 ilustra como o *jitter* pode distorcer a saída de um sinal de áudio. Na primeira imagem, o *buffer* está vazio e a seqüência de áudio transforma-se em um silêncio.

Na Segunda imagem, o *buffer* está completamente cheio e os pacotes que chegam são descartados. O sinal de áudio fica sem a representação dos pacotes perdidos.

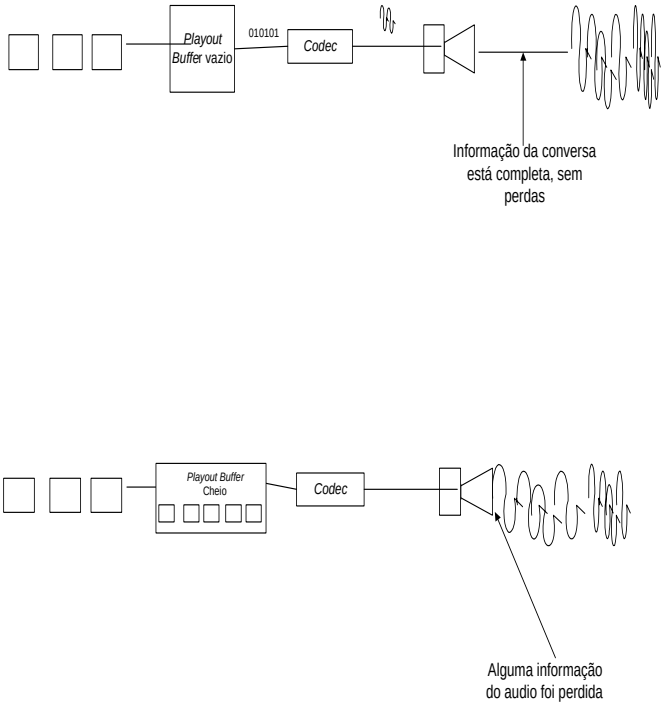

Figura 9. Distorção do sinal de áudio

Por causa do *overflow* do *buffer*, geralmente o resultado é a perda de várias amostras de *frames* que prejudica especialmente algoritmos de baixa taxa, como por exemplo o G.723.1 e o G.729.

3.5.2 ORIGENS DO *JITTER*

Jitter é uma característica inerente a redes de pacotes porque não há uma transmissão síncrona fim a fim como em redes baseadas em circuitos. *Jitter* ocorre em qualquer ponto em uma rede de pacotes onde existem *buffers* de *interfaces*, tanto em redes públicas quanto privadas. Os pontos abaixo introduzem *jitter*:

- Fila de Interfaces

Buffers são necessários para acomodar a rajada característica de redes de dados. As filas das interfaces são *buffers* que absorvem pacotes e os transmitem quando uma interface não está congestionada. Quando uma interface pode transmitir pacotes mais rápido que uma origem,

pacotes não precisam esperar em um *buffer*. Se mais pacotes são oferecidos que a interface pode transmitir em um momento, os pacotes são armazenados em um *buffer* até que a interface esteja pronta para transmiti-los. Grandes *buffers* acomodam tráfego em rajada e reduz o número de pacotes descartados em função de um *overflow*, mas eles também introduzem atrasos variáveis e, consequentemente, *jitter*. Todo pacote tem no mínimo um *buffer* de entrada e um *buffer* de saída que contribui para o atraso e o *jitter* fim-a -fim.

Se políticas de fila não são implementadas para dar tratamento preferencial a pacotes de voz, então estes pacotes sofrem muito atraso enquanto esperam nas filas das interfaces que outros pacotes sejam transmitidos. Como a fila de uma interface pode estar vazia ou cheia ou de qualquer outra forma quando um pacote de voz chega na fila, os pacotes podem esperar um tempo qualquer para que sejam transmitidos, desde nenhum tempo a vários segundos. Cada fila de interface impõe a sua variação de atraso, como pacotes atravessam vários *switches*, logo eles sofrem uma enorme variação do atraso durante o trânsito. Deve ficar claro que a quantidade de *jitter* causado pelo *buffer* de uma interface gera a necessidade de se ter um *buffer* de saída ou *playout buffer* para controlar o *jitter*. Por esta razão, políticas de fila são essenciais para mover pacotes de voz para o início das filas.

- Serialização

O atraso de serialização é crítico para interfaces de baixa velocidade. A quantidade do atraso de serialização que um pacote de voz sofre em um *buffer* de transmissão é variável e limitado ao pacote de maior tamanho na rede. Um pacote de voz que é movido para frente de uma fila de interface pode ser instantaneamente transmitido se não houver outros pacotes no meio de transmissão ou ele pode ter de esperar em um *buffer* atrás de um pacote de 1500 bytes que tenha

iniciado a transmissão. Se a interface transmite a 64 kbps, então o pacote deve esperar 188 ms. Um *buffer* de saída de pacotes de voz pode ficar vazio apenas devido à variação do atraso em uma interface. O caminho típico de áudio atravessa no mínimo algumas interfaces que introduzem atrasos variáveis.

A solução para o problema descrito acima é quebrar os pacotes grandes em fragmentos e permitir que pacotes de voz sejam intercalados entre os fragmentos de um pacote grande.

- Serialização e *Buffers* da nuvem WAN

Considerando atrasos dos *buffers* das interfaces e da serialização nos componentes da rede, deve-se considerar também estes fatores nas redes gerenciadas por um prestador de serviço. Para canais do tipo E1/T1, não é preciso se preocupar com *buffers* e serialização na transmissão da rede, porque há uma seqüência de informações síncronas através do circuito com um atraso fixo e constante. Entretanto, quando se utiliza circuitos frame relay, então a transmissão é feita multiplexando o tráfego de vários clientes e um tráfego específico terá de esperar em vários *buffers* para atravessar a rede. Utilizando circuitos ATM, pode-se controlar melhor o atraso, uma vez que pode-se ordenar circuitos que possuem requisitos de atraso e *jitter* específicos. *Frame Relay* não oferece garantias referentes a atrasos e *jitter*.

3.5.3 MINIMIZANDO O *JITTER*

Segue algumas estratégias que visam minimizar e gerenciar o *jitter*:

- Implementar uma política de fila que priorize o tráfego de voz sobre outros tráfegos de dados. Isto reduz a variação do atraso no *buffer* da interface, mas não reduz o atraso de serialização e as variações na forma como plataforma diferentes de hardware utilizam os *buffers* de transmissão;

- Deve-se reduzir o tamanho máximo de um pacote para minimizar a variação do atraso de serialização. Para aplicações de voz sobre *frame relay*, a taxa de processamento da interface deve ser reduzida para o valor do CIR do PVC;
- Evitar caminhos paralelos de transmissão com diferentes características de atraso. Pacotes que chegam com atrasos diferentes podem sobrecarregar os *buffers* e causar *overflow*;
- Aumentar o tamanho dos *buffers* de saída para acomodar alto atraso. Um alto atraso geral é melhor que uma seqüência de áudio com *jitter*.

3.5.4 LARGURA DE BANDA (VAZÃO)

O planejamento de capacidade de redes que transportam tráfego em tempo real é algo diferente do planejamento da capacidade das redes tradicionais que transmitem apenas dados. Embora os requisitos de largura de banda continuem a ser baseados em taxas de picos, os intervalos sobre os quais as taxas de pico são medidas devem ser mais curtos. Considerando uma rede projetada para uma taxa média de tráfego de 200 kbps por hora, um circuito WAN de 256 kbps aparentemente acomodaria bem o tráfego. Considerando uma outra situação em que durante uma hora completa, o nível de tráfego pode repetidamente chegar a 300 kbps em intervalos de minutos, seguido de um uso menor por algum tempo. Neste cenário, usuários podem ou não queixar-se da performance da rede. Eles definitivamente reclamarão da performance da rede porque eles possuem altas expectativas da qualidade da voz e disponibilidade da rede. Durante os momentos de alta utilização da rede, ocorrerá *overflow* e a qualidade da voz sofrerá com a perda ou atraso dos pacotes de voz.

Considerações econômicas são freqüentemente ouvidas sobre decisões de aumentar a largura de banda. Uma forma de evitar a atualização da largura de banda é otimizar a eficiência do tempo de

resposta através da priorização do tráfego, minimizando atualizações de roteamento e reduzindo a quantidade de *overhead* dos pacotes. Configurações avançadas de roteadores podem ser uma boa ferramenta para evitar atualizações de largura de banda. O custo para suportar configurações complexas de roteadores é algo aceitável. Em alguns casos, o dinheiro gasto com atualização de largura de banda deve ser gasto com recursos adicionais para projetar e manter a complexidade da rede. Em geral, o ideal é comparar o custo total de cada estratégia para tomar decisões. Quando o os custos de largura de banda são muito altos, como circuitos internacionais, configurações complexas de roteadores são facilmente justificadas. Para um canal do tipo E1 ou T1, talvez uma largura de banda adicional possa ser mais barata que suportar roteadores com configurações complexas. Complexidade nem sempre deve ser evitada. É importante considerar todos os fatores quando for tomar uma decisão.

3.5.5 MINIMIZANDO O USO DA LARGURA DE BANDA

Os seguintes passos otimizarão a vazão de dados e voz para uma determinada porção de largura de banda da rede.

- Usar um codificador de voz com alta taxa de compressão. Por exemplo, o G.729 (8 Kbps) e o G.723.1 (6,3 kbps ou 5,3 kbps);
- Minimizar o tamanho dos cabeçalhos dos pacotes que encapsulam dados;
- Reservar largura de banda para aplicações críticas e de tempo real;
- Monitorar a rede e projetar passos para ajudar a planejar atualizações de banda, considerando o tempo gasto com a instalação dos circuitos.

4 COMPORTAMENTO DOS NÓS DIFFSERV EM UMA APLICAÇÃO VoIP

4.1 A ESTRATÉGIA DIFFSERV

O processo de evolução da infra-estrutura de comunicações passa por uma expectativa grande relacionada à adoção da tecnologia do núcleo que será usada para o encaminhamento das informações dentro da filosofia de uma rede digital de serviços integrados. Atualmente, analisa-se firmemente a utilização do IP como tecnologia de núcleo que suportará o tráfego integrado.

Devido à penetração no mercado de tecnologias, como a tecnologia de VoIP, a demanda por qualidade de serviço vem crescendo muito e o *Diffserv* surge com o objetivo de atender a esta demanda dentro do próprio ambiente IP.

No modelo de serviços diferenciados *Diffserv* (Blake *et al*.,1998), pacotes são marcados em um campo específico para criar várias classes de pacotes. Pacotes em classes diferentes recebem serviços diferentes. Algumas classes são tratadas de forma privilegiada em relação a outras. O campo utilizado é o DS (*Differentiated Service*) que, em conjunto com uma estratégia de encaminhamento de pacotes, denominada de *Per-Hop-Behavior* ou PHB, cria classes de serviços com tratamento diferenciado.

Para um cliente utilizar o serviço *Diffserv* do seu provedor de serviço, ele precisa estabelecer um contrato de tráfego – SLA (*Service Level Agreement*) que especifica as classes de serviço suportadas e a quantidade de tráfego alocada a cada classe.

A estrutura do *Diffserv* é definida por um conjunto de roteadores que formam um domínio. Este domínio possui os roteadores de limiar ou fronteira que estão oferecendo o serviço diretamente ao usuário e os roteadores do núcleo que provêem o encaminhamento dos pacotes pelo núcleo.

A marcação do campo DS em cada pacote, para indicar o perfil do serviço desejado, é marcado pelo roteador de fronteira. As regras de suavização e classificação são derivadas do SLA. Sendo assim, o *Diffserv* pode ser relacionado a duas atividades básicas: caracterização do tráfego que está entrando na rede e encaminhamento deste tráfego pelo núcleo da rede.

Os critérios de QoS são atendidos através de um mecanismo de avaliação e suavização de tráfego nos nodos da fronteira e de esquemas de encaminhamento de tráfego no núcleo do domínio.

O nodo da fronteira é responsável pela associação do tráfego do usuário a uma classificação pré-estabelecida. Uma vez classificado, o tráfego é encaminhado pelo núcleo em função de uma política denominada como PHB (*Per hop behavior*). Exemplos do PHB são o AF Encaminhamento Assegurado (AF- *Assured Forwarding*) e o Encaminhamento Expresso (EF– *Expedited Forwarding*).

O encaminhamento expresso, EF-PHB, garante ao usuário uma banda, enquanto o encaminhamento assegurado, AF-PHB, também provê aos usuários uma banda, mas não garante a sua disponibilidade. Entretanto, com base em sua filosofia estatística de provisionamento pode obter resultados interessantes em relação ao aproveitamento da banda com atrasos e perdas mínimas.

O serviço de Encaminhamento Assegurado não se baseia em uma garantia estrita, mas em uma expectativa de serviço que será obtida por um determinado tráfego quando existem momentos de congestionamento. O controle de admissão exerce papel importante neste modelo, uma vez que é encarregado de garantir a existência dos recursos necessários durante todo o percurso da origem ao destino. Um perfil associado a cada tráfego define o serviço esperado por ele. Neste tipo de

serviço, existe uma garantia que os pacotes marcados serão encaminhados com uma alta prioridade, contanto que o tráfego não exceda a taxa contratada, definida pelo perfil de serviço. No entanto, é permitido que um usuário exceda um perfil contratado, sabendo que o tráfego em excesso não será entregue com a mesma probabilidade do tráfego que está dentro do perfil.

O AF tem o objetivo de oferecer serviços confiáveis a partir de seus provedores de serviços, mesmo quando ocorre congestionamento. Pode ser compreendido através de 3 etapas:

1) A classificação é feita nos nós de entrada. Se o tráfego não exceder os parâmetros de tráfego especificados durante a negociação do SLA, eles são considerados dentro do perfil especificado (*in profile*), caso contrário são considerados fora do perfil (*out profile*). Logo, os pacotes são analisados, marcados para serem encaminhados. A classificação do pacote em *in* ou *out* é feita através de um mecanismo chamado de *token bucket*;

2) Todos os pacotes in e out são colocados em uma fila para serem encaminhados;

3) A fila é gerenciada por um esquema chamado RED com *in* e *out* ou RIO. O RED é um algoritmo de descarte de pacotes de cada classe com o objetivo de controlar o congestionamento. Desta forma, ele descarta pacotes com certa probabilidade sempre que a fila excede um determinado tamanho e evita a ocorrência de congestionamento, quando teria que descartar pacotes com altos requisitos de QoS, por exemplo;

4) Os pacotes são escolhidos efetivamente para serem enviados através de um algoritmo de prioridade. O algoritmo utilizado é o WRR *(Weighted Round Robin*) que é usado na interface de saída dos equipamentos como roteadores para escalonamento de filas e envio de pacotes.

A seguir será detalhado o funcionamento dos algoritmos da estratégia *Diffserv* utilizada.

4.2 FUNCIONAMENTO DO *TOKEN BUCKET*

O *Token Bucket* consiste em um balde de capacidade B que representa a capacidade de armazenar fichas (*tokens*). As fichas são repostas através de uma taxa constante de reposição de créditos "R". Quando um pacote chega , verifica-se se há alguma ficha no balde, se existir, o pacote é marcado como *in* e uma ficha é consumida, caso não exista nenhuma ficha no balde, o pacote é marcado como *out*.

O algoritmo a seguir ilustra o funcionamento do *token bucket*.

Utilizando a variável C para armazenar o número de *tokens* no balde e B para armazenar o tamanho do balde. Inicialmente, pode-se considerar que C= B (balde está cheio).

Isto é , a cada 1/R segundos, sendo R a taxa de geração de *tokens* no balde, tem-se:

```
Se   C <  B          {Até encher o balde de tokens}

  C = C +1

Esperapacote()

{Chega um pacote de tamanho  L}

Se  L  < =   C

  C = C-1  { pacotes são aceitos, isto é são marcados como in }

Se não

  { Pacote é marcado como out}
```

O *burst* pode ser representado pela seguinte fórmula: B + (T *R) em bytes e em T segundos.

A figura 10 ilustra o funcionamento do *token bucket*.

