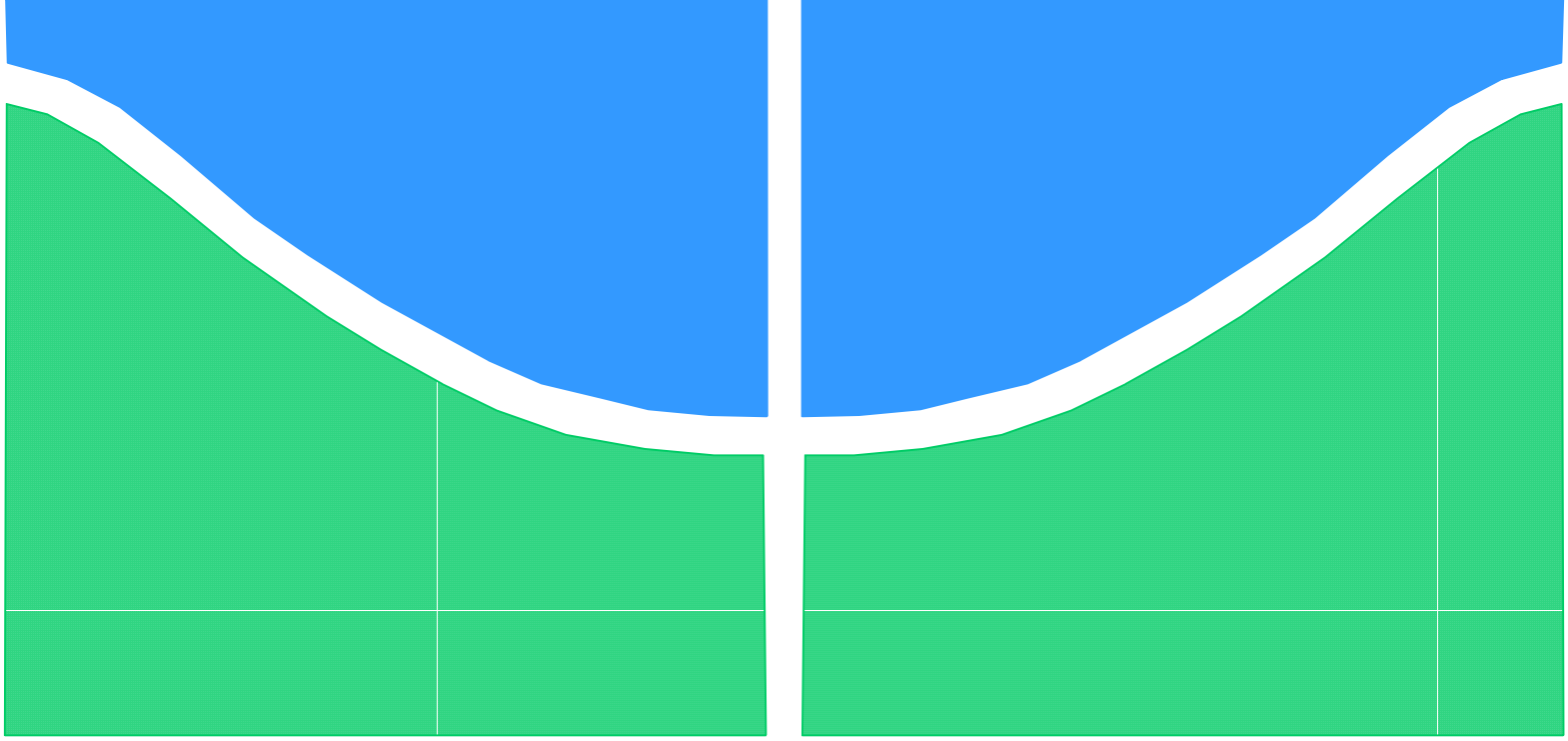

TRABALHO DE GRADUAÇÃO

GERÊNCIA ADAPTATIVA DE TRÁFEGO VOIP UTILIZANDO ARQUITETURA DIFFSERV

Igor Pupe Rocha
Victor Diego Medeiros Lino

Brasília, dezembro de 2009

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

GERÊNCIA ADAPTATIVA DE TRÁFEGO VOIP UTILIZANDO ARQUITETURA DIFFSERV

Igor Pupe Rocha
Victor Diego Medeiros Lino

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Msc. Leandro Vaguetti (UnB/DF) (Orientador) _____

Prof. Msc. Cláudio de Castro Monteiro (IFTO/TO) _____

Prof. Msc. Tiago Trindade (UnB/DF) _____

FICHA CATALOGRÁFICA

LINO, VICTOR DIEGO MEDEIROS
ROCHA, IGOR PUPE
Gerência adaptativa de tráfego VoIP utilizando arquitetura DiffServ, [Distrito Federal] 2009. xvii, 70p., 210 x 297 mm (FT/UnB, Engenheiro, Redes de Comunicação, 2009).
Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.
1.Qualidade de Serviço 2.VoIP
I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LINO, VICTOR DIEGO MEDEIROS e ROCHA, IGOR PUPE, (2009). Gerência adaptativa de tráfego VoIP utilizando arquitetura DiffServ. Trabalho de Graduação em Engenharia de Redes de Comunicação, Publicação FT.TG-nº , Faculdade de Tecnologia , Universidade de Brasília, Brasília, DF, 70p.

CESSÃO DE DIREITOS

Dedicatória(s)

Dedico esse trabalho à pessoa que sempre esteve ao meu lado em pensamento e oração durante esses cinco anos de faculdade. Que me aconselhou em cada passo a ser dado e sempre me apoiou, independentemente da decisão tomada. Que sempre me mostrou o lado bom das situações difíceis e é sempre um grande motivo para que eu busque e alcance meus objetivos. Toda a nossa luta valeu a pena mãe! Essa conquista dedico à você.

Victor Medeiros

Primeiramente, dedico este trabalho a Deus que está sempre presente em minha vida. A minha família, que em todos os momentos me deu toda a base e suporte necessários para a conclusão de mais esta etapa na minha vida. Também aos colegas de curso, que fizeram o dia a dia da universidade divertido.

Igor Rocha

Agradecimentos

Agradeço a Deus por sempre estar presente me dando todo o suporte espiritual. Ao meu pai, por me incentivar e estar sempre ao meu lado. A minha mãe, minha maior mestra, que me ensina com seu amor. Ao meu irmão, grande exemplo de ser humano, e a minha irmã, amiga de todas as horas. Agradeço ao professor Leandro Vaguetti por sua orientação e ensinamentos. Ao meu colega de projeto, Victor, que tanto se empenhou para o sucesso deste projeto. Por fim, a todos os amigos, pelo companherismo e apoio.

Igor Rocha

Agradeço primeiramente à Deus, por ter me coberto de bênçãos por todos esses anos.
A minha mãe, Lindalva, pelo amor incondicional.
A minha irmã, Jéssica, pela compreensão e carinho.
A minha tia, Faidma, pela generosidade e compaixão.
A meus tios Abenílio e Lúcia, por serem como uma verdadeira família para mim.
A meus tios Lusimar, Noêmia, Ana e Maria, pelo apoio e incentivo.
A família Ocius e agregados, por fazerem meus dias mais cômicos e fáceis de lidar.
A meu companheiro de projeto, Igor Pupe, pelo empenho e dedicação à esse projeto.
Ao nosso orientador, Leandro Vaguetti, pela oportunidade e pelos conselhos.
Ao professor Cláudio Monteiro, pela atenção e ajuda em momentos complicados desse projeto.
A todos meus amigos e familiares, sem vocês nada disso seria possível.

Victor Medeiros

RESUMO

A cada dia mais serviços são oferecidos através da Internet, como comércio eletrônico, *e-mails*, *Internet banking*, *streams* de áudio e vídeo. Porém, o modelo atual de enfileiramento usado na Internet é o Best Effort, que trata todos os tipos de tráfego da mesma maneira, o que não atende as necessidades de todos os aplicativos. Surgiram então arquiteturas para proverem qualidade de serviço para essas aplicações. O VoIP (*Voice over IP*) é um serviço que já assume uma parcela considerável do mercado, principalmente no meio corporativo e possui requisitos que precisam ser atendidos para o estabelecimento de uma boa conversação. Neste projeto foi realizado um estudo da tecnologia VoIP e suas vulnerabilidades quanto as caraterísticas de jitter, *delay* e perda de pacotes. Foram analisadas as arquiteturas que provêm QoS e posteriormente foi decidido por se usar a arquitetura DiffServ através da ferramenta ALTQ. Foram realizados testes com o propósito de mostrar o comportamento do tráfego VoIP nas diferentes disciplinas de encaminhamento DiffServ e, ao final, foi desenvolvido um *shell script* adaptativo capaz de analisar a qualidade do tráfego VoIP e o adaptar à disciplina DiffServ necessária para um boa qualidade de experiência auditiva.

Palavras Chave: VoIP, SIP, QoS, Diffserv, ALTQ, Adaptativo.

ABSTRACT

Every day, more services are ofered through the Internet, as e-commerce, e-mail, Internet banking, audio and video streams. However, the current queing model used on Internet is the Best Effort, dealing with all kinds of traffic in the same way, wich does not meet the needs of all applications. Several architectures emerged to provide quality of service to these applications. VoIP (Voice over IP) is a service which has already acquired a considerable share of the market, especially in the corporate segment and have requirements that must be met to establish a good conversation. In this project, a study of VoIP technology and its vulnerabilities such as jitter, *delay* and packet loss, was made. We analyzed the architectures that provide QoS and later it was decided to use the implementation of DiffServ through the ALTQ tool for this purpose. Tests were conducted in order to show the behavior of VoIP traffic for different kinds of DiffServ forward disciplines and, in the end, we developed a *shell script* that can analyze the quality of VoIP traffic and adapt to the DiffServ discipline required for a quality of listening experience.

Key-Words: VoIP, SIP, QoS, Diffserv, ALTQ, Adaptative.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

AF	Assured Forward
ALTQ	Alternate Queuing
B2BUA	Back to Back User Agent
BE	Best Effort
CBQ	Class Based Queuing
CODEC	Coder/Decoder (Codificador/Decodificador)
CS	Class Selector
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DS	Differentiated Services
DSCP	Differential Services Codepoint
EF	Expedited Forward
GIRS	Gerência Integrada de Redes e Serviços
HFSC	Hierarchical Fair Service Curve
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ITU	International Telecommunications Union
LAN	Local Area Network
MCU	Multipoint Control Unit
MIB	Management Information Base
OSI	Open System Interconnection
P2P	Peer to Peer
PC/PDA	Personal Computer/ Personal Digital Assistant
PCM	Pulse Code Modulation
PHB	Per Hop Behavior
PRIQ	Priority Queuing
PSTN	Public Swicthed Telephone Network
QoS	Quality of Service (Qualidade de Serviço)
RAS	Registration, Administration and Status
RED	Random Early Discard
RSVP	Resource reSerVation Protocol
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
RTT	Round Trip Time
SCN	Switched Circuit Networks
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SMIB	Security Management Information Base
SS7	Sistema de Sinalização 7
SSH	Secure Shell
TCA	Traffic Condition Agreement

TCP	Transmition Control Protocol
TMN	Telecommunication Management Network
TOS	Type of Service
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VoIP	Voice over IP (voz sobre IP)
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing

1. INTRODUÇÃO

Desde o surgimento do *Internet Protocol*, em maio de 1974, até os dias de hoje, o mundo tem se voltado cada vez mais à Internet. Tarefas corriqueiras como pagar contas em banco,ou até mesmo assistir televisão, já encontram soluções no mundo IP e não foi diferente com o telefone. O VoIP (Voice over IP), como é conhecida a tecnologia de comunicação de voz sobre redes IP, vem angariando uma parcela de mercado expressiva desde de 2004, quando os investimentos em desenvolvimento tornaram-se significativos em todo mundo, sendo hoje no Brasil mais de 9,1 milhões de usuários do serviço, com cerca de 1 milhão concentrados em empresas, representando 2% das linhas tradicionais[1].

Essa crescente demanda pelo VoIP pode ser explicada pelas várias facilidades oferecidas, tais como baixo custo e, em alguns casos, pela implementação simplificada comparada à redes tradicionais. Serviços como chamadas de vídeo, compartilhamento de arquivos e mensagens instantâneas são suportados, além do fato de que, por ser incorporada à Internet, a chamada é independente da localização geodésica e horário de utilização, parâmetros estes que influenciam diretamente nos custos das chamadas na telefonia fixa e móvel.

Entretanto, como nos primórdios da Internet ninguém imaginou que a rede mundial de computadores se tornaria o que é hoje, ela foi inicialmente projetada utilizando a regra de melhor esforço (Best-Effort), ou seja, não há nenhum mecanismo de qualidade de serviço que garanta a alocação de recursos da rede. Além disso, o protocolo da camada de transporte utilizado pelo VoIP é o UDP (User Datagram Protocol), que aumenta a eficiência (essencial em aplicações em tempo real), mas não é orientado à conexão e não oferece mecanismos para assegurar que os pacotes sejam entregues, ou mesmo que cheguem ordenados.

Com a crescente demanda pela telefonia IP e frente a esses desafios inerentes da Internet e do UDP, os fabricantes de equipamentos de redes começaram a estudar e desenvolver protocolos que garantissem qualidade de serviço fim-a-fim. Assim, surgiram dois modelos de classes de serviço para tráfego Internet: O Differentiated Services (DiffServ), onde há o tratamento diferenciado, dando preferência estatística à determinados fluxos de dados; e o Integrated Services (IntServ), que oferece uma garantia absoluta na alocação de recursos da rede[2].

Dentre muitos fatores que afetam a qualidade de voz do serviço VoIP, alguns parâmetros podem ser destacados como mais importantes para uma análise. No gerenciamento de QoS (Quality of Service) de tráfego VoIP são eles que são usados: CODEC (Codificador/Decodificador), perda de quadros, atraso, variação de atraso (jitter). O CODEC é o código de digitalização da voz e quanto maior a sua taxa de dados, maior a qualidade da voz. A perda de quadros ocorre quando existem erros nas montagens destes ou quando trafegam em redes com congestionamento. Como o VoIP usa UDP, que não admite retransmissões (pois adicionam atraso ao tráfego que tem caráter de tempo real), perda de quadros significa a perda de sons, o que pode tornar a comunicação ruim com sílabas ou palavras cortadas. O atraso de pacotes causa o efeito de eco na comunicação, degradando a qualidade da voz. A variação de atraso (jitter) provoca embaralhamento e gaps no stream de dados[3].

Devido a vulnerabilidade da qualidade de voz em relação a esses parâmetros listados, é exigido um dimensionamento e um gerenciamento de QoS eficiente da rede em que os pacotes estão trafegando. A idéia, então, é o desenvolvimento de uma solução adaptativa para garantir a qualidade de serviço de aplicações VoIP por meio de um monitoramento contínuo deste tráfego, proporcionando à ele um tratamento adequado com a arquitetura DiffServ.

1.1 MOTIVAÇÃO

Apesar do serviço VoIP se encontrar em grande crescimento no mercado, a qualidade da voz e satisfação dos clientes em alguns casos é inferior quando comparado ao serviço de telefonia comum. Com este cenário, tecnologias que garantam a qualidade de serviço para o VoIP estão sendo

fortemente procuradas. A motivação para este projeto é exatamente esta carência de soluções com a abordagem adaptativa, que se mostra interessante devido ao caráter dinâmico das redes IP.

1.2 OBJETIVO

O projeto tem como objetivo desenvolver uma ferramenta capaz de prover qualidade de serviço ao tráfego VoIP de forma a se adaptar ao estado da rede e tomar decisões que maximizem a eficiência do uso de recursos dessa rede.

1.3 ESTRUTURA DO TRABALHO

O trabalho foi dividido em oito capítulos: Introdução, VoIP, QoS, Gerência de Redes, Proposta de Implementação, Cenário de Testes, Testes e Resultados, e por fim, Conclusão.

O primeiro capítulo é a Introdução, que apresenta uma breve introdução as idéias envolvidas no projeto, a motivação, objetivo geral e estrutura do trabalho. O segundo capítulo aborda a tecnologia VoIP (Voice over IP) em suas características e bases, apresentando os conceitos e protocolos existentes assim como suas vulnerabilidades. O terceiro capítulo explicará as necessidades de se garantir a qualidade de serviço nas redes e o protocolo DiffServ, escolhido para o nosso sistema. O quarto capítulo apresenta a idéia, os conceitos e vantagens do gerenciamento de redes. O quinto capítulo descreve a proposta de implementação, com os seus princípios de funcionamento. O sexto capítulo exibe a configuração do cenário de testes, descrevendo as ferramentas utilizadas. O sétimo apresenta os resultados e análises obtidos nos testes realizados. O oitavo e último capítulo apresenta as conclusões sobre a proposta apresentada.

2. VOZ SOBRE IP

O princípio de funcionamento do VoIP é transmitir voz, empacotados em tempo real, através de uma rede IP, seja ela Internet, intranet ou até mesmo uma LAN (*Local Area Networks*). Nesse processo a voz é digitalizada, comprimida e convertida em pacotes IP que são transmitidos através da rede IP. Protocolos de Sinalização são usados para estabelecer e finalizar as chamadas, carregando informações importantes para a localização dos usuários e negociação dos recursos.

Uma das principais razões de se combinar voz e redes de dados é a redução dos custos. Ligações entre telefones IP são realizadas sem custos, a não ser o custo da própria infra-estrutura, e ligações entre um telefone IP e um telefone ligado à PSTN (*Public-Switched Telephone Network*) apresentam custos reduzidos se comparado às tarifas cobradas na telefonia convencional, principalmente em ligações de longa distância. Outro ponto que contribui a favor do VoIP é a vertente da convergência digital, que une esforços para se integrar dados, voz e vídeo, conceito denominado por alguns como *Triple Play*.

2.1 DIGITALIZAÇÃO DA VOZ

O processo de criação de pacotes IP através da voz é dado da seguinte forma[4]:

O sinal de voz analógico é convertido em um sinal PCM (Pulse Code Modulation), que é uma versão digitalizada do sinal, obtido através da amostragem do sinal de voz original em intervalos regulares de tempo.

O eco de linha é removido do sinal PCM. O sinal resultante ainda é analisado para se remover intervalos de silêncio e para se fazer a detecção de tom.

As amostras do sinal PCM são convertidos em frames de voz, e um vocoder (instrumento que sintetiza a voz) comprime os frames, como mostrado na figura 2.1.

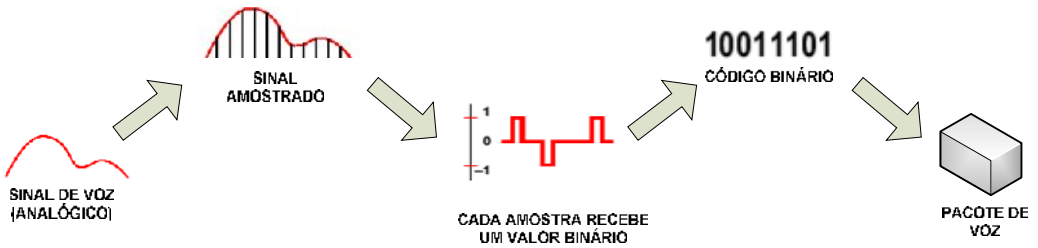

Figura 2.1 - Digitalização do sinal de voz analógico

Os frames de voz são integrados em um pacote de voz. Primeiramente, um pacote RTP (Real Time Protocol) com um cabeçalho de 12 bytes é criado, como mostrado na figura 2.2. Depois um pacote UDP de 8 bytes com os endereços de origem e destino são adicionados, mostrado na figura 2.3. Finalmente, o cabeçalho IP de 20 bytes contendo o gateway de origem e destino é incorporado, detalhado na figura 2.4.

bit offset	0-1	2	3	4-7	8	9-15	16-31
0	Ver.	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	CSRC identifiers (optional) ...						