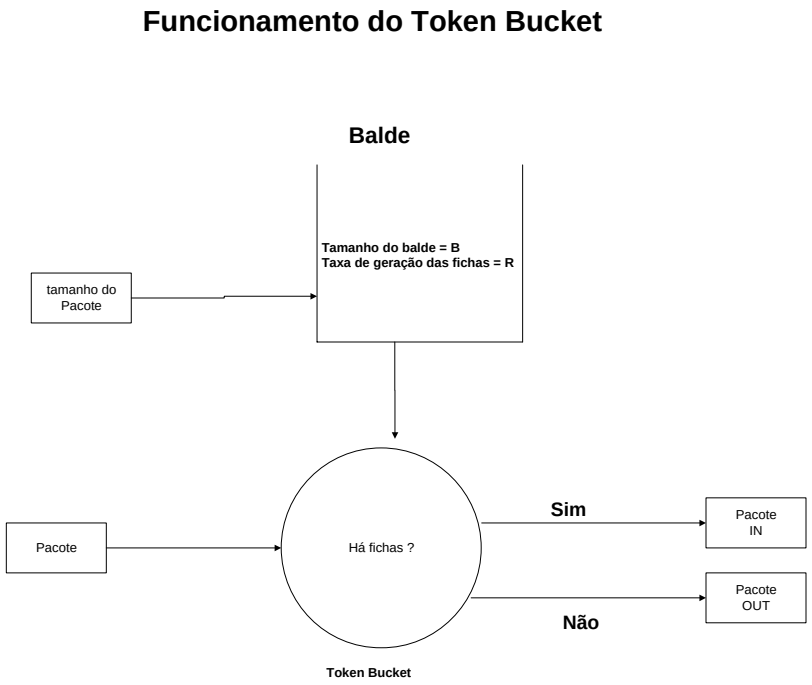

Figura 10. Funcionamento do Token Bucket

4.3 FUNCIONAMENTO DO ALGORITMO RED

O RED é um algoritmo de gerenciamento de filas utilizado em uma implementação AF-PHB do sistema *Diffserv* 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, e provavelmente descartar 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 no 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 encontrar 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 é muito variável (*bursty*), o *min_thresh* deve ser suficientemente grande para manter uma boa utilização do enlace. E também, 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, onde trafega 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 de rajada, 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 algoritmo RED possui dois algoritmos separados. O algoritmo para computar o tamanho médio da fila e o algoritmo para calcular a probabilidade de marcação de pacotes para descarte. O objetivo deste algoritmo é marcar pacotes em intervalos regulares de forma equilibrada, a fim de evitar ajustes e sincronizações globais e 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 pela estimação do 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.

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 com in e out).

O funcionamento do RIO é o mesmo do RED, sendo que ele faz uma diferenciação 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 e OUT. Consequentemente o tamanho médio da fila dos pacotes OUT será maior e maior será a probabilidade da mesma ser maior que um limite superior, por exemplo, 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 esperança de controlar o congestionamento. Fica caracterizado dessa forma uma certa prioridade para os pacotes IN.

Segue abaixo o resumo do algoritmo RIO descrito anteriormente:

Chega novo pacote

Se IN lança na fila IN

Calcula o tamanho médio da fila para pacotes IN

Se o tamanho médio da fila for menor que min_IN

    o pacote é colocado na fila

Se o tamanho médio da fila for maior que Max_IN

     todos os pacotes são marcados para serem descartados

Se   Min _IN < tamanho médio da fila é <   Max_IN

   Calcula p dependendo do tamanho médio da fila

Se p > pmaxIN todos os pacotes são marcados para serem descartados

Senão

    É marcado para descarte com a probabilidade p

Chega novo pacote

Se OUT lança na fila OUT

Calcula o tamanho médio da fila levando em consideração todos dos pacotes (IN +OUT)

Se o tamanho médio da fila é menor que min_OUT

O pacote é colocado na fila

Se o tamanho médio da fila é maior que Max_OUT

todos os pacotes são marcados para serem descartados

Se Min _OUT < tamanho médio da fila é < Max_OUT

Calcula p dependendo do tamanho médio da fila

Se p > pmaxOUT todos os pacotes são marcados para serem descartados

Senão

É marcado para descarte com a probabilidade p

O algoritmo RED é um algoritmo que nenhum pacote é marcado para descarte caso o tamanho médio da fila seja inferior a um limiar inferior e todos os pacotes são marcados para descarte caso o tamanho médio da fila exceda uma limiar superior. Se todos os pacotes marcados são de fato descartados, isto assegura que o tamanho médio da fila nunca exceda significativamente o limiar superior.

4.4 FUNCIONAMENTO DO WRR

Este algoritmo realiza o escalonamento de filas através da escolha de pacotes que deverão ser enviados baseado na prioridade dos pacotes. Ele considera o peso das filas como parâmetro de prioridade. Considerando como exemplo, duas filas com prioridades N e M, para cada N pacotes, o WRR enviará M.

Utilizando WRR, a cada fila é atribuído um peso. Este peso é usado quando ocorre congestionamento na porta para atribuir um peso alto para o tráfego de alta prioridade sem esquecer o tráfego de baixa prioridade. Os pesos garantem uma maior ou menor porção da largura de banda para o tráfego enfileirado de acordo com a prioridade atribuída a cada tráfego. Quando a porta não está congestionada e quando há largura de banda suficiente, todas as filas são igualmente atendidas, caso contrário são atendidas primeiro as filas de maior peso, isto é, de maior prioridade.

5 PROPOSTA DE SIMULAÇÃO

Este capítulo tem como objetivo descrever os parâmetros de configuração, a topologia, o modelo de rede e a ferramenta utilizado durante as simulações.

A proposta de simulação consiste em um modelo de rede *Diffserv* que executa uma aplicação VoIP utilizando a topologia descrita abaixo. A ferramenta de simulação em uso neste trabalho é o NS-2 (*Network Simulator*) (NS, 2002) que é um simulador de eventos discretos desenvolvido em C++. A topologia e o modelo de simulação foram criados utilizando a linguagem OTcl.

5.1 CONFIGURAÇÃO DE PARÂMETROS *DE SIMULAÇÃO*

- **Configuração das Fontes**

As fontes ON- OFF são as mais indicadas para modelar o tráfego de voz, uma vez que simulam bem os eventos de uma conversa real, alternando momentos de conversa (ON) e silêncio (OFF). A distribuição do tamanho do período de transmissão de dados e do período de silêncio é bastante próximo ao de uma função exponencial, como já relatado em (Daigle e Lanford, 1998) e demonstrado através de um modelo matemático.

Em condições semelhantes, o serviço de encaminhamento assegurado pode suportar um maior número de fluxos ON- OFF que fluxos CBR com qualidade final similar. As fontes ON-OFF podem aproveitar com mais eficiência a reserva de uma banda passante extra acima do solicitado, o que representa um ganho de desempenho para tráfegos ON-OFF. Confirmando dessa forma, uma maior eficiência de fontes ON-OFF na modelagem de tráfegos de voz.

Durante os períodos ativos, a fonte envia pacotes em intervalos regulares, com o tempo de interchegada constante. Durante os momentos ON, pacotes com tamanho fixo (definido no

parâmetro "*Packetsize")* são enviados um a um, sendo gerados a uma taxa constante definida no parâmetro "*rate*" durante um tempo médio definido no parâmetro "*burst*", de forma alternada com momentos de silêncio cujo valor médio está definido no parâmetro "*idle*". Os tempos ON e OFF são distribuídos exponencialmente.

Para realizar a modelagem de voz usando as fontes ON-OFF torna-se necessário a configuração dos seguintes parâmetros:

- ✓ *Burst* – determina o tempo médio em que a fonte envia pacotes;
- ✓ *Idle* - determina o tempo médio em que a fonte não envia pacotes, isto é, o tempo de silêncio;
- ✓ *Packetsize* – Determina o tamanho dos pacotes enviados pela fonte;
- ✓ *Rate* – representa a taxa de dados enviada durante os momentos de atividade da fonte.

Quando as fontes ON-OFF estão no estado de atividade "ON" é gerado um tráfego de voz a um pico de 64 kbps (correspondente ao padrão PCM). Os períodos de atividade são determinados por uma variável aleatória com distribuição exponencial de média 400 ms. Os períodos de silêncio OFF também são distribuídos exponencialmente por uma variável de 600 ms (Rezende e Ziviani, 1999). Estes valores resultam em uma taxa média de tráfego de 25,5 kbps. Os pacotes utilizados possuem tamanho de 80 octetos.

As fontes ON-OFF, entretanto, não são indicadas para a modelagem do tráfego FTP, uma vez que este tipo de tráfego é caracterizado como um tráfego que envia pacotes grandes de forma constante, só sendo interrompido pelo final da transmissão ou pelo atraso no recebimento do reconhecimento (ack). O que se pode alterar é um liga/desliga que ativa e interrompe a geração de pacotes FTP ou também o número de pacotes FTP transmitidos. O único parâmetro que será

configurado é o tamanho do pacote (“TCPPacketSize”), uma vez que este é o único parâmetro disponível para configuração das fontes deste tipo de tráfego.

- **Configuração do *Token Bucket***

O tráfego em rajada gerado pelas fontes ON-OFF é suavizado na interface de saída do nó onde a fonte está localizada. A suavização deve ser realizada pelo algoritmo *token bucket*, onde a profundidade do balde é mantida em 1 pacote (Rezende e Ziviani, 1999).

A taxa média de transmissão da fonte representa a taxa de reposição de créditos dos suavizadores.

Os parâmetros que devem ser configurados no *token bucket* são:

- ✓ CIR – representa a taxa de reposição de créditos, isto é, representa a taxa de preenchimento do balde que deve ser igual à taxa esperada pelo usuário, ou seja, o perfil de serviço para o qual o AF deve fornecer garantias. A taxa de transmissão das fontes deve representar a taxa de reposição de créditos dos suavizadores, com o objetivo de preencher toda a banda designada ao tráfego de voz;
- ✓ CBS – representa o tamanho do balde em bytes , isto é, o tamanho da rajada. Para o tráfego de voz, a profundidade do balde é mantida em 1 pacote, uma vez que este tipo de tráfego gera tráfego a uma taxa constante e não em rajada como o FTP.

- **Configuração do RED**

Para o funcionamento do RED, torna-se necessário a configuração dos parâmetros listados abaixo. É importante ressaltar que os parâmetros devem ser configurados com valores que reflitam a realidade de uma rede que possui tipos de tráfegos diferentes, como por exemplo, um tráfego ON-OFF competindo com um tráfego FTP.

- ✓ MeanPktSize – Este parâmetro é usado para atualização do tamanho médio da fila, uma vez que é utilizado no cálculo do tamanho médio da fila depois de um período ocioso. Representa o tamanho médio do pacote em bytes. Foi considerado o tamanho do pacote UDP por estimar que serão os pacotes que serão enviados em um maior número em função da maior prioridade. Pode-se usar também um tamanho médio entre o tamanho dos pacotes UDP e FTP, ou seja, (1500 +576)/2;
- ✓ Numques = 2 - representa o número de fila físicas existentes. No caso, são duas: uma fila para o tráfego UDP e uma fila para o tráfego FTP;
- ✓ Setnumprec = 2 - representa o número de níveis de precedência de descarte, isto é *in* e *out*.

Parâmetros a serem configurados para o RIO:

- ✓ Maxdropprob – representa a máxima probabilidade de descarte de um pacote, quanto maior este valor mais agressivo será o algoritmo de descarte. Para o tráfego *in* será usado um valor menor que para o tráfego *out*, assim como para o tráfego de voz será usado um valor menor que para o tráfego FTP;
- ✓ Min_Thresh – representa o tamanho da fila em número de pacotes a partir do qual o pacote pode ser descartado com probabilidade P;
- ✓ Max_Thresh – representa o tamanho da fila em número de pacotes a partir do qual o pacote sempre será descartado.

A discriminação contra pacotes OUT é criada através de uma escolha cuidadosa de parâmetros. Um pacote é mais agressivo no descarte de pacotes OUT de três formas:

1) Ele descarta pacotes OUT mais cedo que pacotes IN, pela escolha de um min_out menor que um min_in;

2) Para evitar congestionamento, ele descarta pacotes out com maior probabilidade escolhendo um Pmaxout maior que um Pmaxin;

3) Ele descarta pacotes out muito antes que pacotes in pela configuração de um maxout menor que maxin.

- **Configuração do WRR**

O algoritmo WRR enviará primeiro os pacotes da fila de maior prioridade. Esta prioridade é definida através do parâmetro peso que deve ser configurado para cada tipo de tráfego. Quanto maior o valor, maior será a prioridade de atendimento da fila.

5.2 DESCRIÇÃO DA FERRAMENTA UTILIZADA

A ferramenta de simulação utilizada foi o Network Simulador (NS) versão 2.0 (NS, 2002), executando em sistema operacional Linux (RedHat 7.1) e arquitetura Intel PC.

Esta ferramenta foi desenvolvida em 1989, a partir de um projeto da Cornel University, EUA. Atualmente, a DARPA está financiando o seu desenvolvimento através do projeto Saman (Saman, 2002) e pelo NSF através do projeto Conser (Conser, 2002). O simulador já recebeu apoio do Lawrence Berkeley National Laboratory, da Xerox PARC (Palo Alto Research Center), da Universidade da Califórnia em Berkeley, Sun Microsystems e também agrega diversos módulos como contribuição de pesquisadores independentes. É um software de código livre e fornecido gratuitamente (NS, 2002). Uma lista de discussão é mantida com o objetivo de incorporar melhorias e de esclarecer dúvidas de pesquisadores.

O NS é um simulador de eventos discretos focado no desenvolvimento de pesquisas em redes de computadores. Provê suporte aos principais protocolos como por exemplo TCP, UDP, redes sem fio e implementa filas do tipo *drop tail*, *fair queueing* (FQ), *class-based queueing* (CBQ), dentre outras e algoritmos de escalonamento de pacotes como o WRR e o WFQ (Huston e Ferguson, 1998).

O pacote básico do NS-2 possui recursos para a implementação do serviço PHB-AF da estratégia *Diffserv*. Para a implementação do serviço PHB-EF, torna-se necessário a criação de novas classes C++ .

Outros recursos interessantes para simulação de redes de computadores são o gerador de números aleatórios, os vários tipos de geradores de tráfego (exponencial ON-OFF, CBR dentre outros) e os recursos para coleta e registro de dados de cada evento da simulação. Um outro recurso interessante é o NAM (*Network Animator*) que é um animador gráfico, permitindo a visualização da topologia da rede e do fluxo de pacotes.

O NS é uma ferramenta desenvolvida na linguagem C++. Entretanto é disponibilizada para o usuário uma interface para a criação dos *scripts* de simulação através da linguagem Otcl (*Object Tool Command Language*) (OTcl, 2002). A linguagem OTcl é interpretada e por isso mais lenta, entretanto oferece simplicidade para alteração dos parâmetros de configuração, tarefa constante e necessária durante uma projeto de pesquisa em redes.

Os escalonadores de eventos e os componentes de rede são implementados através de objetos C++. Estes objetos são replicados em objetos OTcl e , desta forma, o usuário tem a oportunidade de acessar , através do OTcl, os métodos destes diversos objetos. O usuário escreve o *script* OTcl e executa a simulação. O NS é apenas um interpretador OTcl.

Para execução de uma simulação, torna-se necessário criar o *script* desejado e executar o *script*. Só então é que os resultados da simulação poderão ser obtidos. É exatamente este o ponto fraco do NS.

O NS não fornece estatísticas de forma automática. Ele possui alguns procedimentos matemáticos e alguns objetos para geração de estatísticas que podem ser manipulados através dos *scripts*. Entretanto, estas estatísticas são muito superficiais, não sendo suficientes para análises mais profundas. Torna-se necessário a utilização de uma ferramenta que analise o arquivo de *tracing*, gerado após execução do *script* e onde são armazenados os resultados de cada evento simulado, e faça os cálculos desejados. A ferramenta utilizada neste trabalho foi o Tracegraph (Tracegraph, 2002).

A utilização do NS foi de fundamental importância, uma vez que viabilizou a criação de um modelo de rede e a execução de simulações que permitiram validar as conclusões obtidas nestes trabalho.

O manual da ferramenta pode ser encontrado em (NS, 2002). Um tutorial simplificado sobre a instalação da ferramenta e a criação de *scripts* pode ser obtido em (Portnoi e Araújo, 2002).

5.3 TOPOLOGIA UTILIZADA

O modelo de simulação *Diffserv* (vide figura 11) proposto consiste em cinco nós interligados através de enlaces de 10 Mbit/s (nós de fronteira) e 1 Mbit/s (nós do núcleo). Os nós zero e 1 representam os geradores do tráfego de voz e FTP respectivamente. Os geradores de tráfego encaminham seus pacotes através de links de 10 Mbit/s ao nó 2 ou *edge0* que representa o roteador de fronteira de entrada. O mesmo recebe os pacotes e após a associação do tráfego a uma classificação, encaminha os pacotes ao roteador do núcleo ou *core* (nó 3) utilizando o PHB/AF através de um enlace de 1 Mbit/s. O *core*, por sua vez, encaminha os pacotes para o roteador de fronteira de saída ou *edge1* (nó 4) utilizando a política PHB/AF, e por fim os pacotes são enviados ao nó receptor (nó 5) pelo *edge1* através de um enlace de 10 Mbit/s. mulada

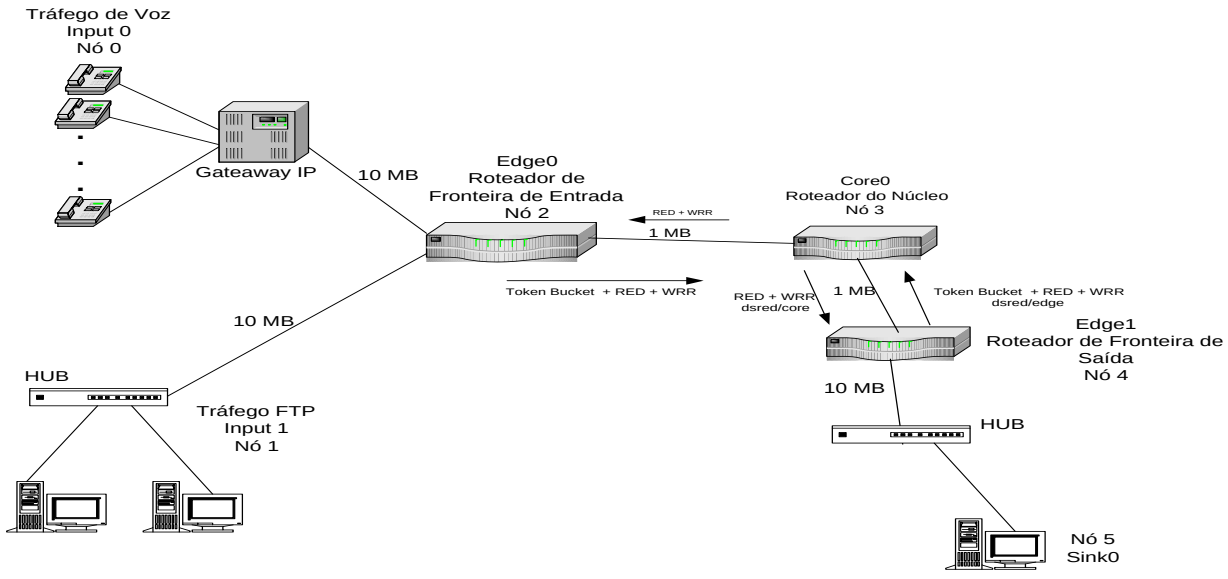

Figura 11. Topologia da rede simulada.