Figura 2.2 –Cabeçalho RTP

bits	0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Figura 2.3 – Cabeçalho UDP

bit offset	0–3	4–7	8–15	16–18	19–31
0	Version	Header length	Differentiated Services	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
160 or 192+	Data				

Figura 2.4 – Cabeçalho Ipv4

O pacote é enviado através de uma rede IP onde roteadores e switches examinam o endereço de destino, e o roteador entrega o pacote apropriadamente ao destino, exemplificado na figura 2.5.

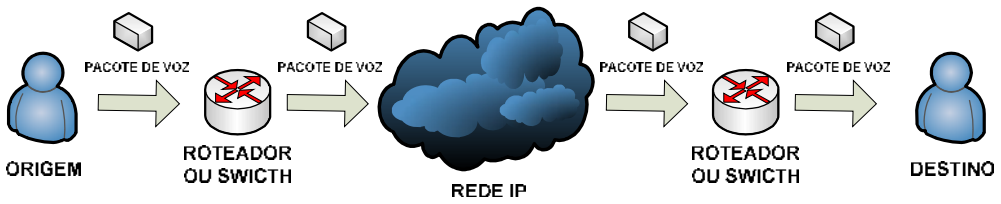

Figura 2.5 – Transmissão de pacotes de voz através rede IP

Quando o nó de destino recebe o pacote, este passa pelo processo reverso para se tornar inteligível. Os pacotes IP são numerados assim que são criados e enviados ao seu destino. O receptor deve reorganizar os pacotes caso eles cheguem fora de ordem para que a voz seja "criada". Os endereços IP e os números de telefone devem ser mapeados de forma correta.

2.2 CONTROLE DE ERROS

Durante a transmissão, pacotes podem ser perdidos ou atrasados, além da possibilidade dos pacotes serem danificados por erros aleatórios. Como estamos falando em uma aplicação em tempo real não podemos simplesmente retransmitir os pacotes defeituosos, então sofisticados sistemas de detecção e correção de erros são utilizados para tentar preencher a lacuna deixada por esses pacotes. Algorítmos tentam "adivinhar" o conteúdo dos pacotes defeituosos ou perdidos e criam informação sonora para substituir essas lacunas, na tentativa de se aproximar o máximo da informação original e tornar esses problemas imperceptíveis ao receptor.

2.3 REAL TIME PROTOCOL

Outro ponto a ser destacado são os dois principais tipos de tráfego que rodam sobre o IP: na camada de transporte do modelo OSI (Open System Interconnection) temos tráfego UDP e na camada de Sessão encontramos tráfego RTP (Real Time Protocol) .A IETF (Internet Engineering Task Force), comunidade internacional que desenvolve e promove padrões para a Internet, adotou o RTP para aplicações em tempo real e sensíveis à *delay*, como é o caso do VoIP. Sendo assim, o pacote VoIP é transportado com um cabeçalho RTP/UDP/IP. Dois bits de informação importantes em aplicações de informação contínua (vídeo e audio) constantes no cabeçalho RTP (Figura 2) são o sequence number (número de sequência) e o timestamping (tempo de ocorrência do evento). O primeiro é utilizado para se determinar a ordem de chegada dos pacotes, enquanto o segundo é usado para se determinar o jitter.

2.4 REAL TIME CONTROL PROTOCOL

O RTP é geralmente usado em conjunto com o RTCP (Real Time Control Protocol), protocolo usado para monitorar estatísticas da transmissão e informações de qualidade de serviço durante uma sessão. Essas informações podem ser usadas pela fonte de informação para se mudar adaptativamente o codec utilizado, melhorando aspectos da conexão, e determinar falhas na transmissão.

2.5 PROTOCOLOS DE SINALIZAÇÃO

O primeiro passo a se tomar para se fazer uma ligação VoIP é estabelecer a conexão entre as partes envolvidas. Entretanto, sabemos que por natureza o IP não é voltado à conexão, ou seja, os pacotes de dados podem chegar desordenados e podem tomar diferentes rotas, de acordo com as decisões tomadas pelos roteadores e switches, o que é inaceitável para o nosso tipo de aplicação.

Semelhantemente ao handshake usado pelo DHCP (Dynamic Host Configuration Protocol), os protocolos de sinalização usados pelo VoIP usam o TCP para estabelecer e terminar uma ligação[4]. A função principal dos protocolos de sinalização são iniciar uma sessão e então achar um caminho em comum entre as partes envolvidas, e por fim terminar a sessão ao final da ligação.

Quando falamos de uma conexão em uma LAN (Local Area Network) a conexão entre os nós é facilmente gerenciada pelos protocolos de sinalização, o problema é quando a conexão passa por uma WAN (Wide Area Network) ou por uma rede de circuitos comutados, casos onde esses protocolos têm outras responsabilidades, como tradução de endereços, gerenciamento da banda, autorização e em alguns casos tomada de decisão de rotas.

Entre os protocolos de sinalização mais usados em aplicação VoIP, podemos destacar dois: o H.323, recomendação da ITU (International Telecommunications Union), e o SIP (Session Initiation Protocol), padrão escolhido pela IETF, ambos padrões destinados à aplicações VoIP e comunicações multimídia.

2.5.1 H.323

Como o H.323 é um protocolo da camada de Sessão do modelo OSI, sua principal função é a de realizar o controle e gerenciamento da ligação. O H.323 conta ainda com outros dois protocolos de sinalização, o H.225 e o H.245.

Quando uma sessão é iniciada entre dois dispositivos H.323, o padrão H.225 usa o protocolo Q931 ISDN (Integrated Services Digital Network) para estabelecer e terminar determinadas funções usando o TCP, objetivando uma conexão confiável. O H.245 então abre outra conexão TCP para determinar as capacidades dos dispositivos, negociar os codecs a serem utilizados e escolher as portas que serão usadas na sessão. Assim, um canal é aberto onde o tráfego VoIP irá viajar usando o UDP na camada de transporte, devido à velocidade, e o RTP na camada superior, para ordenar e sincronizar os pacotes.

É importante notar que para estabelecer, negociar e gerenciar a sessão o protocolo usado é o TCP, devido a sua confiabilidade. Já para o transporte o UDP é o protocolo escolhido, devido sua velocidade de transmissão e cabeçalho menor. O tamanho do pacote ainda pode ser reduzido se for utilizado o compressor de cabeçalho RTP, diminuindo o cabeçalho combinado IP/UDP/RTP para 65% a 10% do tamanho do pacote total[5].

Uma rede H.323 é composta de quatro componentes lógicos, explanados a seguir[5].

2.5.1.1 TERMINAL

O Terminal é o equipamento na extremidade da conexão, a interface com o usuário, podendo ser um telefone IP ou até mesmo um computador rodando uma aplicação H.323. Um terminal H.323 deve suportar os seguintes protocolos: o H.245, que negocia o canal e os recursos da conexão; o RAS (Registration, Administration and Status); o Q931 para sinalização e estabelecimento da conexão; e por fim, suporte ao RTP e ao RTCP, responsáveis pelo transporte do tráfego VoIP.

2.5.1.2 MULTIPOINT CONTROL UNIT

O MCU dá suporte à três ou mais terminais participarem de conferências. Ele é constituido do MC (Multipoint Controller), responsável pelas funções do H.245, e pelo MP (Multipoint Processor), que é responsável por controlar os banda e os diferentes fluxos dos terminais.

2.5.1.3 GATEWAYS

O gateway é responsável por vários serviços, sendo que ele dá suporte à conexões entre redes H.323 e redes que não usam o H.323, como o SCN (Switched Circuit Networks), por exemplo. Além disso, estabelece e termina as conexões, traduz o audio, vídeo e dados, e ainda executa o RAS para ser registrado nos gatekeepers. O gateway usa os protocolos H.225 e H.245 nas redes H.323 e os protocolos ISDN e SS7 (Sistema de Sinalização 7) nas redes de circuito comutado.

2.5.1.4 GATEKEEPER

É o componente mais importante em uma arquitetura H.323, sendo responsável por gerenciar todos os os terminais, gateways e MCUs registrados em uma região da rede H.323. Nele é feita a configuração de endereçamento, autorização e autenticação, além do gerenciamento da banda utilizada e da cobrança de tarifas.

2.5.2 SIP

Diferentemente do H.323, o protocolo SIP começou como um protocolo P2P (Peer to Peer), também usado no controle e processamento de chamadas entre dispositivos VoIP. Sendo assim, percebemos que o SIP teve origem como um protocolo mais simples e leve que o concorrente H.323 que utiliza em sua arquitetura servidores e gateways centralizados. Entretanto, com o investimento em desenvolvimento do protocolo SIP, houve uma certa concorrência entre esses dois protocolos na tentaiva de se dominar o mercado de telefonia IP.

Em 2007, o SIP se tornou o protocolo padrão de sinalização para VoIP e comunicações multimídia[6]. A escolha foi influenciada pelo fato do SIP ser um padrão IETF, sendo mais facilmente adotado pela indústria se comparado com o ITU. Vale ressaltar que os dois padrões são apoiados pelas duas comissões.

Por ser um protocolo baseado em texto, similar ao HTTP, o SIP se integra bem com aplicações Internet como e-mail, mensagem instantânea e conferências de vídeo e áudio. Esse protocolo apresenta cinco elementos lógicos, que serão abordados em tópicos a seguir: o user agent, back-to-back user agente, proxy server, redirect server e o registrar[6].

2.5.2.1 USER AGENT

São os dispositivos na ponta da conexão que iniciam e terminam a sessão através de solicitação e a resposta dos pedidos. Em uma rede IP fechada, os dispositivos de borda tem a capacidade de achar outro dispositivo sem a necessidade de outras entidades.

2.5.2.2 B2BUA

Trata-se da aplicação que intermedia os dispositivos de borda, sendo visto como um elemento de borda pelos demais elementos da rede. Ele mantêm o estado da ligação e é responsável por terminar a

conexão. Pode ser visto como um gateway, representando o caminho da borda de uma rede IP a outra borda de uma rede PSTN.

2.5.2.3 PROXY SERVER

Serve também como intermediário, processando pedidos passando-os a outro servidor SIP, deixando o controle para os dispositivos de borda. O proxy pode traduzir o pedido e reescrever a mensagem antes de transmiti-la. Executa a tradução dos endereços juntamente com o domínio resolvendo o URL (Uniform Resource Locator) dos e-mails ou o número do telefone para endereços IP e usa o DNS para encontrar os servidores SIP fora do domínio. O SIP usa o padrão ENUM para mapear e indexar os números de telefone estrangeiros e associá-los a um endereços IP.

2.5.2.4 REDIRECT SERVER

Mapeia o pedido do cliente para o URL mais próximo da parte sendo chamada e então envia devolta a chamada à parte requerente. O redirect server não passa pedidos a outro servidor. Por exemplo, caso alguem faça uma ligação para um telefone IP em uma estação de trabalho e a pessoa esteja ausente, a ligação deve ser redirecionada ao celular da mesma.

2.5.2.5 REGISTRAR SERVER

O registrar tem a função de registrar os usuários em um banco de dados assim que eles estejem online. Informações indicando o ID e o dispositivo IP que estão usando são armazenados pelo endereço IP, número do telefone ou URL. Tem a função também de atualizar a localização dos usuários, o status ou até mesmo de registrar qual dispositivo IP que o usuário pretende receber as ligações.

3. QUALIDADE DE SERVIÇO

Qualidade de serviço (QoS) pode ser definido como o conjunto de requisitos de serviço que a rede deve atender para assegurar um nível de serviço adequado à transmissão de dados. Seu gerenciamento tem como objetivo fazer com que programas em tempo real façam uso eficiente da largura de banda da rede em que seus pacotes trafegam e oferecer um sistema de entrega com garantias para o tráfego.[7]

Com o aumento de usuários e surgimento contínuo de novas aplicações na Internet, a demanda por QoS é hoje bastante grande. Usuários querem altos níveis de QoS para utilizar aplicações de vídeo-conferência e VoIP. Já os provedores desejam mecanismos de diferenciação de serviços para atender seus usuários. Aplicações multimídia em tempo real ou ao vivo, streaming, clientes que pagam por um serviço superior, entre outros fatores, mostram a necessidade de se tratar diferentes tráfegos na Internet de forma diferenciada, proporcionando qualidade de serviço.

Uma vertente de pensamento acredita que, com as novas tecnologias da camada física e enlace como fibras óticas e WDM, largura de banda será algo abundante e não haverá problemas para as aplicações obterem qualidade de serviço. Porém, outra linha de pensamento diz que apesar de haver uma largura de banda maior no futuro, novas aplicações sempre surgirão para ocupá-la, não descartando a necessidade de se criar mecanismos para prover QoS. Mesmo se a primeira opinião se tornar realidade na Internet, ainda levará muito tempo para se concretizar, e hoje existe a necessidade desses mecanismos para QoS.

3.1 ABORDAGENS DE QOS

Existem três abordagens principais para a evolução da arquitetura da camada de rede (IP). A primeira seria permanecer com o modelo de melhor esforço usado atualmente. A segunda é a arquitetura de serviços integrados (Integrated Services) e a terceira é a arquitetura de serviços diferenciados (Differentiated Services), como mostrado na figura 3.1.

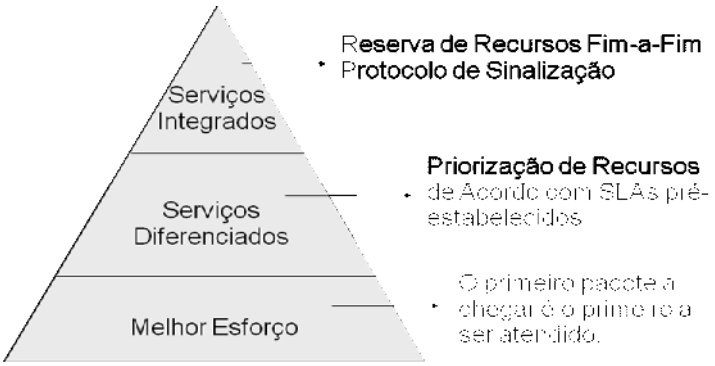

Figura 3.1 – Níveis de QoS

3.1.1 MELHOR ESFORÇO

A Internet atualmente utiliza o modelo de melhor esforço, que consiste em todos os usuários e aplicações utilizando a mesma banda e seus tráfegos concorrendo entre si. Os pacotes são encaminhados da melhor forma possível conforme as rotas disponíveis. Quando há congestionamento, são formadas filas de pacotes nos roteadores que os enviarão assim que for possível. Se uma fila já está no limite de sua capacidade, o próximo pacote que chegar neste roteador será descartado sem nenhuma distinção. É um modelo simples e eficiente para entregar pacotes, mas não garante que todos

serão entregues. Serviços de multimídia, como é o caso do VoIP, necessitam que seus pacotes possuam algumas garantias em suas entregas o que o modelo de melhor esforço não pode oferecer.

3.1.2 SERVIÇOS INTEGRADOS (INTSERV)

A arquitetura de serviços integrados propõe uma extensão da arquitetura original do melhor esforço. São adicionados novos componentes e mecanismos suplementares, mas sem substituir o serviço IP básico. O modelo supõe que os recursos da rede devem ser gerenciados e estes novos mecanismos são usados para fazer reservas de recursos na Internet com o intuito de atender os requerimentos de aplicações de tempo real. Antes de fazer a transmissão de pacotes, a aplicação faz a reserva de recursos ao longo do caminho até o destino. O transmissor descreve as características de seu fluxo e seus requerimentos à rede, que aceitará ou não fazer a reserva dos recursos de acordo com a disponibilidade. Estabelecida a reserva, a aplicação envia seus pacotes pelo caminho reservado, e a rede garante o cumprimento do que foi requerido. A rede faz as reservas de recursos de forma dinâmica através do protocolo de sinalização RSVP (Resource reSerVation Protocol). Esse protocolo faz com que o modelo de serviços integrados tenha problemas de escalabilidade. Em uma grande rede, o número de mensagens RSVP para a reserva de recurso se torna grande, além de se ter alta dificuldade em conseguir reservar recursos por longos caminhos. Assim, o IntServ se mostra pouco viável.

3.1.3 SERVIÇOS DIFERENCIADOS (DIFFSERV)

Este modelo se encontra entre os dois extremos representados pelos modelos anteriores. Ele não trata todos os fluxos da mesma maneira como o melhor esforço, mas também não trata individualmente cada fluxo como em Serviços Integrados. Com Serviços Diferenciados, o tráfego é dividido em um pequeno número de grupos denominado classes de encaminhamento. Cada classe de encaminhamento representa um tratamento de encaminhamento predefinido em termos de prioridade de descarte e alocação de banda. O policiamento de tráfego é feito nas bordas da rede e um encaminhamento baseado em classes é feito no interior. Roteadores na borda de uma rede com Serviços Diferenciados farão o mapeamento dos pacotes entrantes nas diferentes classes de encaminhamento e os roteadores internos da rede irão encaminhá-los baseados somente na classe de encaminhamento do pacote. Por não usar um protocolo de sinalização e o policiamento do tráfego ser deslocado para as bordas, o modelo Serviços Diferenciados apresenta uma característica importante que é a escalabilidade. Esta arquitetura se mostra viável para proporcionar QoS em aplicações VoIP e será detalhada posteriormente no projeto.

3.2 PARÂMETROS DE QOS

Pode se dizer que a necessidade de QoS na Internet hoje é um fato, mas quais são os seus componentes e como ele são mensurados? Qualidade de serviço na Internet pode ser expressa pela combinação de atraso, jitter, largura de banda e confiabilidade[8].

3.2.1 ATRASO

Atraso é o tempo levado pelo processo de envio de um pacote, desde a sua saída até a sua chegada no destino. Quanto maior o atraso, maior o estresse para o protocolo da camada de transporte operar eficientemente. No caso do TCP, se o atraso é grande entre o emissor e o receptor do pacote, o ACK será lento e o TCP se torna insensível as dinâmicas mudanças de curto prazo da rede. Para aplicações interativas de voz e vídeo, a introdução de atraso causa ao sistema problemas que comprometem seu funcionamento.

3.2.2 VARIAÇÃO DE ATRASO (JITTER)

Jitter é a variação do atraso na comunicação fim a fim. Um Jitter elevado afeta o TCP o fazendo adotar uma estimativa do RTT (Round Trip Time) conservadora e configurando um timeout alto para o restabelecimento do fluxo de dados. Em aplicações de tempo real como VoIP, que são baseadas no UDP, um alto nível de jitter é inaceitável, pois deixa o sinal distorcido. Nesses casos, a única forma de corrigir o sinal é aumentar a Reassembly Playback Queue do receptor, o que adiciona atraso no sinal, dificultando a manutenção de sessões interativas.