A escolha da topologia acima descrita justifica-se por ser uma topologia simples e clara que viabiliza a análise do tráfego de voz quando concorrendo com um outro tipo de tráfego, além de permitir a implementação dos elementos característicos de um modelo *Diffserv*.

O script em anexo neste documento mostra a criação da topologia da rede simulada.

5.4 O TRÁFEGO MONITORADO

As simulações realizadas envolvem 16 fontes *ON-OFF* para geração do tráfego de voz e 2 fontes para geração do tráfego FTP.

As fontes geradoras do tráfego de voz foram configuradas como fontes exponenciais *ON-OFF*. Quando as fontes *ON-OFF* estão no estado de atividade ON é gerado um tráfego de voz a um pico de 64 Kbps. Os períodos de atividade são determinados por uma variável aleatória com distribuição exponencial de média 400 ms. Os períodos de silêncio OFF também são distribuídos exponencialmente em uma variável aleatória de média 600 ms. O tamanho dos pacotes e a taxa de transmissão dessas fontes também são parâmetros configuráveis. A criação das fontes geradoras de tráfego é mostrada no *script* que está como anexo deste trabalho.

5.5 O MODELO DE SIMULAÇÃO UTILIZADO

O modelo de simulação escolhido utiliza a estratégia *Diffserv* por se tratar de um modelo de rede que possui mecanismos de avaliação e suavização de tráfego e esquemas de encaminhamento de tráfego que permitem que as necessidades de QoS das aplicações VoIP possam ser atendidas.

No modelo de simulação, os tráfegos FTP e voz são concorrentes. O tráfego FTP é do tipo *best effort*, ou seja, sem quaisquer garantias de atraso, perda ou qualquer outro requisito de QoS, e o tráfego de voz possui requisitos de QoS. A diferenciação de serviços tornou-se essencial para que

o tráfego de voz pudesse ser transmitido com prioridade e consequentemente os parâmetros de QoS pudessem ser alcançados.

Para simular a estratégia *Diffserv*, foi necessário implementar os mecanismos que compõe a política PHB-AF. Os mecanismos implementados são os listados abaixo:

- classificação do tráfego em uma fila de prioridade implementado através do *token bucket*;
- gerenciamento ativo de filas implementado através do RIO;
- escalonamento das filas nas interfaces de saída e seleção dos pacotes que serão enviados implementado através do WRR.

O mecanismo de *token bucket* foi utilizado por viabilizar a classificação do tráfego em quatro filas de prioridade (uma fila *in* e uma fila *out* para o tráfego do tipo FTP e uma fila *in* e uma fila *out* para o tráfego do tipo voz), permitindo dessa forma que o tráfego de voz pudesse ser priorizado em detrimento ao tráfego FTP através da configuração dos seus parâmetros. A marcação dos pacotes de voz de acordo com um perfil de serviço garantem que o tráfego poderá ser transmitido com uma alta prioridade, o que implicará em uma menor probabilidade de descarte, uma maior liberação de banda e um menor tempo em fila. O mecanismo de *token bucket* foi essencial para o controle do percentual de perdas através da restrição do tráfego FTP que tende a tomar toda a banda passante, prejudicando bastante o tráfego de voz.

O tráfego FTP gerado pelo NS é uma aplicação simples: apenas aciona o serviço TCP para tentar transmitir à maior velocidade possível. O controle da velocidade somente é feito pelo próprio TCP, através de mecanismos de controle de fluxo como o *sliding window* e mecanismos de controle de congestionamento. A tendência é que, se o tráfego de voz fosse homogêneo, o tráfego FTP dividisse a banda com ele. Como o tráfego de voz não é homogêneo, o TCP/FTP

busca tomar a banda disponível, só liberando quando houver congestionamento. Ou seja, FTP é um tráfego dito “elástico”, mas a voz usando UDP não é e, consequentemente, termina sendo afetada pelo congestionamento.

O RIO (RED com In e Out) foi utilizado por representar um eficiente mecanismo de gerenciamento ativo de filas que busca evitar o descarte de pacotes com alta prioridade. Com a utilização deste mecanismo, os pacotes de voz, já classificados em uma classe de alta prioridade pelo *token bucket*, tiveram a probabilidade de descarte bastante reduzida, uma vez que o RIO exerce um controle do congestionamento e busca descartar pacotes de baixa prioridade para que o congestionamento não ocorra e prejudique os pacotes com alta prioridade. A configuração dos parâmetros do RIO por fila de prioridade viabilizou o controle do *jitter*, do atraso e do percentual de perdas do tráfego de voz.

O WRR (*Weighted Round Robin*) foi o algoritmo de prioridade escolhido para o escalonamento nas interfaces de saída e decide quais pacotes serão enviados baseado nas filas de prioridade, o que permitiu exercer um controle sobre o percentual de perdas do tráfego de voz. Além disso, este algoritmo está implementado nos roteadores disponibilizados comercialmente. Desta forma, a configuração adequada de roteadores existentes atualmente pode prover o comportamento proposto pelo serviço AF.

Os parâmetros referentes à estratégia *Diffserv* configurados no modelo estão vinculados ao mecanismo de suavização de tráfego *token bucket*, ao algoritmo de controle de congestionamento RIO e ao algoritmo de prioridade WRR, conforme descrição textual presente no capítulo 4.

A Figura 12 a seguir mostra os parâmetros a serem configurados nos roteadores.

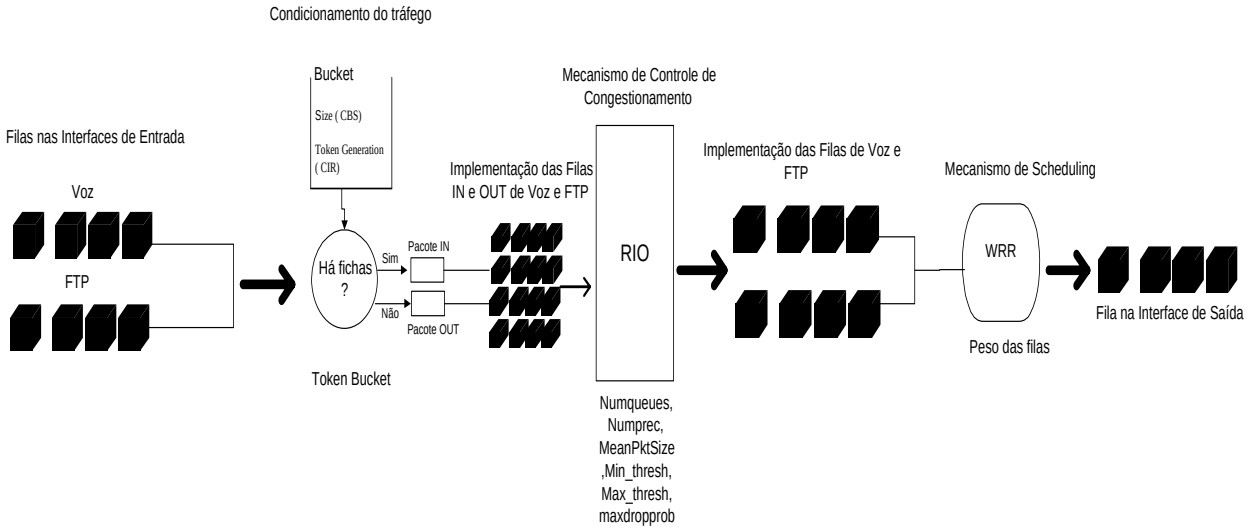

Figura 12. Parâmetros da estratégia Diffserv .

O *script* contendo a criação do modelo *Diffserv*, assim como a configuração dos valores dos parêmetros encontra-se como anexo deste documento.

A análise da figura 12 coincide um pouco com a análise do modelo *Diffser*v por mostrar exatamente os elementos característicos da estratégia *Diffserv* descritos no capítulo 4.

Na interface de entrada, são formadas duas filas que acomodam os diferentes tipos de tráfego: voz e FTP. Os pacotes passam pelo mecanismo de *token bucket* para que sejam classificados em

uma das filas de prioridade (i*n* ou o*ut*), sendo estas filas gerenciadas pelo algoritmo RIO de forma a evitar o congestionamento e reduzir o descarte de pacotes. Baseado no peso da filas, que define a prioridade de atendimento, o WRR é executado com o objetivo de formar a fila na interface de saída para o envio dos pacotes (desenfileiramento).

.

6 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS DA SIMULAÇÃO

Este capítulo tem como objetivo descrever as simulações realizadas, assim como proceder a uma análise dos seus resultados.

Os resultados serão apresentados através de cenários de simulação que mostram a definição dos valores dos parâmetros de tráfego e os parâmetros de QoS obtidos. A análise dos resultados coletados busca definir o relacionamento existente entre os diversos parâmetros de operação do *Diffserv* e os parâmetros de QoS que são críticos para uma aplicação VoIP, como por exemplo atraso, *jitter* e taxa de perdas. O objetivo desta análise é analisar alguns parâmetros de configuração de roteadores e seus relacionamentos necessários para garantir níveis aceitáveis de determinados parâmetros de QoS que viabilizem um bom desempenho fim a fim das aplicações VoIP.

A análise e apresentação dos resultados será feita inicialmente por parâmetro de QoS e, como conclusão do capítulo, será apresentada uma análise envolvendo todos os parâmetros considerados na simulação.

Os cenários de simulação apresentam alguns valores de parâmetros de configuração que são fixos em todos os cenários.

Este é o caso dos parâmetros das fontes ON- OFF. Os períodos de atividade destas fontes são determinados por uma variável aleatória com distribuição exponencial de média 400 ms. Os períodos de silêncio OFF também são distribuídos exponencialmente por uma variável de 600 ms. (Rezende e Ziviani, 1999). Por isso, os parâmetros "*burst*" e "*idle*" , em todos os cenários, são configurados respectivamente com os valores 0,4 s e 0,6 s. O tamanho do pacote de voz

(parâmetro “UDPPacketsize”) foi configurado com o valor de 80 bytes (Douskalis, 2000) e o parâmetro “*rate*” é configurado com o valor de 64 kbps seguindo a recomendação G.711 do padrão ITU-T, por ser a menos eficiente em termos de consumo de banda. Como as fontes ON-OFF são usadas como geradoras de tráfego de voz, o parâmetro “*size*” foi configurado com o tamanho fixo dos pacotes enviados pelas mesmas, isto é, 80 bytes.

O parâmetro “*Numqueue*” é configurado com o valor 2, por existirem duas fila físicas: uma fila para o tráfego UDP e uma fila para o tráfego FTP.

O parâmetro “*Setnumprec*” sempre será configurado com o valor 2, uma vez que existem dois níveis de precedência de descarte, isto é in e out.

O parâmetro “*MeanPktSize*” sempre é configurado com o tamanho do pacote UDP (80 bytes) por serem estes os pacotes que serão enviados em um maior número, em função da maior quantidade de fontes de voz.

A escolha do número de fontes de voz e FTP representa uma opção de projeto.

O tamanho do pacote do tráfego FTP, os valores de configuração dos parâmetros do algoritmo RED, do *token bucket* e do algoritmo WRR são configurados com valores variáveis.

Os valores dos parâmetros de configuração foram escolhidos com o objetivo de avaliar o impacto dos mesmos sobre os parâmetros de QoS coletados, além de mostrar uma seqüência lógica de evolução das simulações. Referências como (Rezende e Ziviani, 1999) e (Floyd e Jacobson, 1993) foram utilizadas, além de configurações praticadas comercialmente como as utilizadas em roteadores Cisco.

Recomendações sobre a configuração dos parâmetros do esquema *Diffserv* podem ser vistas no capítulo 4.

Inicialmente será avaliado globalmente o comportamento do modelo proposto para analisar o grau de dispersão dos resultados por canal.

6.1 COMPORTAMENTO SIMILAR DOS CANAIS DE VOZ

O objetivo inicial da simulação é mostrar que os diversos canais de voz apresentam comportamento similar quando submetidos aos mesmos parâmetros de tráfego.

O cenário considerado, no que diz respeito à implantação do serviço *Diffserv* nos roteadores usados no *backbone* da rede simulada pode ser observado através da figura 12.

Os parâmetros da implantação *Diffserv*, explicitados na figura 12, são configurados conforme os valores listados no quadro 2.

Quadro 2. Valores de configuração de parâmetros de simulação

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**12 de voz e duas FTP**

Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	3000 bytes
Max_Thresh IN voz	6000 bytes
Min_Thresh OUT voz	1500 bytes
Max_Thresh OUT voz	3000 bytes
Min_Thresh IN FTP	2 bytes
Max_Thresh IN FTP	4 bytes
Min_Thresh OUT FTP	1 bytes
Max_Thresh OUT FTP	2 bytes
maxdropprob IN voz	0.01
maxdropprob OUT voz	0.05
maxdropprob IN FTP	0.8
maxdropprob OUT FTP	0.9
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	10 Kbps
CBS FTP	20 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

Os valores escolhidos tiveram o objetivo inicial de priorizar, da forma mais extrema possível, o tráfego de voz.

O quadro 3 abaixo ilustra os resultados encontrados por canal de voz.

Quadro 3. Resultados por canal de voz

	Atraso (mseg)	**Valor máximo do *Jitter* (mseg)**	**Percentual de perda**

Voip0	94	35	7.5
Voip1	96	35	7.6
Voip2	93	20	7.0
Voip3	93	20	8.6
Voip4	90	20	7.0
Voip5	91	20	7.6
Voip6	91	20	8.7
Voip7	97	40	8.2
Voip8	93	20	8.0
Voip9	96	26	8.0
Voip10	91	20	6.2
Voip11	94	35	7.9

O quadro 4 apresenta o cálculo do intervalo de confiança de 95 % por canal de voz.

Quadro 4. Intervalo de confiança por canal de voz

VOIP0			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	95,78	7,50	35
	97,32	8.42	35
	92,65	8,10	30
	91,65	6,94	18
	94,00	5,87	35
Média	94,28	7,1025	30,6
Desvio Padrão	2,299119397	0,948415345	7,368853371
IC	91,42572387	5,9250756	21,45182688

	97,13427613	8,2799244	39,74817312
VOIP1			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	96,00	7.60	35
	93,59	8,19	35
	96,62	9,23	17
	96,42	7,85	30
	94,17	6,74	17
Média	95,36	8,0025	26,8
Desvio Padrão	1,384720188	1,026170064	9,176055798
IC	93,64091843	6,728545842	15,40824843
	97,07908157	9,276454158	38,19175157
VOIP2			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	93,00	7,00	20
	92,04	7,11	28
	93,67	7,60	28
	93,81	6,75	10
	90,74	6,99	35
Média	92,652	7,09	24,2
Desvio Padrão	1,277564088	0,313926743	9,549869109
IC	91,06594897	6,700270954	12,34417231
	94,23805103	7,479729046	36,05582769
VOIP3			

	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	93,00	8,60	20
	95,88	7,10	20
	96,05	7,17	18
	91,77	8,63	40
	96,34	6,50	20
Média	94,61	7,60	23,60
Desvio Padrão	2,081890967	0,962522727	9,208691547
IC	92,02340535	6,405061779	12,16773229
	97,19259465	8,794938221	35,03226771
VOIP4			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	90,00	7,00	20
	90,62	6,50	20
	95,53	8,32	32
	95,24	6,43	35
	87,36	6,51	38
Média	91,75	6,95	29,00
Desvio Padrão	3,538361203	0,797790699	8,485281374
IC	87,35724862	5,961570816	18,46582066
	96,14275138	7,942429184	39,53417934
VOIP5			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	91,00	7,60	20
	92,44	7,50	20
	93,02	7,12	15
	96,17	6,80	17

	88,33	5,67	30
Média	92,19	6,94	20,40
Desvio Padrão	2,868252081	0,776865497	5,770615219
IC	88,6311656	5,973548722	13,23598352
	95,7528344	7,902451278	27,56401648
VOIP6			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	91,00	8,70	20
	94,93	7,21	40
	92,92	6,63	18
	92,00	7,06	35
	99,28	7,16	30
Média	94,03	7,35	28,60
Desvio Padrão	3,275145798	0,78750873	9,476286192
IC	89,96002131	6,374335521	16,83552292
	98,09197869	8,329664479	40,36447708
VOIP7			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	97,00	8,20	40
	93,53	7,80	20
	92,73	6,99	35
	97,33	8,03	15
	92,61	6,62	30
Média	94,64	7,53	28,00
Desvio Padrão	2,334887578	0,687655437	10,36822068
IC	91,74131893	6,674299883	15,12821753

	97,53868107	8,381700117	40,87178247
VOIP8			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	93,00	8,00	20
	90,95	6,77	25
	97,80	9,07	35
	88,55	6,21	36
	91,50	7,32	20
Média	92,36	7,47	27,20
Desvio Padrão	3,436822661	1,111409016	7,854934755
IC	88,09330516	6,094224672	17,44837389
	96,62669484	8,853775328	36,95162611
VOIP10			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**
	91,00	6,20	20
	94,09	7,31	35
	93,27	7,30	17
	90,27	7,98	20
	94,20	6,15	15
Média	92,57	6,99	21,40
Desvio Padrão	1,817424001	0,791877516	7,893034904
IC	90,30973182	6,004911827	11,60107389
	94,82226818	7,971088173	31,19892611
VOIP11			
	Atraso(mseg)	**Percentual de Perda**	**Jitter Máximo(mseg)**

	94,00	7,90	35
	93,47	7,96	26
	96,13	6,51	40
	92,04	5,95	23
	98,02	7,14	20
Média	94,73	7,09	28,80
Desvio Padrão	2,3526942	0,873424296	8,408329204
IC	91,81121263	6,007674358	18,36135408
	97,65278737	8,176325642	39,23864592

Conclui-se que o comportamento dos canais é similar. Desta forma, será desenvolvida uma avaliação do comportamento de um canal e feita uma generalização dos resultados para os demais.

6.2 PERCENTUAL DE PERDA

O percentual de perda para aplicações VoIP deve ser menor ou igual a 10% [ITU-T recomendação G.114].

No contexto da simulação, entende-se por percentual de perda como sendo o percentual de pacotes descartados durante toda a simulação, sendo portanto calculado através da divisão do número de pacotes descartados pelo número de pacotes enviados durante a simulação.

Do ponto de vista deste trabalho de investigação e análise, entende-se inicialmente que o percentual de perdas na operação de roteadores pode vir a ser influenciado pelo seguinte conjunto de parâmetros de configuração:

A figura 12 ilustra a ocorrência destes parâmetros no contexto da implementação *Diffserv* utilizada na simulação.

- Parâmetro de operação do algoritmo de escalonamento WRR identificado na figura 12 como o “peso” da fila;
- Número de fontes geradoras de tráfego de entrada;
- Parâmetros do algoritmo de controle de congestionamento RED identificados na figura 12 como “*min_thresh*” e “*max_thresh*”;
- Parâmetros do *token bucket* identificados na figura 12 como CIR e CBS.

Em seguida, apresenta-se e discute-se os resultados das simulações realizadas, onde os diferentes cenários são definidos, considerando-se alternativas de configuração para os parâmetros de configuração indicados.

6.2.1 PARÂMETRO DO ALGORITMO WRR

Quanto maior o peso de uma fila, maior a prioridade de atendimento. Na ferramenta NS-2, o parâmetro peso deve variar de 1 a 10. Quando ocorre congestionamento, este peso é usado para definir uma maior ou menor porção da largura de banda para o tráfego enfileirado de acordo com a prioridade atribuída a cada tráfego. Logo, espera-se que, quanto maior for o peso da fila menor será o percentual de perdas do tráfego associado.

O cenário adiante busca a verificação desta expectativa.

Cenário 1 :

Objetivo: Mostrar a influência do parâmetro peso da fila sobre o percentual de perda. Os parâmetros abaixo foram mantidos inalterados. Os únicos parâmetros alterados foram os pesos das filas.

O quadro 5 mostra os valores dos parâmetros configurados neste cenário.

Quadro 5. Valores de configuração de parâmetros de simulação do cenário 1

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**16 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	100 bytes
Max_Thresh IN voz	200 bytes
Min_Thresh OUT voz	50 bytes
Max_Thresh OUT voz	100 bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes

Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05
Maxdropprob IN FTP	0.10
Maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	1024 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	N
Prioridade da fila FTP para o WRR	N

O quadro 6 apresenta os resultados obtidos na simulação, considerando a alteração nos pesos das filas FTP e VoIP. Buscou-se, através da escolha dos valores para o parâmetro peso, estabelecer possibilidades de combinação que pudessem trazer resultados que mostrassem, de forma clara, como este parâmetro influencia no percentual de perda. Por restrições da ferramenta NS-2, este parâmetro só pode receber valores de 1 a 10, sendo que a soma dos pesos dos tráfegos de voz e FTP deve atingir o valor 10, representando um percentual de 100 %.

Quadro 6. Resultados do cenário 1

IC- Cenário 01 – Percentual de Perda					
Peso Voz, FTP (WRR)	**9,1**	**8,2**	**6,4**	**5,5**	**4,6**
	25,98	54,64	76,84	77,17	80,19
	24,57	55,65	73,15	77,26	81,53
	26,17	51,08	73,77	76,91	83,47
	25,28	54,42	73,72	75,33	82,44
	25,15	54,94	73,76	76,09	81,73

Média	25,43	54,15	74,25	76,55	81,87
Desvio Padrão	0,65011537	1,77566	1,4722	0,82433	1,208561
IC	24,6229046	51,9416	72,4203	75,5286	80,37161
	26,2370954	56,3504	76,0757	77,5754	83,37239

A figura 13 a seguir mostra a relação entre o peso das filas e o percentual de perda. Foi utilizada a média do percentual de perda das amostras acima listadas.

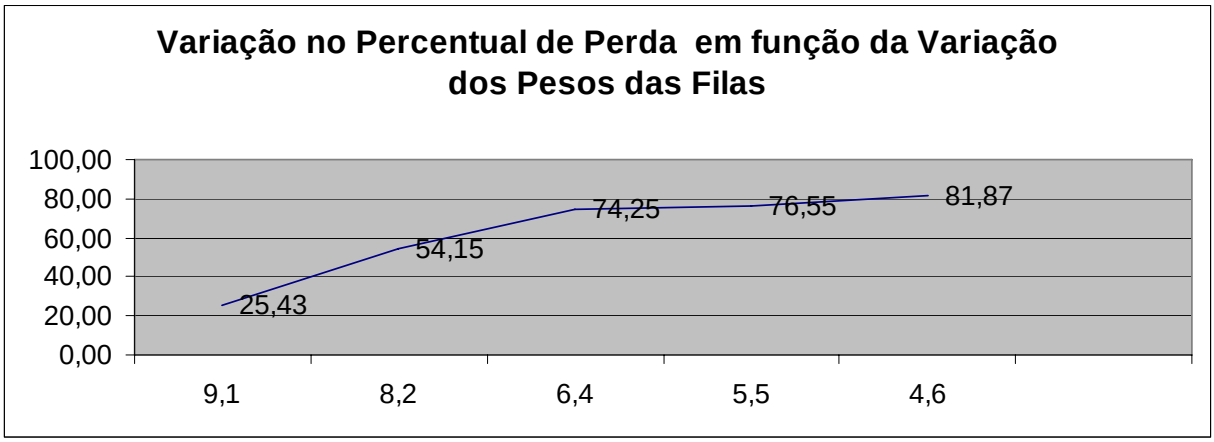

Figura 13. Relação entre percentual de perda e peso das filas

A primeira constação óbvia dos resultados da simulação é que as perdas associadas aos canais de voz são altas (entre 25,43 e 81,87 %), mesmo com uma priorização máxima para a fila VoIP. Isto demonstra que a simples priorização no algoritmo de escalonamento não garante uma operação tal que resulte em baixas perdas para os canais de voz.

A explicação advém dos seguintes fatos:

- A priorização 9:1 (VoIP:FTP) efetivamente garante que as filas VoIP serão descarregadas 9 entre 10 vezes;

- Como os tamanhos dos pacotes são diferentes (VoIP = 80 bytes e FTP = 1500 bytes), a banda líquida associada às filas VoIP acaba sendo inferior a 90 % da capacidade do enlace.

A investigação prossegue então com o seguinte intuito:

Em que condições de configuração, a priorização permite uma efetiva priorização dos canais de voz no WRR que levam a uma perda dentro dos limites recomendados pelo ITU-T ?

6.2.2 NÚMERO DE FONTES

Quanto maior o número de fontes, maior será o número de pacotes enviados por fluxo. Isto contribui para um aumento da probabilidade de descarte, ou seja, do percentual de perdas.

Os resultados abaixo buscam verificar a afirmação descrita acima.

Cenário 2 :

Objetivo: Mostrar a influência do número de fontes sobre o percentual de perda. Os parâmetros abaixo foram mantidos inalterados. O único parâmetro alterado foi o número de fontes.

O quadro 7 mostra os valores dos parâmetros configurados neste cenário.

Quadro 7. Valores de configuração de parâmetros de simulação do cenário 2

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes

Número de Fontes	**N de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	100 bytes
Max_Thresh IN voz	200 bytes
Min_Thresh OUT voz	50 bytes
Max_Thresh OUT voz	100 bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
maxdropprob IN voz	0.01
maxdropprob OUT voz	0.05
maxdropprob IN FTP	0.10
maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

Não houve alterações significativas nas configurações dos parâmetros nos *scripts*, apenas o valor do CIR do tráfego de voz. O número de fontes de voz foi reduzido através da não execução de 4 fontes geradoras do tráfego de voz. Esta configuração foi realizada através da atribuição do caracter de comentário(#) nos geradores de voz: *voipgenerator* 12,13,14 e 15.

O quadro 8 apresenta os resultados das simulações do cenário 2.

Quadro 8. Resultados do cenário 2

Cenário 02 - Percentual de Perda				
Número de Fontes	**16**	**14**	**12**	**6**
	25,98	17,45	11,03	0
	24,57	18,41	12,08	0
	26,17	15,64	9,46	0
	25,28	19,92	9,35	0
	25,15	19,66	9,16	0
Média	25,43	18,22	10,22	0,00
Desvio Padrão	0,65011537	1,74992	1,28196	0
IC	24,6229046	16,0435	8,62449	0
	26,2370954	20,3885	11,8075	0

A figura 14 ilustra o resultado da simulação obtido para análise da taxa de perdas em relação ao número de fontes com configuração fixa de parâmetros WRR(9:1) e tráfego FTP (2 fontes e pacote de 1500 bytes). Foi utilizada a média do percentual de perda das amostras acima listadas.

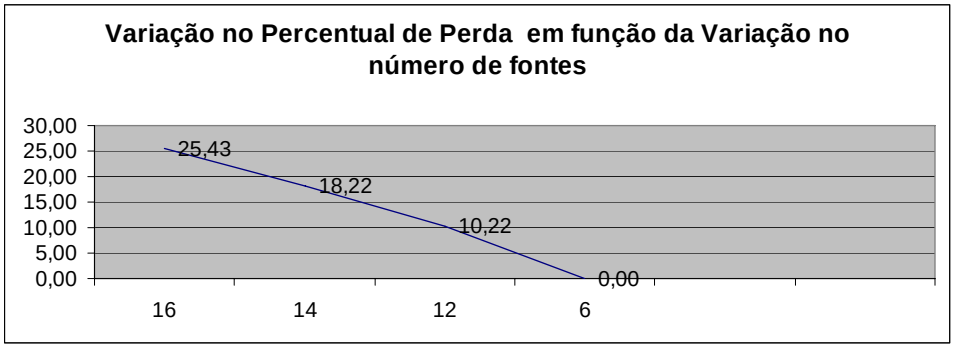

Figura 14. lação entre percentual de perda e número de fontes

A primeira conclusão obtida é que atende-se os requisitos ITU-T para perdas apenas com configurações de até 12 canais de voz. Além disso, verificou-se também que a taxa de perda zero só é atingida com a configuração de, no máximo, 6 canais de voz.

As perguntas que seguem são as seguintes:

1. A partir de que valor de tráfego FTP (dados), observa-se um efetivo comprometimento do algoritmo de escalonamento WRR ?

2. A dependência entre os tráfegos VoIP e FTP depende mais do tráfego ou do tamanho dos pacotes utilizados pela aplicação de dados ?

O cenário apresentado a seguir busca apresentar resultados que esclareçam estes questionamentos.

6.2.3 TAMANHO DO PACOTE DE DADOS (FTP)

Em função do mecanismo de funcionamento do algoritmo WRR já explicado anteriormente, quanto maior o tamanho do pacote FTP, maior será a largura de banda alocada para este tipo de tráfego, o que fará com que o tráfego de voz seja prejudicado na disputa pelo acesso a largura de banda compartilhada. Espera-se que quanto maior o tamanho do pacote FTP, maior será o percentual de perdas.

Os resultados mostram que pacotes FTP muito grandes comprometem o mecanismo de priorização implementado pelo algoritmo WRR, ou seja, mesmo configurando uma relação de

prioridade 9:1 para os tráfegos de voz e FTP, não se consegue uma priorização efetiva do tráfego de voz.

Torna-se necessário, portanto, definir um tamanho máximo de pacote FTP que, através da efetiva priorização efetuada pelo WRR, não comprometa o tráfego de voz.

Cenário 3 :

Objetivo: Definir um tamanho máximo de pacote FTP através do estabelecimento de prioridades utilizando a configuração do parâmetro do WRR.

O quadro 9 mostra os valores dos parâmetros configurados neste cenário.

Quadro 9. Valores de configuração de parâmetros de simulação do cenário 3

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	**N** bytes
Número de Fontes	16 de voz e duas FTP
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	100 bytes
Max_Thresh IN voz	200 bytes

Min_Thresh OUT voz	50 bytes
Max_Thresh OUT voz	100 bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05
Maxdropprob IN FTP	0.10
Maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	1024 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	N bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

O quadro 10 apresenta os resultados da simulação, considerando a alteração no tamanho do pacote de dados.

Quadro 10. Resultados do cenário 3

IC- Cenário 03 – Percentual de Perda				
WRR : 9:1				
Tam.Pacote FTP	100	750	900	1000
	0	5,40	8,09	11,59
	0	5,21	8,69	11,29
	0	3,38	8,87	11,52
	0	3,71	8,30	12,43
	0	3,35	9,95	11,17
Média	0,00	4,21	8,78	11,60

Desvio Padrão	0	1,0117559	0,723118248	0,49406477
IC	0	2,9539405	7,882274047	10,9866359
	0	5,4660595	9,677725953	12,2133641

A figura 15 a seguir mostra a relação entre o tamanho do pacote e o percentual de perdas. Foi utilizada a média do percentual de perdas das amostras consideradas.

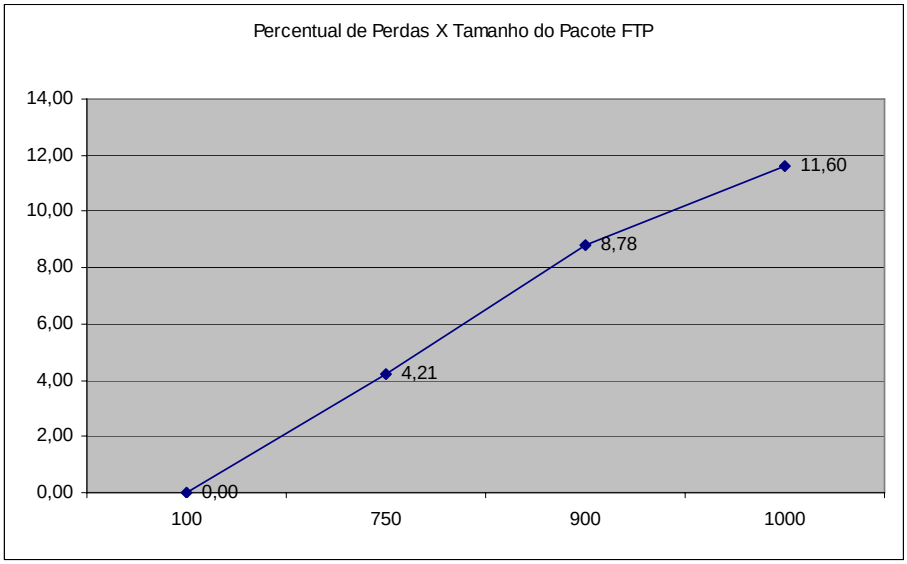

Figura 15. Relação entre percentual de perdas e tamanho do pacote de dados

O quadro 11 apresenta os resultados para o mesmo cenário, alterando apenas o peso das filas de voz e FTP para respectivamente 6 e 4.

Quadro 11. Resultados para os pesos 6:4

Parâmetros do WRR: 6: 4				
Tam pacote FTP	100	110	120	130
	6,53	8,08	9,10	11,82
	4,4	9,50	9,26	12,26

	5,46	10,00	10,00	13,67
	7,47	9,00	12,23	16,82
	6,18	7,92	12,46	11,34
Média	6,01	8,90	10,61	13,18
Desvio Padrão	1,15385	0,8962143	1,621850795	2,21197197
IC	4,57554	7,7873814	8,596529099	10,4359143
	7,44046	10,012619	12,6234709	15,9280857

A figura 16 a seguir mostra a relação entre o tamanho do pacote e o percentual de perdas quando a relação de prioridade utilizada foi 6:4. Foi utilizada a média do percentual de perdas das amostras consideradas.