3.2.3 LARGURA DE BANDA

Largura de banda é a máxima taxa de transferência de dados suportada entre os dois pontos comunicantes. É importante notar que esta taxa é limitada tanto pela infra-estrutura do caminho percorrido pelo tráfego na rede, como pelo número de fluxos que estão compartilhando componentes comuns deste caminho fim a fim.

3.2.4 CONFIABILIDADE

Confiabilidade como uma propriedade de sistemas de transmissão, pode ser considerado como a taxa média de erros no meio. Pode ser também produto do sistema de comutação, onde pode ocorrer entrega de pacotes fora da ordem em que foram envidados ou até perda e corrupção destes. No caso de aplicações de vídeo e voz baseada em UDP, falta de confiabilidade causa distorção induzida no sinal analógico original no receptor.

3.3 ARQUITETURA SERVIÇOS DIFERENCIADOS (DIFFSERV)

Com a dificuldade de implementação e desenvolvimento de Serviços Integrados e RSVP, a arquitetura DiffServ aparece como solução mais viável para provimento de QoS, principalmente por ter alta escalabilidade.

O DiffServ é uma arquitetura mais simples que a IntServ, não utiliza protocolo de sinalização e baseia seu funcionamento em um esquema de priorização de recursos por SLAs (Service Level Agreements) previamente definidos. Possui escalabilidade por fazer agregação de fluxos e separação das funções entre roteadores de borda e de núcleo.

3.3.1 PRINCÍPIOS DE FUNCIONAMENTO

A Arquitetura DiffServ apresenta os seguintes princípios básicos[9]:

- **Alocação de recursos para tráfegos agregados ao invés de fluxos individuais.**

Os fluxos são divididos em grupos denominados classes de encaminhamento e a performance de um fluxo individual é assegurada por priorização e provisão para a sua classe de encaminhamento, contrastando com a reserva por fluxo proposta pelo IntServ.

- **Policiamento de tráfego deslocado para as bordas e encaminhamento baseado em classes no centro.**

Somente os roteadores de borda da rede irão fazer classificação de tráfego e marcação de pacotes. Feita a marcação, os roteadores internos irão observar qual classe de encaminhamento cada pacote pertence e determinarão o tratamento que cada um deve receber.

- **Definição de comportamentos de encaminhamento e não de serviços.**

Cada classe de encaminhamento representa um tratamento e não um serviço. Os serviços podem ser construídos combinando classes de encaminhamento e controle de admissão.

Recursos são assegurados através de provisionamento e priorização ao invés de fazer reserva por fluxos individuais como o IntServ. Ao alocar recursos para classes de encaminhamento e controlando a quantidade de tráfego para essas classes, DiffServ cria níveis de serviços diferentes, mas não garante a fluxos individuais, garantias específicas de banda ou atraso.

- **Ênfase em SLAs ao invés de sinalização dinâmica.**

DiffServ tem como objetivo assegurar que SLAs entre clientes e provedores de serviços sejam honradas. Dessa forma não há necessidade de se autenticar e tarifar cada fluxo que necessita da reserva como acontece com o IntServ.

- **Foco em um domínio único vs. fim-a-fim.**

Classes de serviço são definidas em um domínio único e entre domínios diferentes que o provedor de serviço se extende. Pode haver também um mapeamento de definições entre domínios acordados previamente. A visão fim a fim existe se os dois pontos estão no domínio DiffServ.

3.3.2 TRATAMENTO DE ENCAMINHAMENTO

Serviços e tratamentos de encaminhamento são conceitos diferentes, apesar de apresentarem similaridades. Serviço é definido como uma performance geral observada pelo cliente enquanto tratamento de encaminhamento é exatamente o mecanismo implementado no roteador. Por exemplo, para um serviço sem nenhuma perda de pacote se adota um tratamento de encaminhamento de maior prioridade de transmissão. Existem diversas formas de se implementar serviços com diferentes tratamentos de encaminhamento.

Em Serviços Diferenciados, o tratamento de encaminhamento em um nó é definido por *per-hop behavior* (PHB) representados por valores de 6 bits chamados de *Differential Services Codepoint* (DSCP). Pacotes com o mesmo DSCP recebem o mesmo tratamento de encaminhamento. Um grupo de PHBs pode ser formado quando vários PHBs partilham de uma coação comum. Um único PHB pode ser visto como um caso especial de um grupo de PHBs. São tipicamente implementados por meio de gerenciamento de *buffer* e agendamento de pacotes.

Os serviços a serem implementados com os tratamentos de encaminhamento, são definidos por SLAs (Service Level Agreements) entre usuário e provedor de serviço. Um termo importante da SLA é o *Traffic Condition Agreement* (TCA). Ele define parametros para perfis de tráfego e ações de policiamento como parâmetro de Token Bucket para cada classe, *throughput*, *delay*, prioridade de descarte, marcação adicional, serviços de modelagem entre outros. Além do TCA, o SLA contém acordos de disponibilidade, segurança, tarifação, precificação, monitoração.

3.3.3 CAMPO DIFFSERV

A abordagem DiffServ usa 6 bits do cabeçalho IP para fazer a marcação e classificação do tráfegos que entram em um domínio DS. Sua base está na redefinição do campo TOS do cabeçalho IP original.

O campo TOS possui 8 bits. 3 bits são para precedência e representa prioridade de tráfego, 3 bits são para tipo de serviço e indica preferências de *throughput,* atraso e perda. 2 bits não possuem função e não são usados.

O DiffServ redefine o TOS, usando 6 bits para DSCP (*Differential Services Codepoint*) que codifica os PHBs para cada pacote. Os outros 2 bits não são utilizados. O DSCP funciona como um índice e o mapeamento de DSCPs deve ser configurável.

O campo *DiffServ* suporta 64 valores, ou seja, 64 PHBs diferentes. Foi definida uma divisão de três pools de tarefas e gerenciamentos de DSCPs. Um pool de 32 códigos recomendados e a serem

padronizados, um pool de 16 códigos para uso experimental ou local e um outro pool de 16 códigos para uso experimental e local, mas que está sujeito a padronização caso o primeiro pool seja completamente usado.

Atualmente, estão sendo padronizados DSCPs para dois grupos de PHBs: Encaminhamento Expresso (EF) e Encaminhamento Assegurado (AF). Também existe uma padronização de códigos chamada de Seletor de Classe (CS) que proporciona uma retro-compatibilidade limitada com o *best effort*. Esses grupos de PHBs serão detalhados posteriormente.

A tabela 3.1 mostra a distribuição de DSCPs atualmente padronizados[9]:

DSCP	PHB
000 000	CS0(DE)
000 001	EXP/LU
000 010	-
000 011	EXP/LU
000 100	-
000 101	EXP/LU
000 110	-
000 111	EXP/LU
001 000	CS1
001 001	EXP/LU
001 010	AF11
001 011	EXP/LU
001 100	AF12
001 101	EXP/LU
001 110	AF13
001 111	EXP/LU

DSCP	PHB
010 000	CS2
010 001	EXP/LU
010 010	AF21
010 011	EXP/LU
010 100	AF22
010 101	EXP/LU
010 110	AF23
010 111	EXP/LU
011 000	CS3
011 001	EXP/LU
011 010	AF31
011 011	EXP/LU
011 100	AF32
011 101	EXP/LU
011 110	AF33
011 111	EXP/LU

DSCP	PHB
100 000	CS4
100 001	EXP/LU
100 010	AF41
100 011	EXP/LU
100 100	AF42
100 101	EXP/LU
100 110	AF43
100 111	EXP/LU
101 000	CS5
101 001	EXP/LU
101 010	-
101 011	EXP/LU
101 100	-
101 101	EXP/LU
101 110	EF
101 111	EXP/LU

DSCP	PHB
110 000	CS6
110 001	EXP/LU
110 010	-
110 011	EXP/LU
110 100	-
110 101	EXP/LU
110 110	-
110 111	EXP/LU
111 000	CS7
111 001	EXP/LU
111 010	-
111 011	EXP/LU
111 100	-
111 101	EXP/LU
111 110	-
111 111	EXP/LU

Tabela 3.1 – Alocação de DSCPs Atual

3.3.4 CLASSIFICAÇÃO E CONDICIONAMENTO DE TRÁFEGO

Em Serviços Diferenciados, os roteadores de borda e internos possuem tarefas distintas. Na borda acontece a classificação e condicionamento de tráfego. São nestes roteadores que os pacotes são mapeados nas classes de encaminhamento suportadas e é assegurado que o tráfego esteja nos conformes com a SLA do usuário.

Os nós da borda traduzem o TCA (Traffic Condition Agreement) em perfis de tráfego para cada usuário que está diretamente conectado. O perfil de tráfego especifica propriedades temporais do stream de tráfego de um usuário em termos de parâmetros de condicionamento de tráfego e provê uma série de regras para determinar se um pacote é considerado dentro do perfil especificado ou não.

O módulo de classificação contém o classificador (classifier) e o marcador (marker). O módulo de condicionamento contém medidor (meter), marcador (marker), modelador (shaper) e descartador (dropper), como mostrado na figura 3.2.