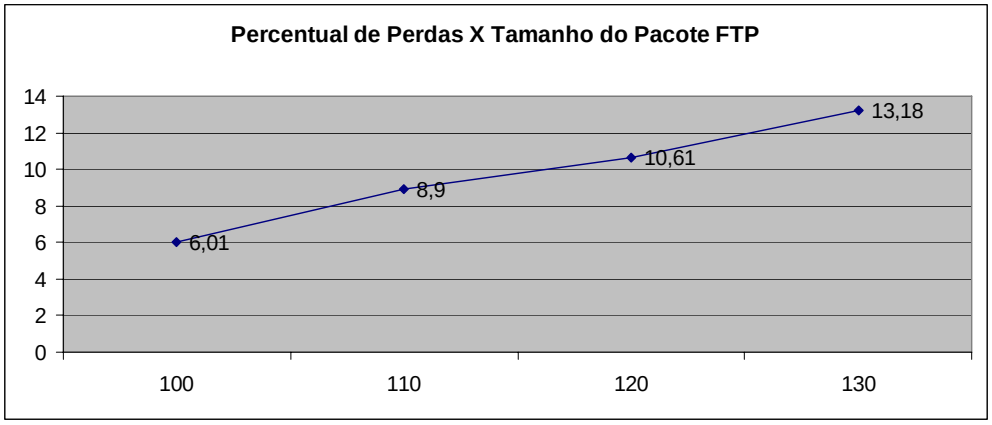

Figura 16. Relação entre perda e tamanho do pacote para 6:4

Os resultados mostram que o tamanho do pacote FTP afeta diretamente o percentual de perdas, uma vez que interfere na eficiência do mecanismo de escalonamento WRR. Pode-se verificar que utilizando uma priorização 9:1, o tamanho máximo de pacote permitido foi de 900 bytes. Quando a priorização utilizada foi de 6:4, o tamanho máximo permitido foi de 110 bytes. A partir destes valores de tamanho de pacote FTP, observa-se o comprometimento do algoritmo WRR.

A redução do tamanho do pacote FTP dá ao tráfego de voz condições mais justas de concorrência pelo acesso à banda, já que os pacotes de voz possuem 80 bytes. Torna-se clara que a dependência entre os tráfegos depende mais do tamanho dos pacotes utilizados. Os requisitos de perda do tráfego de voz são atendidos, uma vez que este tráfego recebe uma maior prioridade de atendimento implementada pela configuração do WRR com 6:4. Por outro lado, o tráfego FTP não é totalmente restrito como na configuração 9:1, uma vez que utiliza-se uma distribuição de prioridade de atendimento mais justa.

6.2.4 PARÂMETROS DO ALGORITMO RED

Os parâmetros do RED definirão o descarte. Quanto maior o parâmetro "MaxP", mais agressivo será o algoritmo de descarte e quanto maior o "*max_thresh*" e "*min_thresh*", menor a probabilidade de descarte de pacotes, uma vez que os pacotes terão *buffer* suficiente para serem enfileirados e consequentemente encaminhados, no lugar de descartados. Valores altos para estes parâmetros garantem que o descarte de pacotes seja forçosamente atenuado pela existência de uma fila muito grande. Buscou-se, através da escolha dos valores dos parâmetros, mostrar o impacto de baixos e altos valores destes parâmetros sobre o percentual de perda. Os valores foram escolhidos por serem os que geraram resultados significativos e viabilizaram mostrar, de forma clara, a relação entre os parâmetros da estratégia *Diffserv* e o parâmetro de QoS descrita acima. Logo, espera-se que quanto maior os valores do "*max_thresh*" e do "*min_thresh*", menor será o percentual de perdas.

O cenário adiante busca a verificação desta expectativa.

Cenário 4 :

Objetivo: Mostrar a influência dos parâmetros "*max_thresh*" e "*min_thresh*" sobre o percentual de perda. Os parâmetros abaixo foram mantidos inalterados. Apenas foram alterados os parâmetros "*max_thresh*" e "*min_thresh*" do tráfego de voz.

O quadro 12 mostra os valores dos parâmetros configurados neste cenário.

Quadro 12. Valores de configuração de parâmetros de simulação do cenário 4

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**12 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	N bytes
Max_Thresh IN voz	N bytes
Min_Thresh OUT voz	N bytes
Max_Thresh OUT voz	N bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05

Maxdropprob IN FTP	0.10
Maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

O quadro 13 apresenta os resultados das simulações do cenário 4.

Quadro 13. Resultados do cenário 4

IC- Cenário 04 – Percentual de Perda				
Min_thresh, Max_thresh	**15,30/10,20**	**100,200/50,100**	**500,1000/250,500**	**3000,6000/1500,3000**
	16,00	11,03	7,60	9,47
	12,08	12,08	9,71	8,16
	16,03	9,46	10,74	7,28
	14,31	9,35	7,45	8,86
	12,29	9,16	10,75	8,79
Média	14,14	10,22	9,25	8,51
Desvio Padrão	1,91876783	1,281963338	1,631272509	0,830343302
IC	11,759917	8,62448746	7,22483237	7,481157901
	16,524083	11,80751254	11,27516763	9,542842099

A figura 17 mostra a relação entre o "*min_thresh*" e o "*max_thresh*" e o percentual de perda. Foi utilizada a média do percentual de perda das amostras acima listadas.

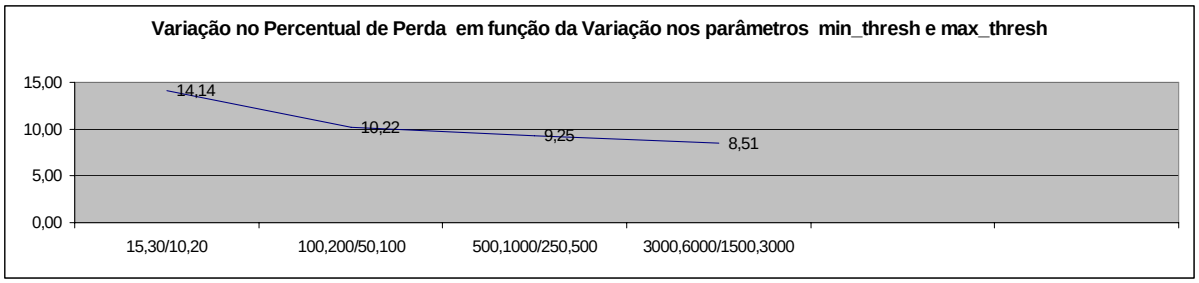

Figura 17. Relação entre percentual de perda e min_thresh e max_thresh

A figura 20 mostra que aumentando o valor do "*min_thresh*" e do "*max_thresh*" , tem-se uma redução do percentual de perda, ratificando a afirmação feita no tópico "Parâmetros do Algoritmo Red".

6.2.5 PARÂMETROS DO *TOKEN BUCKET*

Quanto mais restritos os parâmetros do *token bucket*, isto é menores, maior o número de pacotes classificados como "out" e sabe-se que os pacotes out têm maior prioridade de descarte que os pacotes "in", o que deverá contribuir para um aumento do percentual de perdas.

Além disso, é importante restringir os parâmetros de *token bucket* do tráfego FTP para que o percentual de perdas do tráfego de voz possa ser reduzido, o que se justifica através de uma liberação de banda disponível para o tráfego de voz, em função de um maior descarte do tráfego FTP. O tráfego FTP concorre com o de voz e roda sobre o TCP que tem mecanismo de controle de fluxo que, de certa forma, beneficia o tráfego FTP no que diz respeito ao percentual de perdas. Os resultados obtidos confirmam esta afirmação.

Cenário 5 :

Objetivo: Mostrar a influência dos parâmetros CIR e CBS sobre o percentual de perda. Os parâmetros abaixo foram mantidos inalterados. Os únicos parâmetros alterados foram o CIR e o CBS do tráfego de voz.

O quadro 14 mostra os valores dos parâmetros configurados neste cenário.

Quadro 14. Valores de configuração de parâmetros de simulação do cenário 5

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**12 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	3000 bytes
Max_Thresh IN voz	6000 bytes
Min_Thresh OUT voz	1500 bytes
Max_Thresh OUT voz	3000 bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes

Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05
Maxdropprob IN FTP	0.10
Maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	N Kbps
CBS voz	N bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

O quadro 15 apresenta os resultados das simulações do cenário 5, considerando a alteração no CIR e no CBS do tráfego de voz. Buscou-se através da escolha dos valores dos parâmetros, estabelecer combinações que pudessem mostrar resultados significativos.

Quadro 15. Resultados do cenário 5

IC- Cenário 05 - Percentual de Perda			
CIR e CBS de Voz	**768,80**	**100,20**	**1,10**
	9,47	11,52	12,15
	8,16	12,03	10,00
	7,28	11,33	17,12
	8,86	10,53	16,30
	8,79	10,83	15,42
Média	8,51	11,25	14,20
Desvio Padrão	0,8303433	0,587894548	3,011116072
IC	7,4811579	10,51814953	10,45980496
	9,5428421	11,97785047	17,93619504

A figura 18 mostra a relação entre os parâmetros CIR e CBS do *token bucket* e o percentual de perda. Foi utilizada a média do percentual de perda das amostras acima listadas.

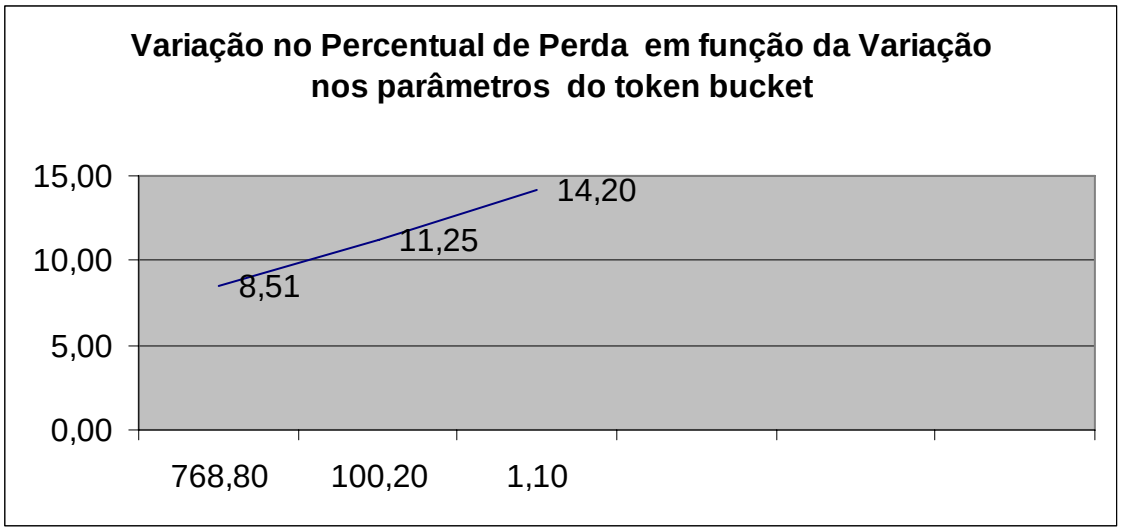

Figura 18. Relação entre percentual de perdas e parâmetros do token

A primeira constação é que quanto menor os valores do CIR e do CBS, mais restrito estará o tráfego de voz, ou seja, um maior número de pacotes deverá ser marcado com "out", o que implica em um aumento do percentual de perdas.

6.3 ATRASO

O atraso para uma aplicação VoIP deve ser menor ou igual a 200 ms [ITU-T recomendação G.114].

O atraso considerado foi o atraso médio fim a fim.

Do ponto de vista deste trabalho, entende-se que o atraso na operação de roteadores pode vir a ser influenciado pelos parâmetros de configuração descritos a seguir:

A figura 12 ilustra estes parâmetros no contexto utilizado na simulação.

- Parâmetros do algoritmo RED identificados na figura 12 como "*min_thresh*" e "*max_thresh*";

- Número de fontes geradoras de tráfego de entrada.

Em seguida, apresenta-se e discute-se os resultados das simulações, onde os diferentes cenários são definidos, considerando-se alternativas de configuração para os parâmetros acima indicados.

6.3.1 PARÂMETROS DO ALGORITMO RED

Sabe-se que parte do atraso é decorrente da velocidade do link, dos tempos de propagação de cada link, da velocidade dos geradores de tráfego e do número de fontes geradoras de tráfego. O atraso adicional é decorrente do tempo em fila que é influenciado pela configuração dos parâmetros min_thresh e max_thresh e pela banda disponível para o tráfego que pode ser controlada pelos parâmetros do *token bucket*. Quanto maior o min_thresh e max_thresh , maior será o número de pacotes enfileirados ao invés de descartados e consequentemente maior será o atraso.

O cenário a seguir busca a verificação desta expectativa.

Cenário 1 :

Objetivo: Mostrar a influência dos parâmetros "*min_thresh*" e "*max_thresh*" sobre o atraso. Apenas estes parâmetros do tráfego de voz foram alterados. Os demais foram mantidos fixos.

O quadro 16 mostra os valores dos parâmetros configurados neste cenário.

Quadro 16. Valores de configuração de parâmetros de simulação do cenário 1

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes

Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**12 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	N bytes
Max_Thresh IN voz	N bytes
Min_Thresh OUT voz	N bytes
Max_Thresh OUT voz	N bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05
Maxdropprob IN FTP	0.10
Maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

O quadro 17 apresenta os resultados obtidos na simulação, considerando a alteração nos "*min_thresh*" e "*max_thresh*".

Quadro 17. Resultados do cenário 1

IC- Cenário 01 - Atraso				
Min_thresh, Max_thresh	**15,30/10,20**	**100,200/50,100**	**500,1000/250,500**	**3000,6000/1500,3000**
	92,43	101,74	109,01	107,90
	86,80	110,10	117,35	120,00
	87,15	103,48	102,72	105,14
	92,68	88,00	103,40	103,06
	94,81	85,10	101,26	102,74
Média	90,77	97,68	106,75	107,77
Desvio Padrão	3,591327053	10,68104302	6,615721427	7,141282798
IC	86,31549337	84,42385956	98,53481379	98,90234777
	95,23250663	110,9441404	114,9611862	116,6336522

A figura 19 ilustra o resultado da simulação obtido para análise do atraso em relação aos parâmetros min_thresh e max_thresh. Foi utilizada a média do atraso das amostras acima listadas.

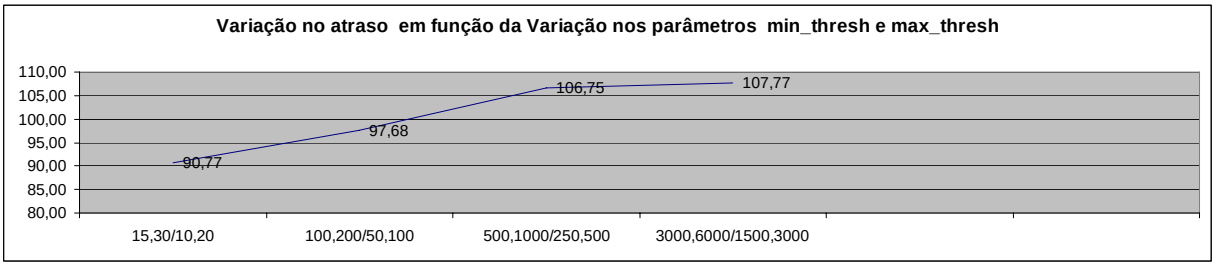

Figura 19. Relação entre o atraso e os parâmetros "min_thresh" e " max-thresh"

6.3.2 NÚMERO DE FONTES

Quanto maior o número de fontes, maior será o atraso, uma vez que maior será a quantidade de tráfego concorrente.

Os resultados abaixo visam verificar a afirmação descrita acima.

Cenário 2 :

Objetivo: Mostrar a influência do número de fontes sobre o atraso. O número de fontes foi o único parâmetro alterado, os demais parâmetros foram mantidos fixos.

O quadro 18 mostra os valores dos parâmetros configurados.

Quadro 18. Valores de configuração de parâmetros de simulação do cenário 2

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**N de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2

MeanPktsize	80 bytes
Min_Thresh IN voz	3000 bytes
Max_Thresh IN voz	6000 bytes
Min_Thresh OUT voz	1500 bytes
Max_Thresh OUT voz	3000 bytes
Min_Thresh IN FTP	2 bytes
Max_Thresh IN FTP	4 bytes
Min_Thresh OUT FTP	1 bytes
Max_Thresh OUT FTP	2 bytes
Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05
Maxdropprob IN FTP	0.80
Maxdropprob OUT FTP	0.90
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	10 Kbps
CBS FTP	20 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

O quadro 19 apresenta os resultados obtidos na simulação, considerando a alteração no número de fontes.

Quadro 19. Resultados do cenário 2

IC- Cenário 02 - Atraso			
Número de Fontes	**16**	**12**	**6**
	126,67	101,74	60,04
	130,47	110,10	59,10
	134,36	103,48	59,22
	131,00	88,00	59,02

	126,92	85,10	59,25
Média	129,88	97,68	59,33
Desvio Padrão	3,191336711	10,68104302	0,409731619
IC	125,9220674	84,42385956	58,81733256
	133,8459326	110,9441404	59,83466744

A figura 20 mostra a relação entre o número de fontes e o atraso. Foi utilizada a média do atraso das amostras acima listadas.

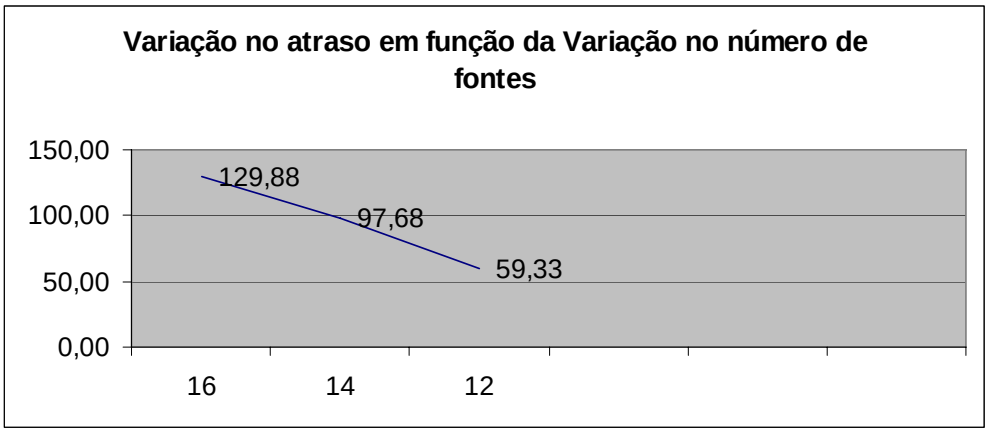

Figura 20. Relação entre atraso e número de fontes

6.4 JITTER

O j*itter* para aplicações de voz deve ser menor ou igual a 40 ms [ITU-T recomendação G.114]. O *jitter* considerado nesta avaliação foi o *jitter* médio fim- a -fim.

Do ponto de vista deste trabalho, entende-se inicialmente que o *jitter* pode vir a ser influenciado pelos parâmetros "*min_thresh*" e "*max_thresh*". Quanto maior o "*min_thresh*" e o "*max_thresh*", menor será o *jitter*, pois maiores seriam os buffers que compensariam o *jitter*.

O *jitter* também é decorrente de uma variação do tamanho dos buffers, por isso para que ele seja reduzido, o RED deve ser configurado com limiares do tamanho médio da fila não muito distantes para evitar uma grande variação.

A configuração de valores altos para o "*min_thresh*" e o "*max_thresh*"e a restrição do tráfego FTP, conforme mostrado no quadro 20, permitiu chegar a um *jitter* médio da simulação por fonte de voz inferior a 40 mseg como pode ser visualizado na figura. 22.

Cenário 1 :

Objetivo: Mostrar a influência dos parâmetros min_thresh e max_thresh sobre o *jitter*. Estes parâmetros do tráfego de voz foram os únicos alterados. Os demais foram mantidos fixos.

O quadro 20 mostra os valores dos parâmetros configurados neste cenário.

Quadro 20. valores de configuração de parâmetros de simulação do cenário 1

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes

Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**12 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	N bytes
Max_Thresh IN voz	N bytes
Min_Thresh OUT voz	N bytes
Max_Thresh OUT voz	N bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
Maxdropprob IN voz	0.01
Maxdropprob OUT voz	0.05
Maxdropprob IN FTP	0.10
Maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

O quadro 21 apresenta os resultados das simulações do cenário 1, obtidos para análise do *jitter* em relação aos parâmetros "*min_thresh*" e "*max_thresh*".

Quadro 21. Resultados do cenário 1

IC- Cenário 01 - Jitter				
Min_thresh, Max_thresh	**15,30/10,20**	**100,200/50,100**	**500,1000/250,500**	**3000,6000/1500,3000**
	45	40	25	25
	47	42	37	27
	35	35	23	35
	47	38	20	14
	35	37	33	16
Média	41,80	38,40	27,60	23,40
Desvio Padrão	6,260990337	2,701851217	7,127411872	8,561541917
IC	34,0272	35,04574644	18,75156804	12,77114587
	49,5728	41,75425356	36,44843196	34,02885413

A figura 21 mostra a relação entre o *jitter* e os parâmetros "*min_thresh*" e "*max_thresh*". Foi utilizada a média do *jitter* das amostras acima listadas.

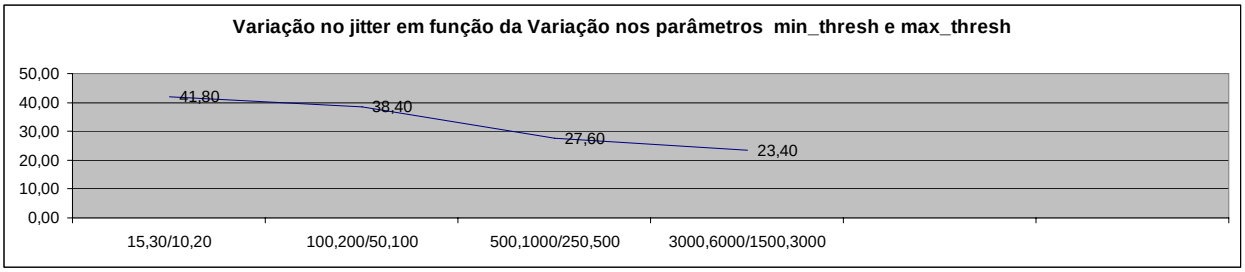

Figura 21. Relação entre jitter e os parâmetros "min_thresh" e "max_thresh"

A figura 22 ilustra o *jitter* resultante da simulação.

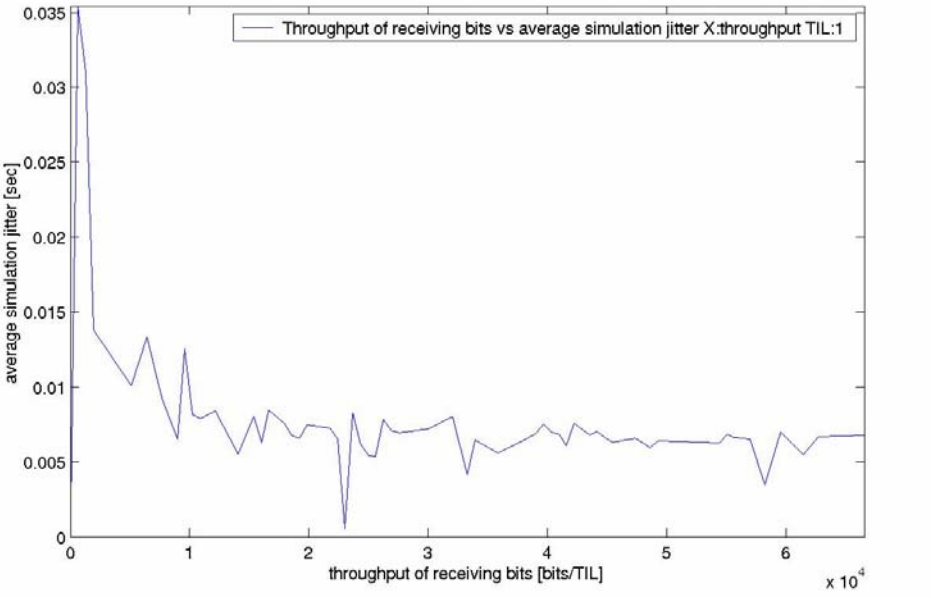

Figura 22. Jitter resultante

O quadro 22 mostra valores de configuração de parâmetros escolhidos com o objetivo de mostrar que o aumento do *jitter* está associado a uma redução dos valores dos parâmetros "*min_thresh*" e "*max_thresh*", assim como ao aumento do intervalo entre estes dois limites.

Quadro 22. Valores de configuração de parâmetros de simulação

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes

Número de Fontes	**12 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	4 bytes
Max_Thresh IN voz	40 bytes
Min_Thresh OUT voz	3 bytes
Max_Thresh OUT voz	30 bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
maxdropprob IN voz	0.01
maxdropprob OUT voz	0.05
maxdropprob IN FTP	0.10
maxdropprob OUT FTP	0.20
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	512 Kbps
CBS FTP	1500 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

A figura 23 mostra o *jitter* resultante da simulação e evidencia um *jitter* maior que aquele mostrado na figura 22, ratificando as expectativas em relação à simulação descrita no quadro 22.

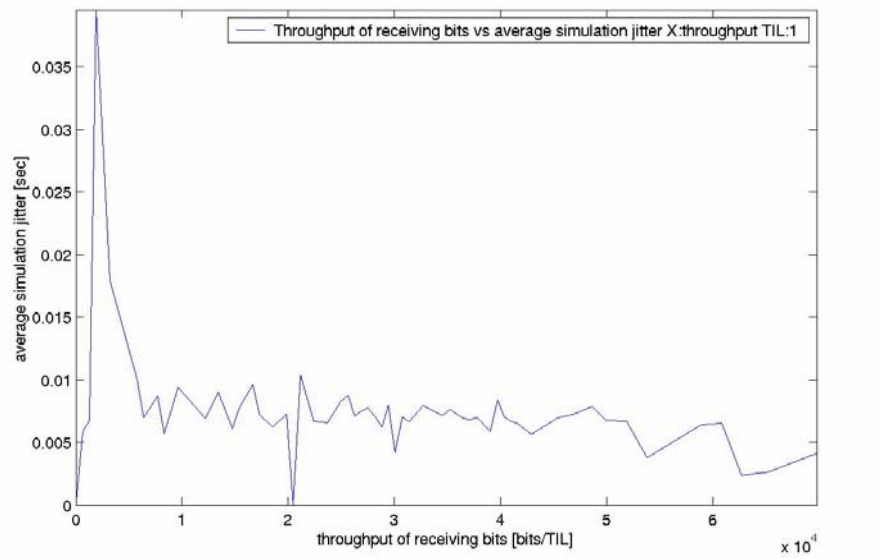

Figura 23. Jitter resultante da simulação descrita no quadro 22

6.5 UTILIZAÇÃO

O objetivo deste tópico é comentar sobre a utilização dos enlaces medida durante as simulações. O objetivo desta análise foi avaliar como os parâmetros de configuração afetam a utilização dos enlaces, o que sem dúvida, afeta a performance do modelo de rede utilizado nesta pesquisa.

A utilização dos links foi medida através da fórmula (n º bytes enviados / largura do link em bytes) *100, sendo considerado o tráfego total, ou seja, o tráfego VoIP e o tráfego FTP.

Acredita-se inicialmente que os parâmetros que afetam a utilização são o "*min_thresh*" e o "*max_thresh*". Quanto maior for o valor destes parâmetros, maior será o número de pacotes enfileirados ao invés de descartados e consequentemente maior será a utilização do enlace.

Os dados do quadro 23 mostram valores de configuração utilizados em uma simulação com o objetivo de verificar a expectativa descrita acima.

Quadro 23. Valores de configuração de parâmetros de simulação

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	1500 bytes
Número de Fontes	**12 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	3000 bytes
Max_Thresh IN voz	6000 bytes
Min_Thresh OUT voz	1500 bytes
Max_Thresh OUT voz	3000 bytes
Min_Thresh IN FTP	2 bytes
Max_Thresh IN FTP	4 bytes
Min_Thresh OUT FTP	1 bytes
Max_Thresh OUT FTP	2 bytes
maxdropprob IN voz	0.01
maxdropprob OUT voz	0.05
maxdropprob IN FTP	0.8
maxdropprob OUT FTP	0.9
Parâmetros do *Token Bucket*:	
CIR voz	768 Kbps
CBS voz	80 bytes
CIR FTP	10 Kbps

CBS FTP	20 bytes
Prioridade da fila de voz para o WRR	9
Prioridade da fila FTP para o WRR	1

A utilização está em torno de 92 % nos enlaces que ligam o nó 2 ao nó 3 e o nó 3 ao nó 4 (links de 1 Mbps que interligam o *backbone* aos roteadores de fronteira como pode ser observado na fugura 12). No enlace que liga o nó zero (gerador do tráfego de voz) ao roteador de entrada, o percentual de utilização foi de apenas 6 %, uma vez que o link é de 10 Mbps e a simulação só utiliza 16 fontes de voz a 64 kbps. O enlace que interliga o gerador FTP ao roteador de entrada teve uma utilização de 6,6 %, uma vez que o tráfego FTP foi restrito através da configuração de parâmetros do token e do RED. O enlace que liga o roteador de saída ao nó receptor teve uma utilização de 9 %, uma vez que tem uma largura de 10 Mbps e trafega o mesmo número de bytes que passa pelo núcleo da rede.

A utilização ideal seria, em torno de, 60 – 70 %. Uma forma óbvia de reduzir a utilização de um enlace seria reduzir o número de fontes ou a velocidade das mesmas ou aumentar a velocidade dos *links*. Uma forma de conseguir passar a quantidade de tráfego desejada de forma controlada e inteligente seria reduzir os valores do "*min_thresh*" e do "*max_thresh*". Entretanto, isto faria

com que o *jitter* e o percentual de perdas aumentasse o que prejudicaria bastante o tráfego de voz. Por este motivo, considerou-se como aceitável as utilizações encontradas durante as simulações e citadas no parágrafo anterior.

7 CONCLUSÕES E RECOMENDAÇÕES

Neste capítulo apresentam-se as conclusões obtidas no desenvolvimento deste estudo e também as recomendações para futuros trabalhos.

7.1 CONCLUSÕES

Os resultados referentes a perda, atraso e *jitter*, apresentados neste documento, foram coletados por canal de voz, sendo que todos os canais apresentaram comportamento equivalente e resultados muito próximos que permitiram chegar às conclusões descritas neste documento.

Vale ressaltar que, os parâmetros de simulação afetam as estatísticas do tráfego até um ponto que se pode chamar de eficiência máxima dos parâmetros. O efeito que os parâmetros possuem sobre as estatísticas da simulação são limitados, isto é, existe um valor limite ou de eficiência máxima.

O atraso, por exemplo, não seria infinito na simulação. Ele não é determinado apenas pelos parâmetros do RED, uma vez que os mesmos afetam o atraso dos pacotes, mas dependem do tamanho médio da fila que é determinada pelas taxas de chegada (taxa de geração das fontes) e as taxas de serviço (velocidade dos links), logo em tese não adianta alterar os parâmetros "*max_thresh*" e "*min_thresh*" além do valor da fila média calculado, pois chegou-se ao valor limite ou de eficiência máxima. Este fato pode ser observado no quadro 17, onde é mostrado que qualquer valor acima de 500, 1000, referentes ao "*min_thresh*" e "*max_thresh*" do tráfego de voz, não afeta de forma significativa o valor do atraso.

O mesmo ocorre com relação ao percentual de perdas. Por mais que os parâmetros do *token bucket* e do RED possam interferir no percentual de perdas, a velocidade dos links, a taxa de

geração das fontes e o número de fontes são parâmetros que podem vir a definir de forma irremediável o percentual de perdas.

Causa pouco impacto sobre os parâmetros de QoS , por exemplo, aumentar além do limite de eficiência máxima, os valores do max_thresh e min_thresh, pois o percentual de perdas, o atraso, o *jitter* e a utilização ficarão o mesmo, o que se justifica pelo própria definição do mecanismo de funcionamento do RED.

Este fato também foi observado com relação aos parâmetros do *token bucket*. Causa pouco impacto sobre os parâmetros de QoS, aumentar o CIR, além da velocidade de geração do tráfego da fonte, uma vez que não haverá sincronia entre a geração e o consumo dos *tokens*. A conseqüência será que alguns *tokens* serão gerados, mas não consumidos.

Para um cenário específico de simulação com 16 fontes de voz, o atraso médio não ultrapassou 120 mseg, sendo que o atraso máximo não ultrapassou 120 mseg, mesmo para valores bastante altos de "*min_thresh*" e "*max_thresh*" (3000, 6000)**.** O menor atraso obtido foi de 87 mseg mesmo para valores muito pequenos do "*min_thresh*" e "*max_thresh*" (15,30). Estes fatos ocorreram porque atingiu-se o limite de eficiência máxima dos parâmetros. Estes dados podem ser observados no quadro 17.

A partir das simulações executadas, constatou-se que deve-se configurar os valores de "*min_thresh*" e "*max_thresh*" do tráfego de voz altos o suficiente para garantir uma boa utilização, um valor de atraso aceitável por uma aplicação VoIP, um baixo *jitter* e um baixo percentual de perdas. Os parâmetros do *token bucket,* do RED e do tamanho do pacote devem ser configurados de forma a restringir o tráfego FTP e liberar recursos para o tráfego de voz, uma vez que o tráfego FTP não tem grandes exigências no que diz respeito aos parâmetros de QoS e

por rodar sobre o TCP, tende a tomar toda a banda existente, prejudicando o tráfego de voz concorrente. Durante as simulações, pôde-se verificar que o gráfico do tráfego entre *edge0* e *core0* é parecido com o gráfico do tráfego entre input1 e edge0(FTP), o que significa que o tráfego FTP procura se adaptar ao estado do enlace.