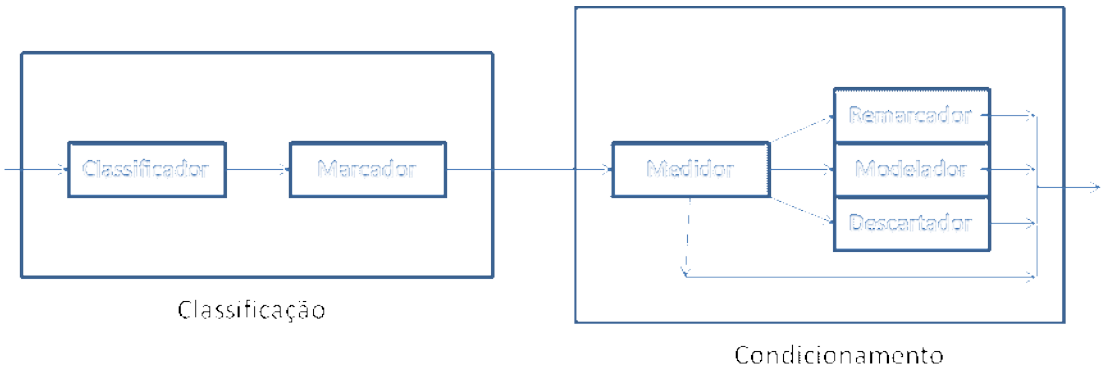

Figura 3.2 – Blocos de Classificação e Condicionamento

- Classificador (*Classifier*): Divide o fluxo de pacotes em grupos baseados no valor do DSCP.
- Marcador (*Marker*): Preenche o campo DS com um DSCP particular e marca o pacote em uma classe de encaminhamento
- Medidor (*Meter*): Mede o fluxo de trafego dos pacotes e compara com o perfil de trafego especificado
- *Remarcador* (*Remarker*): Remarca pacotes que estão em não conformidade com o perfil de tráfego marcado
- Modelador (*Shaper*): Atrasa o pacote em não conformidade com o perfil de tráfego marcado, até o fluxo entrar em conformidade.
- Descartador (*Dropper*): Descarta o pacote em não conformidade com o perfil de tráfego marcado.

Veremos em tópicos posteriores que não haverá a medição nem remarcação dos pacotes no módulo de condicionamento do roteador de núcleo, pois a arquitetura proposta é um pouco diferente da apresentada acima: a classificação dos pacotes não será feita pela aplicação, e sim pela própria estrutura de rede, não havendo por isso necessidade de uma medição e remarcação.

3.3.5 GRUPOS PHB

O DiffServ implementa basicamente dois tipos de serviços: o Expedited Forwarding (EF) e o Assured Forwarding (AF). Para estes seviços, o IETF (Internet Engineering Task Force) tem padronizado o grupo AF PHB e o grupo EF PHB.

3.3.5.1 GRUPO AF PHB

O *Assured Forwarding*, em essência, indica uma prioridade de descarte. Basicamente, ele assegura que pacotes dentro do perfil recebam a capacidade esperada da rede, e os pacotes fora do pefil serão permitidos apenas quando houver excesso de banda disponível.

Classes AF são especificadas em termos de largura banda. Dentro das prioridades de cada classe, os pacotes ainda podem ser classificados em relação a três níveis de importância. No caso de um congestionamento, os pacotes com menor importância dentro de uma determinada classe serão excluídos. A garantia de entrega dos pacotes depende da quantidade de recursos alocados para a classe de cada pacote, a carga da classe e em caso de congestionamento, o nível de importância do pacote.

Existem 4 classes AF e cada uma possui 3 prioridades de descarte. Os 3 primeiros bits do campo DS codificam a classe e os próximos 2 bits codificam a prioridade.

Na tabela 3.2 podemos ver os códigos do AF PHB padronizados[9].

	Classe 1	Classe 2	Classe 3	Classe 4
Baixa prioridade de descarte	001 010	010 010	011 010	100 010
Média prioridade de descarte	001 100	010 100	011 100	100 100
Alta prioridade de descarte	001 110	010 110	011 110	100 110

Tabela 3.2 – DSCPs alocados para o grupo AF PHB

Roteadores de borda devem condicionar o tráfego que entra na rede para uma classe AF especifica baseando-se no requerimento de provisonamento e quantidade de recurso para aquela classe. No interior da rede, roteadores encaminharão os pacotes para as filas provisionadas para a classe de acordo com os 3 primeiros bits do campo DS. Os mecanismos de descarte irão se basear nos próximos 2 bits do campo DS.

As classes AF podem receber taxas maiores que as especificadas desde que recursos adicionais estejam disponíveis.

3.3.5.2 GRUPO EF PHB

EF PHB é definido como um tratamento de encaminhamento para um tráfego agregado onde a taxa de partidas de pacotes deste tráfego saindo de qualquer roteador DS deve ser maior ou igual a uma taxa configurada. O tráfego deve receber esta taxa independentemente dos outros tráfegos existentes no roteador. O DSCP usado para EF PHB é 101110.

Uma simples abordagem para EF PHB é um esquema de fila com prioridade e token bucket. A fila para tráfego EF deve ser o de maior prioridade da rede para assegurar o tratamento do EF e o token bucket limita a quantidade de tráfego EF para que ele não “domine” a rede deixando outros tráfegos sem recursos.

4. GERÊNCIA DE REDES

Vários padrões e recomendações foram criadas no intuito de se gerenciar redes de comunicação, no entanto um deles se sobressaiu, o TMN (Telecommunication Management Network), criado em 1998 pela ITU-T. Esse padrão foi criado com o propósito de gerenciar redes, serviços e equipamentos heterogêneos, funcionando independentemente do fabricante ou tecnologia. Seu funcionamento é baseado na interação com as redes de telecomunicação em vários pontos, através de interfaces padronizadas, podendo utilizar parte da rede de telecomunicações para realizar suas funções[10].

4.1 ESTRUTURA FUNCIONAL

De acordo com a hierarquia GIRS (Gerência Integrada de Redes e Serviços) a estrutura funcional pode ser dividida em cinco níveis, restringindo suas atividades objetivando melhor execução das tarefas, mas podendo haver comunicação entre níveis, adjacentes ou não. A figura 4.1 mostra a divisão e hierarquia desses níveis[11]:

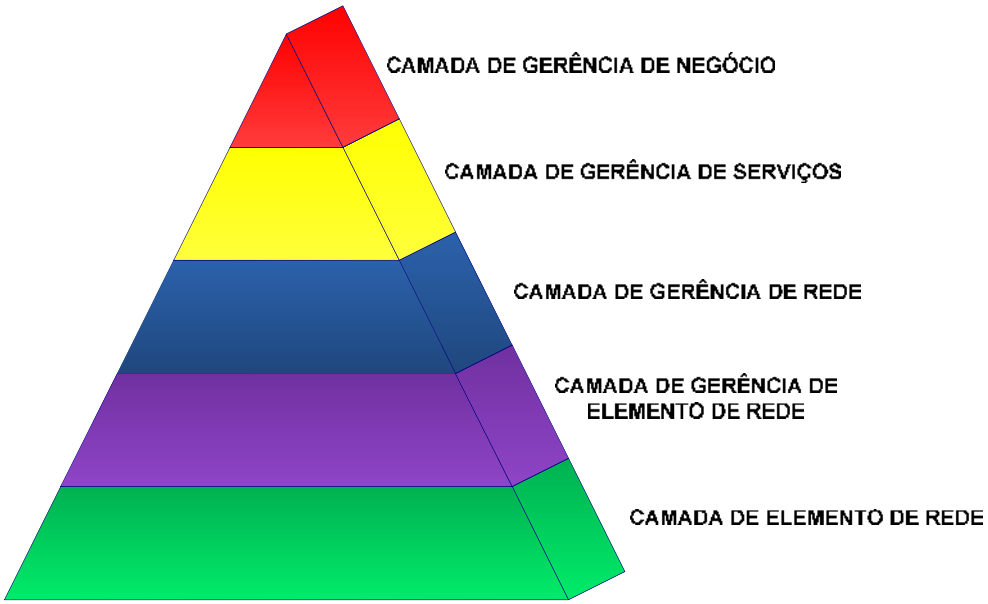

Figura 4.1 – Níveis da Estrutura Funcional GIRS

4.1.1 CAMADA DE GERÊNCIA DE NEGÓCIO

Responsável pelo gerenciamento global do empreendimento, sendo a camada onde ocorrem as ações executivas. Os acordos entre as operadoras e a definção dos objetivos são realizadas neste nível.

4.1.2 CAMADA DE GERÊNCIA DE SERVIÇOS

Executa o provisionamento de serviços, abertura e fechamento de contas, relatórios de falhas e manutenção dos dados de QoS. Questões de tarifação e reclamação dos clientes também são feitas nesta camada.

4.1.3 CAMADA DE GERÊNCIA DE REDE

Gerencia o conjunto de elementos como um todo, tendo uma percepção das subredes envolvidas. Faz a coleta de dados da camada imediatamente inferior, a camada de gerência de elementos de rede, fazendo a análise destas informações objetivando ter uma visão fim-a-fim da rede.

4.1.4 CAMADA DE GERÊNCIA DE ELEMENTOS DE REDE

Esta camada gerencia os equipamentos de redes, obtendo informações relevantes à rede como um todo, repassando essas informações à camada de gerencia de redes. Para tanto, a rede é dividida em subredes afim de facilitar e dinamizar a coleta e repasse dos dados.

4.1.5 CAMADA DE ELEMENTO DE REDE

Corresponde às entidades de redes, podendo ser um hardware ou software, que serão monitorados para a coleta de informações. Nessas entidades, deve existir um agente, que são os responsáveis pela obtenção e repasse dos dados para o sistema de gerência.

4.2 ÁREAS FUNCIONAIS NO GERENCIAMENTO DE REDES

Na tentativa de dividir e organizar as funcionalidades do gerenciamento de redes (planejamento, instalação, operação, manutenção e provisionamento) a ISO dividiu a gerência de redes em 5 áreas funcionais[12], abordadas a seguir:

4.2.1 GERÊNCIA DE DESEMPENHO

Envolve as funções de análise e avaliação de comportamento dos equipamentos da rede. Temos dois tipos básicos de funções nesta área[12]:

Medidas de Tráfego: Possibilitam o controle da entrega de relatórios;

Medidas de Desempenho: Contêm informações relevantes para a análise do desempenho da rede.

Através da obtenção e análise de parâmetros como tempo de resposta, taxa de utilização, disponibilidade, vazão, entre outros, pode-se ter um retrato fiel da rede, essencial para a avaliação e planejamento do sistema. Deve-se deixar claro que a gerência de desempenho tem aspectos proativos, ponto importante quando estamos tratando de redes de comunicação, que objetivam alta disponibilidade. As principais funções da gerência de desempenho estão citadas na figura 4.2.

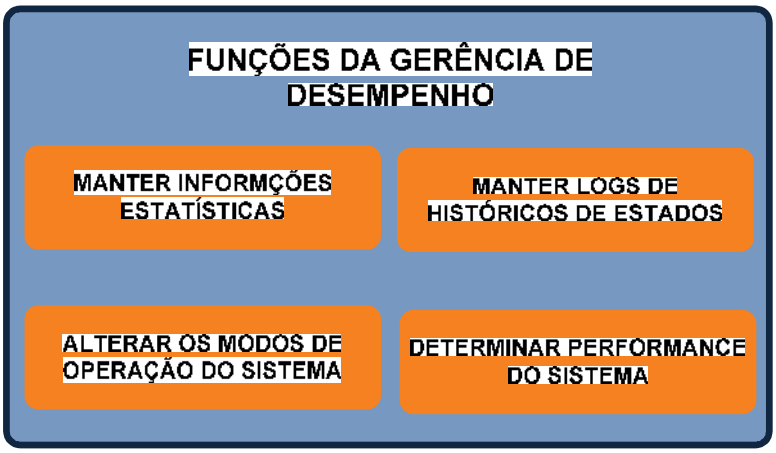

Figura 4.2 – Funções da Gerência de Desempenho

4.2.2 GERÊNCIA DE FALHAS

Tem como objetivo a detecção, isolamento e correção de operações anormais, como degradação e erros na rede. Falhas no sistema podem interromper o funcionamento da rede por um período temporário ou permanente, dependendo da natureza e gravidade da falha. Temos 3 tipos de funções de gerenciamento de falhas[12]:

Supervisão de Alarmes: Gerencia as informações sobre eventos que atentam contra o bom desempenho do sistema;

Teste: Execução de procedimentos, com parâmetros específicos, na tentativa de encontrar a causa de erros;

Relatório de Problemas: Utilzado para encontrar e controlar as ações executadas para liberar alarmes e outros problemas.

Neste tipo de gerência, os erros e avarias sofridas pelo sistema devem ser salvos em um banco de dados em logs e históricos, que serão usados para a determinação e precaução dos erros sofridos, buscando uma atitude proativa. As principais funções da gerência de falhas estão citadas na figura 4.3.

Figura 4.3 – Funções da Gerência de Falhas

4.2.3 GERÊNCIA DE CONFIGURAÇÃO

Dispõem recursos ao usuário para a criação ou modificação de elementos da rede, ou seja, localiza os recursos, registra qual o seu tipo e outras informações importantes. Suas funções podem ser divididas em[12]:

Gerenciamento de Ordem de Serviço: Identifica e gerencia o provisionamento de recursos, podendo ser eles físicos ou lógicos;

Configuração de Recursos: Leque de funções que possibilitam que os recursos da rede sejam criados, roteados, controlados e identificados;

Informações de Recursos: Lista os recursos alocados, além de obter e verificar as informações dos recursos disponíveis.

As principais funções da gerência de configuração estão citadas na figura 4.4.

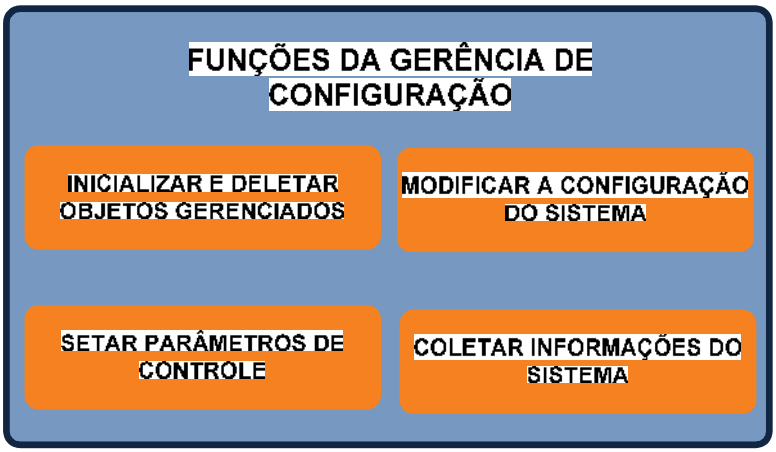

Figura 4.4 – Funções da Gerência de Configuração

4.2.4 GERÊNCIA DE TARIFAÇÃO

Conjunto de ferramentas que estipulam o custo associado ao uso da rede de telecomunicação, ou seja, mede e coleta informações sobre o uso dos recursos e serviços oferecidos e associa o quanto deve ser cobrado. As funções do gerenciamento de tarifação são[12]:

Informar ao cliente o custo dos serviços utilizados;

Oferecer a possibilidade de limitar a tarifação;

Quando for utilizado mais de um recurso de forma conjunta, calcular os custos envolvidos.

A figura 4.5 lista as principais ações da gerência de tarifação.

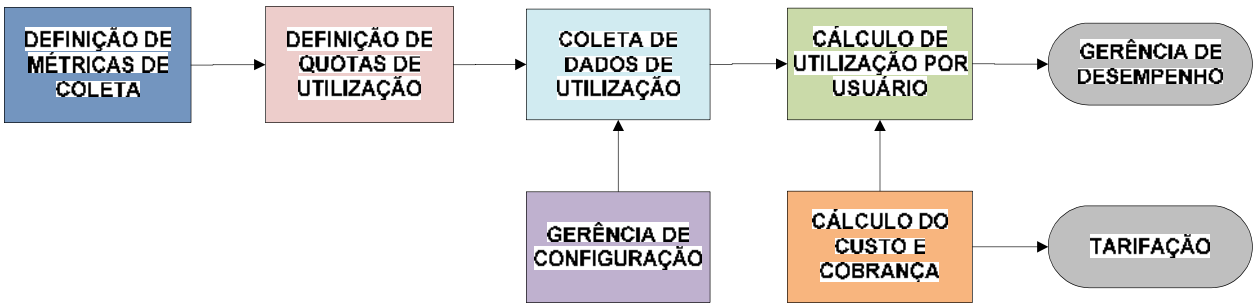

Figura 4.5 – Fluxograma da Gerência de Tarifação

4.2.5 GERÊNCIA DE SEGURANÇA

Responsável pelas políticas de segurança, controlando o acesso à rede por usuários ou ações não autorizadas que ofereçam alguma ameaça ao sistema de telecomunicação.Também responde pelas informações trafegadas, sendo de sua responsabilidade o controle e segurança dos dados.

De acordo com a Arquitetura de Segurança do Modelo OSI, para que um sistema seja seguro 5 questões devem existir[13]:

Autenticação tanto de entidades pares quanto da origem dos dados (authentication);

Controle de acesso aos recursos da rede (access control);

Confidencialidade dos dados (confidenciality);

Integridade dos dados (integrity);

A não-rejeição ou não-repudiação (non-repudiation).

Informações referentesao gerenciamento de segurança são armazenadas em um banco de dados específico, uma MIB (Management Information Base) voltada para a segurança, a SMIB.

5. PROPOSTA DE IMPLEMENTAÇÃO

Como foi observado nas seções anteriores do trabalho, a necessidade de se gerenciar e garantir que o tráfego VoIP tenha prioridade em relação a outros tipos de tráfego é evidente. Pensando nisso, propõe-se um sistema que monitore a rede e, através dos parâmetros coletados no nó receptor, tenha condições de estipular de forma adaptativa diferentes formas de tratamento para os pacotes. A idéia é que a qualidade da transmissão seja aferida por quem ouve a ligação, e não por quem fala ou por outras partes da rede, dando uma avaliação mais fiel da qualidade percebida da ligação.

A seguir são abordados os principais aspectos do sistema a ser implementado.