Pelo motivo exposto acima, deve-se configurar a prioridade do tráfego de voz com um valor maior que a do tráfego FTP, isto pode ser feito através da configuração do parâmetro peso do algoritmo WRR. Assim como deve-se restringir o tráfego FTP através da configuração de baixos valores para o "*min_thresh*", "*max_thresh*" (5 e 10; 3 e 6 bytes), CIR e CBS (110 Kbps e 512 bytes respectivamente) e um valor alto para a probabilidade máxima de descarte (0.10 e 0.20). O tamanho do pacote FTP também deve ser restrito, não podendo ser muito maior que o do tráfego VoIP para que o parâmetro do algoritmo WRR possa fazer efeito no que diz respeito a prioridade de acesso a recursos, como por exemplo, largura de banda. Por exemplo, sugere-se 80 bytes para o pacote de voz e 110 bytes para o pacote FTP.

Valores de configuração são recomendados na simulação descrita na seção "Recomendações".

7.2 RECOMENDAÇÕES

É claro que todos os aspectos de uma rede não podem ser simultaneamente melhorados. Deve-se balancear o projeto da rede para otimizar a vazão (*throughput*) dos dados, atraso total, *jitter* e perda. Não existem muitas regras – as prioridades e requisitos de uma dada situação irão ditar os compromissos apropriados.

Por motivos já exaustivamente comentados neste documento, o tráfego de voz deve ser priorizado. Por outro lado, deve-se tentar restringir, o mínimo possível, o tráfego FTP, afinal este tipo de tráfego também precisará dos recursos da rede para executar de forma eficiente. Ele só

não é prioritário. Consequentemente, deve-se buscar a configuração dos parâmetros do esquema *Diffserv* que priorize o tráfego de voz através do alcance dos limites dos valores dos parâmetros de QoS recomendados pelo ITU-T. Uma vez alcançado estes limites, não há mais necessidade de restrição do tráfego FTP.

Esta recomendação pode ser exemplificada quando considera-se a configuração do tamanho do pacote e da prioridade do tráfego FTP. Configurando o tamanho de um pacote FTP com 110 bytes, pode-se configurar os pesos do algoritmo WRR em uma relação 6:4 e manter o percentual de perdas do tráfego de voz em limites aceitáveis. Não justifica uma configuração do tipo 9:1, por exemplo.

Com relação aos parâmetros de QoS de um tráfego VoIP, pode-se recomendar que o *jitter* seja reduzido, o quanto possível, com a utilização de *buffers* de saída. O uso de buffers reduz o *jitter*, mas reduz a eficiência da banda, que é um recurso caro e normalmente limitado. Grandes *buffers* de saída reduzem o *jitter* fim- a- fim de qualquer origem, mas aumentam o atraso geral.

Na escolha de codificador, deve-se escolher um que minimize o atraso, o uso da largura de banda ou a carga do processamento. Pode-se tomar esta decisão em conjunto com o método de controle do *jitter*. Por exemplo, para controlar o aumento da largura de banda causado pelo uso de grandes buffers, pode-se escolher um codificador de baixa taxa de transmissão. Estas decisões e suas conseqüências devem ser analisadas pelo projetista da rede.

Um grande número de simulações foram feitas com o objetivo de determinar os valores ótimos dos vários parâmetros de tráfego. No entanto, todas estas análises dependem da caracterização da carga da rede que influencia diretamente no tamanho da fila.

Os valores dos parâmetros a serem escolhidos devem conciliar baixo atraso, baixo percentual de perdas, baixo *jitter* e, se possível, uma boa utilização de banda.

Conciliar estes parâmetros de QoS não é uma tarefa muito simples, uma vez que, em algumas situações, eles se tornam conflitantes, exigindo uma boa dose de bom senso para priorizar algum deles em detrimento de outros e uma boa dose de esforço para, em cada modelo de rede, conseguir determinar os valores ideais dos parâmetros de tráfego.

Pode-se citar como exemplo o *jitter* e o atraso. Para se ter um baixo *jitter*, deve-se configurar o "*min_thresh*" e "*max_thresh*" com valores altos, entretanto os altos valores dos mesmos contribuem para aumentar o atraso. Por outro lado, é melhor um alto atraso que um alto *jitter*, uma vez que o *jitter* já é um componente do atraso e portanto a tolerância a ele é realmente muito pequena.

Por outro lado, os parâmetros de QoS, além de sofrerem influência dos parâmetros de tráfego, são dependentes também da estrutura de rede existente no que diz respeito à quantidade e velocidade de nós geradores de tráfego e a velocidade dos enlaces, uma vez que quanto maior o tráfego existente na rede, maior é a probabilidade de se ter uma alta utilização, um alto atraso e um alto percentual de perdas.

O quadro 24 mostra uma sugestão de valores que garantiram que os parâmetros de QoS fossem atendidos. Os valores do "*min_thresh*" e "*max_thresh*" foram escolhidos porque permitiram que o descarte de pacotes seja forçosamente atenuado pela existência de uma fila muito grande e ao mesmo tempo fornecesse uma quantidade de *buffer* suficiente para compensar o *jitter* e manter o atraso em um limite aceitável. Os valores do CIR e CBS foram escolhidos porque viabilizaram que o descarte de pacotes de voz seja atenuado através de liberação de banda para este tráfego.

Os valores dos pesos da fila foram escolhidos com o objetivo de priorizar o atendimento ao tráfego de voz e a probabilidade de descarte porque permitiu reduzir a probabilidade de descarte em caso de congestionamento.

Quadro 24. Valores de configuração de parâmetros de simulação sugeridos

Tempo de Simulação	100 s
Parâmetros ON/OFF:	
Burst	400 ms = 0,4 s
Idle	600 ms = 0,6 s
Size	80 bytes
Rate	64 Kbps
UDPPacketSize	80 bytes
TCPPacketSize	110 bytes
Número de Fontes	**16 de voz e duas FTP**
Parâmetros RED:	
Numqueues	2
Numprec	2
MeanPktsize	80 bytes
Min_Thresh IN voz	100 bytes
Max_Thresh IN voz	200 bytes
Min_Thresh OUT voz	50 bytes
Max_Thresh OUT voz	100 bytes
Min_Thresh IN FTP	5 bytes
Max_Thresh IN FTP	10 bytes
Min_Thresh OUT FTP	3 bytes
Max_Thresh OUT FTP	6 bytes
maxdropprob IN voz	0.01
maxdropprob OUT voz	0.05
maxdropprob IN FTP	0.10
maxdropprob OUT FTP	0.20

Parâmetros do *Token Bucket*:	
CIR voz	1024 Kbps
CBS voz	80 bytes
CIR FTP	110 Kbps
CBS FTP	512 bytes
Prioridade da fila de voz para o WRR	6
Prioridade da fila FTP para o WRR	4

Trabalhos futuros poderiam utilizar outros modelos de simulação, variando a configuração da topologia estudada, além de implementar um modelo que executasse o serviço PHB-EF, analisando os resultados e comparando com os obtidos neste documento. Para implementar um modelo PHB-EF, seria necessário acrescentar classes ao pacote básico do NS, uma vez que o mesmo só implementa o PHB-AF.

REFERÊNCIAS

ARANGO, M; HUIET EMA, C. **Simple *gateway* Control Protocol (SGCP), Version 1.0**.1998

BLAKE, S.; BLACK, D.; CARLSON, M.; DAVIES, E.; WANG, Z. ; WEISS, W. **– An Architecture for Differentiated Services**. Internet RFC, rfc2475. Dezembro, 1998 .

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

CHWAN-HWA, Wu; IRWIN, J. David. **Emerging Multimedia Computer Communication Technologies**, Prentice-Hall, 1998.

CONSER (Collaborative Simulation for Education and Research) [online]. Disponível na Internet via URL: http://www.isi.edu/conser/index.html. Acesso em 28.02.2002.

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.

DAIGLE,N; LANFORD, J. D. **Models for Analysis of packet voice comunications systems**, IEE JSAC, pp 847-855, 1986

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

DIERKS, T; .ALLEN, C.**The TLS Protocol**, Version 1.0, IETF RFC 2246,1998

DOUSKALIS, Bill. **IP Telephony – The Integration of Robust VoIP Services**. Prentice-Hall, 2000. 317 p.

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

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

GOODE, Bur. – **Voice Over Internet Protocol (VoIP**). Proceedings of the IEEE, vol 90, no 9, setembro, 2002.

HUSTON, Geoff; FERGUSON, Paul. **Quality of Service – Delivering QoS on the Internet and in Corporate Networks**. Wiley Computer Publishing. 1998.

KEAGY, Scott**. Integrating Voice and Data Networks.** Cisco Press, 2000.

MARTINS, Joberto. **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 2000.

MINOLI, D. ; MINOLI, E. **Delivering Voice Over IP Networks**. Ed. John Wiley & Sons, 1998. 276 p.

OTcl (MIT Object Tool Command Language) [online]. Disponível na Internet via URL: http://OTcl-tclcl.sourceforge.net/OTcl/. Arquivo capturado em 28.02.2002.

PERKINS, M; EVANS, K; PASCAL,D; THORPE,K. **Characterizing the subjective performance of the ITU-T 8 kb/s speech coding algorithm – ITU-T G.729**, IEEE Commun. Vol. 35,pp 74-81, September 1997

PORTNOI. Marcos; ARAÚJO. Rafael G. B. de. **Uma Visão Geral da Ferramenta de Simulação de Redes – Network Simulator (NS)**. Salvador, UNIFACS, 2003.

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.

REZENDE, J. F; ZIVIANI Artur. **Tráfego de Voz em um ambiente de dieferenciação de serviços na Internet**. Rio de Janeiro, 1999.Disponível em http://www.gta.ufrj.br.

SAMAN (Simulation Augmented by Measurement and Analysis for Networks). Disponível na Internet via URL: http://www.isi.edu/saman/index.html. Acesso em 28.02.2002.

STEVENS, R. W. **TCP/IP Illustrated - The Protocols**, Vol. 1, Addison-Wesley, 1994.

THE NETWORK SIMULATOR - ns-2 [online]. Disponível na Internet via URL: http://www.isi.edu/nsnam. Acesso em 28.02.2002.

THOMAS, A. M. **Deploying IP Multicast in the Enterprise**, Prentice-Hall, 1997.

THOMAS, B; FELDMAN, N; DOOLAN, P; ANDERSON, L; FREDETTE, A . **LDP Specification**, Junho 1999.

TRACEGRAPH [online] Disponível na Internet via URL: http://www.geocities.com/tracegraph. Arquivo capturado em 28.02.2002.

WHITE, Paul P. – **RSVP and Integrated Services in the Internet: A Tutorial**. IEEE Communication Magazine, maio 1997

ANEXO: CÓDIGO DO SCRIP USADO NAS SIMULAÇÕES

```
#           VoIP (Voice over IP) Project and Simulation

#  --- Topology
#
# input0 ----- edge0 -------- core0
#  (0)         (2)           (3)
#             /              |
#           /                |

# input1 - /                 edge1 ------- sink0
#  (1)                       (4)         (5)
#
#
# VoIP generators attach to input0
# FTP generators attach to input1

#################################################################
#
#  --- Generic Procedures Section
#
# Procedure exponential traffic generator + sink, using TCP transport
# creates exponential traffic generator objects and sink objects, and connects them
# size - bytes
# burst - seconds
# idle - seconds
# rate - bits per second (bps)
proc attach-expoo-traffic_tcp {nodein nodeout flowID size burst idle rate class} {

            #get simulator instance
            set ns [Simulator instance]

      #Create TCP agent and TCPSink, attach them to nodes and link them
            set connection_list [$ns create-connection-list TCP $nodein TCPSink $nodeout $flowID]
            set source [lindex $connection_list 0]
```

```
        set sink [lindex $connection_list 1]
        $source set class_ $class

    #Create Expoo traffic agent and set its configuration parameters
        set traffic [new Application/Traffic/Exponential]
    $traffic set packetSize_ $size
    $traffic set burst_time_ $burst
    $traffic set idle_rate_ $idle
    $traffic set rate_ $rate

    #Attach traffic source to traffic generator
        $traffic attach-agent $source
        return [concat $traffic $sink];          #return a list of $traffic and $sink object references
}
###
# Procedure exponential traffic generator + sink, using UDP transport
# creates exponential traffic generator objects and sink objects, and connects them

proc attach-expoo-traffic_udp {nodein nodeout flowID size burst idle rate class} {

        #get simulator instance
        set ns [Simulator instance]

    #Create UDP agent and Sink, attach them to nodes and link them
        set connection_list [$ns create-connection-list UDP $nodein Null $nodeout $flowID]
        set source [lindex $connection_list 0]
        set sink [lindex $connection_list 1]
        $source set class_ $class

    #Create Expoo traffic agent and set its configuration parameters
        set traffic [new Application/Traffic/Exponential]
    $traffic set packetSize_ $size
    $traffic set burst_time_ $burst
    $traffic set idle_rate_ $idle
    $traffic set rate_ $rate
```

```
    #Attach traffic source to traffic generator
        $traffic attach-agent $source
        return [concat $traffic $sink];          #return a list of $traffic and $sink object references
}
###
#Procedure FTP application + sink
# creates FTP generator objects and sink objects, and connects them
proc attach-ftp-traffic {nodein nodeout packetSize flowID class} {

        #get simulator instance
        set ns [Simulator instance]

    #Create TCP agent and TCPSink, attach them to nodes and link them
        set connection_list [$ns create-connection-list TCP $nodein TCPSink $nodeout $flowID]
        set source [lindex $connection_list 0]
        set sink [lindex $connection_list 1]
        $source set packetSize_ $packetSize
        $source set class_ $class

    #Create FTP traffic agent
    set traffic [new Application/FTP]

    #Attach traffic source to traffic generator
        $traffic attach-agent $source
        return [concat $traffic $sink];          #return a list of $traffic and $sink object references
}

###
#Procedure Print dsRED queue statistics
proc dsRED-printstats {} {
        global q_edge0core0 q_core0edge1

        #get simulator instance
        set ns [Simulator instance]

        #Set time after which procedure should be called again (seconds)
```

```
        set time 10

        #Print dsRED queue statistics
        puts "\n\n----------\nEdge0 to Core0 queue statistics at [$ns now] sec:"
        $q_edge0core0 printStats

        puts "\nCore0 to Edge1 queue statistics at [$ns now] sec:"
        $q_core0edge1 printStats

        #Re-schedule procedure
        $ns at [expr [$ns now]+$time] "dsRED-printstats"
}
###

# --- End Generic Procedures
#################################################################

#################################################################
# --- Begin Simulation Setup

# --- Random Numbers and Seed
#Seed default random number generator with current time (for Expoo traffic)
global defaultRNG
$defaultRNG seed 0

# --- Create simulator object
set ns [new Simulator]

# --- Set Simulation Time (in seconds)
set MAX_TIME 10

# --- Open files for tracing and set-up simulation and nam tracing
set nf [open out.nam w]
set tr [open out.tr w]
$ns namtrace-all $nf
$ns trace-all $tr
```

```
# --- Define color index (class color)
# list of colors available at ~ns/man/mann/colors.n
$ns color 0 red
$ns color 1 blue
$ns color 2 chocolate
$ns color 3 yellow
$ns color 4 green
$ns color 5 tan
$ns color 6 gold
$ns color 7 black
$ns color 8 white
$ns color 9 darkblue
$ns color 10 cyan
$ns color 11 magenta
$ns color 12 orange
$ns color 13 bisque
$ns color 14 purple
$ns color 15 coral
$ns color 16 grey
$ns color 17 khaki
$ns color 18 AntiqueWhite
$ns color 19 aquamarine
$ns color 20 azure
$ns color 21 DarkSalmon
$ns color 22 DarkSeaGreen
$ns color 23 firebrick

# --- Set packet size for RED calculations and traffic generators
set UDPpacketSize 576
set TCPpacketSize 1300

# --- Set TokenBucket parameters
set cir0 1024000;#committed information rate (bps)
set cbs0 2000;                  #committed burst size of bucket (bytes)
set cir1 64000;
```

```
set cbs1 3000;

# --- Set RED drop parameters for queues
# [variable]physical_queue,virtual_queue
# physical queue 0, IN queue
set min_thresh00 20
set max_thresh00 40
set max_drop_prob00 0.02

# physical queue 0, OUT queue
set min_thresh01 10
set max_thresh01 20
set max_drop_prob01 0.10

# physical queue 1, IN queue
set min_thresh10 20
set max_thresh10 40
set max_drop_prob10 0.02

# physical queue 1, OUT queue
set min_thresh11 10
set max_thresh11 20
set max_drop_prob11 0.10



# --- Set WRR scheduling weights for physical queues
#     the sum of individual weights is supposed to be 10
set phy_queue0_weight 5
set phy_queue1_weight 5

# --- End Simulation Setup
#############################################################

#############################################################
# --- Create Network Topology
```

```
# --- create nodes
puts -nonewline "Creating Topology..."
set input0 [$ns node];          #traffic enters here
set input1 [$ns node];          #second traffic enters here
set edge0 [$ns node];           #generic router
set core0 [$ns node];           #generic router
set edge1 [$ns node];           #generic router
set sink0 [$ns node];           #data sink

# --- create links, queues and connect nodes
$ns duplex-link $input0 $edge0 10Mb 10ms DropTail
$ns duplex-link $input1 $edge0 10Mb 10ms DropTail

$ns simplex-link $edge0 $core0 1Mb 10ms dsRED/edge
$ns simplex-link $core0 $edge0 1Mb 10ms dsRED/core

$ns simplex-link $core0 $edge1 1Mb 10ms dsRED/core
$ns simplex-link $edge1 $core0 1Mb 10ms dsRED/edge

$ns duplex-link $edge1 $sink0 10Mb 10ms DropTail

# --- define shapes and colors of nodes (for nam)
$input0 color gold; $input0 shape box; $input0 label input0
$input1 color gold; $input1 shape box; $input1 label input1
$edge0 color blue; $edge0 shape box; $edge0 label edge0; $edge0 label-at top
$core0 color blue; $core0 shape circle; $core0 label core0; $core0 label-at top
$edge1 color blue; $edge1 shape box; $edge1 label edge1; $edge1 label-at bottom
$sink0 color gold; $sink0 shape box; $sink0 label sink0

# --- set visual orientation of links for nam
$ns duplex-link-op $input0 $edge0 queuePos 0.5
$ns duplex-link-op $input1 $edge0 queuePos 0.5
$ns duplex-link-op $edge0 $core0 queuePos 0.5
$ns duplex-link-op $core0 $edge1 queuePos 0.5
$ns duplex-link-op $edge1 $sink0 queuePos 0.5
```

```
$ns duplex-link-op $input0 $edge0 orient right
$ns duplex-link-op $input1 $edge0 orient right-up
$ns duplex-link-op $edge0 $core0 orient right
$ns duplex-link-op $core0 $edge1 orient down
$ns duplex-link-op $edge1 $sink0 orient right

puts "    Done."

# --- End Creating Topology
#################################################################

#################################################################
# --- Setup RED queues and parameters

set q_edge0core0 [[$ns link $edge0 $core0] queue]
set q_edge1core0 [[$ns link $edge1 $core0] queue]
set q_core0edge0 [[$ns link $core0 $edge0] queue]
set q_core0edge1 [[$ns link $core0 $edge1] queue]

# Set dsRED parameters from Edge0 to Core0

$q_edge0core0 meanPktSize $UDPpacketSize
$q_edge0core0 setSchedularMode WRR; #Weighted Round Robin scheduling mode for physical queues 0 and 1
$q_edge0core0 addQueueWeights 0 $phy_queue0_weight; #set weight for physical queue 0
$q_edge0core0 addQueueWeights 1 $phy_queue1_weight; #set weight for physical queue 1
$q_edge0core0 set numQueues_ 2
$q_edge0core0 setNumPrec 2
$q_edge0core0 addPolicyEntry [$input0 id] [$sink0 id] TokenBucket 10 $cir0 $cbs0; #voip
$q_edge0core0 addPolicyEntry [$input1 id] [$sink0 id] TokenBucket 20 $cir1 $cbs1; #ftp
$q_edge0core0 addPolicerEntry TokenBucket 10 11
$q_edge0core0 addPolicerEntry TokenBucket 20 21
$q_edge0core0 addPHBEntry 10 0 0
$q_edge0core0 addPHBEntry 11 0 1
$q_edge0core0 addPHBEntry 20 1 0
$q_edge0core0 addPHBEntry 21 1 1
$q_edge0core0 configQ 0 0 $min_thresh00 $max_thresh00 $max_drop_prob00; #physical_queue virtual_queue min_thresh max_thresh max_drop_prob
```

$q_edge0core0 configQ 0 1 $min_thresh01 $max_thresh01 $max_drop_prob01

$q_edge0core0 configQ 1 0 $min_thresh10 $max_thresh10 $max_drop_prob10; #physical_queue virtual_queue min_thresh max_thresh max_drop_prob

$q_edge0core0 configQ 1 1 $min_thresh11 $max_thresh11 $max_drop_prob11

Set dsRED parameters from Edge1 to Core0

$q_edge1core0 meanPktSize $UDPpacketSize

$q_edge1core0 setSchedularMode WRR; #Weighted Round Robin scheduling mode for physical queues 0 and 1

$q_edge1core0 addQueueWeights 0 $phy_queue0_weight; #set weight for physical queue 0

$q_edge1core0 addQueueWeights 1 $phy_queue1_weight; #set weight for physical queue 1

$q_edge1core0 set numQueues_ 2

$q_edge1core0 setNumPrec 2

$q_edge1core0 addPolicyEntry [$sink0 id] [$input0 id] TokenBucket 11 $cir0 $cbs0; #voip

$q_edge1core0 addPolicyEntry [$sink0 id] [$input1 id] TokenBucket 20 $cir1 $cbs1; #ftp

$q_edge1core0 addPolicerEntry TokenBucket 10 11

$q_edge1core0 addPolicerEntry TokenBucket 20 21

$q_edge1core0 addPHBEntry 10 0 0

$q_edge1core0 addPHBEntry 11 0 1

$q_edge1core0 addPHBEntry 20 1 0

$q_edge1core0 addPHBEntry 21 1 1

$q_edge1core0 configQ 0 0 $min_thresh00 $max_thresh00 $max_drop_prob00; #physical_queue virtual_queue min_thresh max_thresh max_drop_prob

$q_edge1core0 configQ 0 1 $min_thresh01 $max_thresh01 $max_drop_prob01

$q_edge1core0 configQ 1 0 $min_thresh10 $max_thresh10 $max_drop_prob10; #physical_queue virtual_queue min_thresh max_thresh max_drop_prob

$q_edge1core0 configQ 1 1 $min_thresh11 $max_thresh11 $max_drop_prob11

Set DS RED parameters from Core0 to Edge0

$q_core0edge0 meanPktSize $UDPpacketSize

$q_core0edge0 setSchedularMode WRR; #Weighted Round Robin scheduling mode for physical queues 0 and 1

$q_core0edge0 addQueueWeights 0 $phy_queue0_weight; #set weight for physical queue 0

$q_core0edge0 addQueueWeights 1 $phy_queue1_weight; #set weight for physical queue 1

$q_core0edge0 set numQueues_ 2

$q_core0edge0 setNumPrec 2

$q_core0edge0 addPHBEntry 10 0 0

$q_core0edge0 addPHBEntry 11 0 1

$q_core0edge0 addPHBEntry 20 1 0

```
$q_core0edge0 addPHBEntry 21 1 1
$q_core0edge0 configQ 0 0 $min_thresh00 $max_thresh00 $max_drop_prob00
$q_core0edge0 configQ 0 1 $min_thresh01 $max_thresh01 $max_drop_prob01
$q_core0edge0 configQ 1 0 $min_thresh10 $max_thresh10 $max_drop_prob10
$q_core0edge0 configQ 1 1 $min_thresh11 $max_thresh11 $max_drop_prob11

# Set DS RED parameters from Core0 to Edge1
$q_core0edge1 meanPktSize $UDPpacketSize
$q_core0edge1 setSchedularMode WRR; #Weighted Round Robin scheduling mode for physical queues 0 and 1
$q_core0edge1 addQueueWeights 0 $phy_queue0_weight; #set weight for physical queue 0
$q_core0edge1 addQueueWeights 1 $phy_queue1_weight; #set weight for physical queue 1
$q_core0edge1 set numQueues_ 2
$q_core0edge1 setNumPrec 2
$q_core0edge1 addPHBEntry 10 0 0
$q_core0edge1 addPHBEntry 11 0 1
$q_core0edge1 addPHBEntry 20 1 0
$q_core0edge1 addPHBEntry 21 1 1
$q_core0edge1 configQ 0 0 $min_thresh00 $max_thresh00 $max_drop_prob00
$q_core0edge1 configQ 0 1 $min_thresh01 $max_thresh01 $max_drop_prob01
$q_core0edge1 configQ 1 0 $min_thresh10 $max_thresh10 $max_drop_prob10
$q_core0edge1 configQ 1 1 $min_thresh11 $max_thresh11 $max_drop_prob11

# Print dsRED tables and parameters
$q_edge0core0 printPolicyTable
$q_edge0core0 printPolicerTable
puts "RED parameters:"
puts "          min_thresh  max_thresh  max_drop_probability"
puts [format "Queue 00:     %3d         %3d         %1.3f" $min_thresh00 $max_thresh00 $max_drop_prob00]
puts [format "Queue 01:     %3d         %3d         %1.3f" $min_thresh01 $max_thresh01 $max_drop_prob01]
puts [format "Queue 10:     %3d         %3d         %1.3f" $min_thresh10 $max_thresh10 $max_drop_prob10]
puts [format "Queue 11:     %3d         %3d         %1.3f" $min_thresh11 $max_thresh11 $max_drop_prob11]
puts "\n"

# --- End setting up dsRED queues
#############################################################
```

```
#############################################################
# --- Create Network and Traffic agents, and attach them to nodes
# current setup:  16 expoo (voip) and 2 ftp generators

# --- create exponential generator and sink
# and link them (nodein nodeout flowID size burst idle rate class)
# the parameter 'class' also defines the color in nam

puts "Creating Traffic Generators..."

#voipgenerator0
puts "   voipgenerator0..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 0 $UDPpacketSize 0.4s 0.6s 64k 0]; #returns object references
set voipgenerator0 [lindex $traffic_sink_list 0]
set voipsink0 [lindex $traffic_sink_list 1]

#voipgenerator1
puts "   voipgenerator1..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 1 $UDPpacketSize 0.4s 0.6s 64k 1]; #returns object references
set voipgenerator1 [lindex $traffic_sink_list 0]

#voipgenerator2
puts "   voipgenerator2..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 2 $UDPpacketSize 0.4s 0.6s 64k 2]; #returns object references
set voipgenerator2 [lindex $traffic_sink_list 0]

#voipgenerator3
puts "   voipgenerator3..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 3 $UDPpacketSize 0.4s 0.6s 64k 3]; #returns object references
set voipgenerator3 [lindex $traffic_sink_list 0]

#voipgenerator4
puts "   voipgenerator4..."
```

```
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 4 $UDPpacketSize 0.4s 0.6s 64k 4]; #returns object references
set voipgenerator4 [lindex $traffic_sink_list 0]

#voipgenerator5
puts "   voipgenerator5..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 5 $UDPpacketSize 0.4s 0.6s 64k 5]; #returns object references
set voipgenerator5 [lindex $traffic_sink_list 0]

#voipgenerator6
puts "   voipgenerator6..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 6 $UDPpacketSize 0.4s 0.6s 64k 6]; #returns object references
set voipgenerator6 [lindex $traffic_sink_list 0]

#voipgenerator7
puts "   voipgenerator7..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 7 $UDPpacketSize 0.4s 0.6s 64k 7]; #returns object references
set voipgenerator7 [lindex $traffic_sink_list 0]

#voipgenerator8
puts "   voipgenerator8..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 8 $UDPpacketSize 0.4s 0.6s 64k 8]; #returns object references
set voipgenerator8 [lindex $traffic_sink_list 0]

#voipgenerator9
puts "   voipgenerator9..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 9 $UDPpacketSize 0.4s 0.6s 64k 9]; #returns object references
set voipgenerator9 [lindex $traffic_sink_list 0]

#voipgenerator10
puts "   voipgenerator10..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 10 $UDPpacketSize 0.4s 0.6s 64k 10]; #returns object references
set voipgenerator10 [lindex $traffic_sink_list 0]
```

```
#voipgenerator11
puts "   voipgenerator11..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 11 $UDPpacketSize 0.4s 0.6s 64k 11]; #returns object references
set voipgenerator11 [lindex $traffic_sink_list 0]

#voipgenerator12
puts "   voipgenerator12..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 12 $UDPpacketSize 0.4s 0.6s 64k 12]; #returns object references
set voipgenerator12 [lindex $traffic_sink_list 0]

#voipgenerator13
puts "   voipgenerator13..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 13 $UDPpacketSize 0.4s 0.6s 64k 13]; #returns object references
set voipgenerator13 [lindex $traffic_sink_list 0]

#voipgenerator14
puts "   voipgenerator14..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 14 $UDPpacketSize 0.4s 0.6s 64k 14]; #returns object references
set voipgenerator14 [lindex $traffic_sink_list 0]

#voipgenerator15
puts "   voipgenerator15..."
set traffic_sink_list [attach-expoo-traffic_udp $input0 $sink0 15 $UDPpacketSize 0.4s 0.6s 64k 15]; #returns object references
set voipgenerator15 [lindex $traffic_sink_list 0]

#puts "   Done."

#FTP generator 0
puts "   ftpgenerator0..."
set traffic_sink_list [attach-ftp-traffic $input1 $sink0 $TCPpacketSize 16 16]; #returns object references
set ftpgenerator0 [lindex $traffic_sink_list 0]
set ftpsink0 [lindex $traffic_sink_list 1]
```

```
#FTP generator 1
puts "   ftpgenerator1..."
set traffic_sink_list [attach-ftp-traffic $input1 $sink0 $TCPpacketSize 17 17]; #returns object references
set ftpgenerator1 [lindex $traffic_sink_list 0]

puts "Done."

# --- End creating traffic and network agents
#################################################################


#################################################################
# --- Schedule Section
$ns at 0.0 "puts \"Begin Simulation...   Simulation Time: $MAX_TIME s.\""
$ns at 0.0 "$voipgenerator0 start"
$ns at 0.0 "$voipgenerator1 start"
$ns at 0.0 "$voipgenerator2 start"
$ns at 0.0 "$voipgenerator3 start"
$ns at 0.0 "$voipgenerator4 start"
$ns at 0.0 "$voipgenerator5 start"
$ns at 0.0 "$voipgenerator6 start"
$ns at 0.0 "$voipgenerator7 start"
$ns at 0.0 "$voipgenerator8 start"
$ns at 0.0 "$voipgenerator9 start"
$ns at 0.0 "$voipgenerator10 start"
$ns at 0.0 "$voipgenerator11 start"
$ns at 0.0 "$voipgenerator12 start"
$ns at 0.0 "$voipgenerator13 start"
$ns at 0.0 "$voipgenerator14 start"
$ns at 0.0 "$voipgenerator15 start"
$ns at 0.0 "$ftpgenerator0 start"
$ns at 0.0 "$ftpgenerator1 start"

$ns at 0.0 "dsRED-printstats"
```

```
$ns at $MAX_TIME "$voipgenerator0 stop"
$ns at $MAX_TIME "$voipgenerator1 stop"
$ns at $MAX_TIME "$voipgenerator2 stop"
$ns at $MAX_TIME "$voipgenerator3 stop"
$ns at $MAX_TIME "$voipgenerator4 stop"
$ns at $MAX_TIME "$voipgenerator5 stop"
$ns at $MAX_TIME "$voipgenerator6 stop"
$ns at $MAX_TIME "$voipgenerator7 stop"
$ns at $MAX_TIME "$voipgenerator8 stop"
$ns at $MAX_TIME "$voipgenerator9 stop"
$ns at $MAX_TIME "$voipgenerator10 stop"
$ns at $MAX_TIME "$voipgenerator11 stop"
$ns at $MAX_TIME "$voipgenerator12 stop"
$ns at $MAX_TIME "$voipgenerator13 stop"
$ns at $MAX_TIME "$voipgenerator14 stop"
$ns at $MAX_TIME "$voipgenerator15 stop"
$ns at $MAX_TIME "$ftpgenerator0 stop"
$ns at $MAX_TIME "$ftpgenerator1 stop"

$ns at [expr $MAX_TIME + 2.0] "finish";             #ends simulation 2 seconds after generators stop

# --- End Schedule
#################################################################

#################################################################
# --- Finish procedure and final housekeeping
# --- analyze trace files and print statistics
proc finish {} {
        global ns nf tr
        global MAX_TIME
        $ns flush-trace

        #Separate flows into distinct trace files
        set awkCode {
           {
                if ($8 == 0) {
```

```
            print $0 > "outflow0.tr"
}
    else if ($8 == 1) {
        print $0 > "outflow1.tr"
}
    else if ($8 == 2) {
        print $0 > "outflow2.tr"
}
    else if ($8 == 3) {
        print $0 > "outflow3.tr"
}
    else if ($8 == 4) {
        print $0 > "outflow4.tr"
}
    else if ($8 == 5) {
        print $0 > "outflow5.tr"
}
    else if ($8 == 6) {
        print $0 > "outflow6.tr"
}
    else if ($8 == 7) {
        print $0 > "outflow7.tr"
}
    else if ($8 == 8) {
        print $0 > "outflow8.tr"
}
    else if ($8 == 9) {
        print $0 > "outflow9.tr"
}
    else if ($8 == 10) {
        print $0 > "outflow10.tr"
}
    else if ($8 == 11) {
        print $0 > "outflow11.tr"
}
    else if ($8 == 12) {
```

```
                print $0 > "outflow12.tr"
            }
                else if ($8 == 13) {
                print $0 > "outflow13.tr"
            }
                else if ($8 == 14) {
                print $0 > "outflow14.tr"
            }
                else if ($8 == 15) {
                print $0 > "outflow15.tr"
            }
                else if ($8 == 16) {
                print $0 > "outflow16.tr"
            }
                else if ($8 == 17) {
                print $0 > "outflow17.tr"
            }
              }
            }

            exec awk $awkCode out.tr

            #close trace files
            close $nf
            close $tr

            puts "End Simulation."
            # --- start nam
            exec nam out.nam &
            exit 0; #exit now
}

# --- End Finish procedure and housekeeping
#################################################################

#################################################################
```

```
# --- Run Simulation
$ns run

#################################################################
```