5.1 ARQUITETURA

O DiffServ foi a arquitetura escolhida, devido à sua simplicidade e escalabilidade. Utiliza-se a configuração representada na figura 5.1:

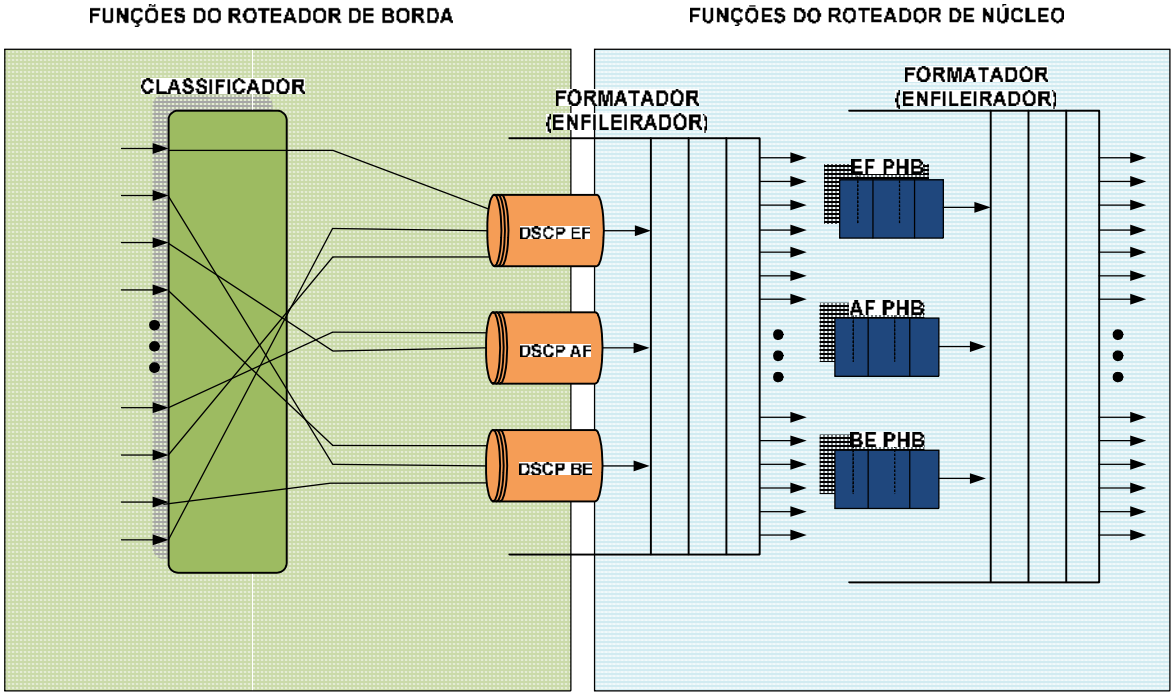

Figura 5.1 – Visão da Arquitetura DiffServ

No roteador de borda, os pacotes recebidos são diferenciados entre pacotes VoIP e demais pacotes. Essa diferenciação será feita de acordo com os parâmetros coletados pela ferramenta de monitoramento e medição de tráfego, no nosso caso o TShark. Após essa etapa, o campo ToS (Type of Service) do cabeçalho IP é redefinido por um campo da arquitetura DiffServ, o campo DS (DS field). Dos oito bits do campo ToS, os seis primeiros serão usados pelo DSCP para selecionar o PHB. É nessa etapa que é introduzida a adaptabilidade do nosso sistema. De acordo com o status da rede, diferentes prioridades são atribuídas aos pacotes VoIP utilizando-se a ferramenta Altq. Assim, pacotes semelhantes poderão ter tratamentos distintos, mas a qualidade observada no nó receptor deverá sempre ser mantida em um nível satisfatório.

5.2 PARÂMETROS DE AVALIAÇÃO DOS PACOTES VOIP

Os parâmetros utilizados para julgar o estado da rede são aqueles que mais afetam a percepção de qualidade da voz no VoIP: jitter, delta máximo e perda de pacotes. Na sessão Qualidade de Serviço abordada anteriormente neste trabalho foi explicado como esses parâmetros degradam a qualidade deste tipo de serviço.

Periodicamente, o TShark coletará informações destes parâmetros no nó receptor da rede. Estabelecemos para cada parâmetro limite superior e inferior para que haja a troca de priorização dos pacotes. Os limites adotados foram baseados nos valores usualmente utilizados quando se trata de tráfego VoIP, feito alguns ajustes para se enquadrarem na realidade do ambiente de testes que será descrito na sessão seguinte. A tabela 5.1 mostra os limites considerados no nosso sistema:

PARÂMETRO	LIMITE INFERIOR	LIMITE SUPERIOR
JITTER	13 ms	18 ms
DELTA MÁXIMO	130 ms	160 ms
PERDA DE PACOTE	0,5 %	5 %

Tabela 5.1 – Limites dos Parâmetros de QoS para Avaliação da Rede

Esses parâmetros são utilizados na classificação do estado da rede. Caso o valor observado seja abaixo do limite inferior, classificamos o estado da rede como BOM. Se estiver entre o limite inferior e superior, a classificação dada é OK. Já se estiver acima do limite superior, classificamos como RUIM.

Caso um dos três parâmetros esteja classificado como RUIM, aumentamos a priorização dos pacotes VoIP. Entretanto, só diminuimos a priorização dos pacotes VoIP quando todos os 3 parâmetros estiverem classificados como BOM. Deste modo os pacotes não são classificados somente pelo tipo (VoIP ou não), mas leva em conta também o estado da rede no momento da transmissão, inserindo a adaptabilidade no nosso sistema. Um exemplo dessa mudança de classificação é dada na figura 5.2:

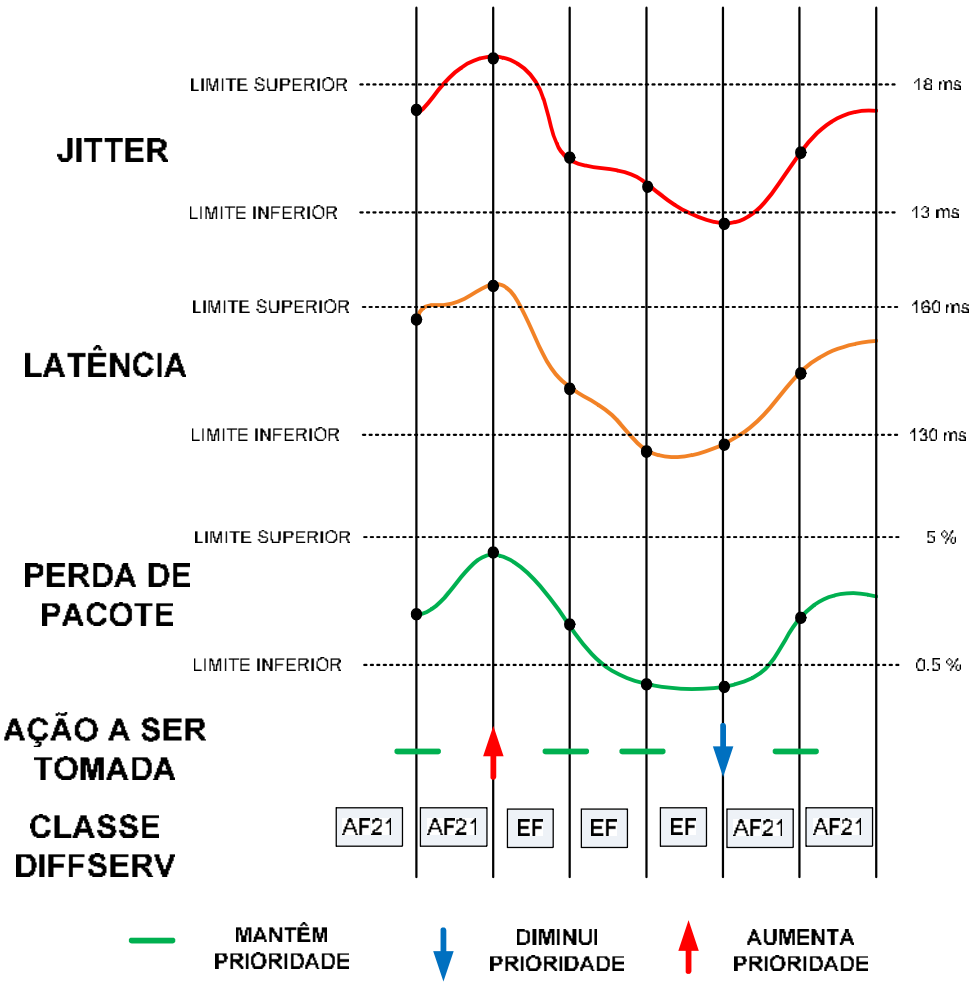

Figura 5.2 – Esquema de Avaliação e Classificação do Tráfego VoIP

6. CENÁRIO DE TESTES

Os conceitos abordados nos capítulos anteriores dão subsídio para se prosseguir para a parte prática do projeto. Neste tópico é detalhado o cenário de testes, descrevendo cada ferramenta utilizada e qual o papel dessa ferramenta na configuração da rede adotada.

6.1 FERRAMENTAS UTILIZADAS

6.1.1 TSHARK

O Tshark é um analisador de protocolo de rede capaz de capturar pacotes em tempo real ou ler pacotes salvos em um arquivo. O formato nativo de captura é o libcap, que também é o formato usado pelo tcpdump e outras várias ferramentas[15].

O TShark foi utilizado nos testes para se obter os parâmetros necessários para a análise do status da rede. Os opções utilizadas foram as seguintes:

```
# tshark –a duration:25 –d udp.port==(porta rtp),rtp –i eth0 –w (arquivo de saida) –z rtp,streams
```

-a duration:25: determina o tempo de captura de pacotes para 25 segundos;

-d udp.port==**(porta rtp)**,rtp: A porta utilizada pelo RTP será obtida pelo script que abordaremos mais adiante. Esse comando especifica que os pacotes udp dessa porta serão decodificados como rtp;

-i eth0: especifica o nome da interface que será ouvida pelo Tshark;

-w **(arquivo de saida)**: cria um arquivo com os dados capturados pelo TShark;

-z rtp,streams: coleta estatísticas de todas as streams RTP e calcula o delta máximo, a média e o máximo jitter e a porcentagem de perda de pacote.

6.1.2 ALTQ

O Altq (*Alternate Queueing*) é uma ferramenta desenvolvida por Kenjiro Cho que tem por finalidade prover uma plataforma flexível que possa suportar os diversos tipos de funções de QoS de uma rede. Além disso temos a possibilidade de gerenciamento de bandas de redes através de controle do tráfego de saída[16].

O Altq é utilizado para se fazer o condicionamento dos pacotes VoIP nos roteadores de borda, ou seja, marcá-los com o ToS desejado, e para enfileirá-los nos roteadores de núcleo de acordo com a classe DiffServ desejada.

6.1.3 *SHELL SCRIPT*

O *Shell script* nada mais é do que diversas linhas de comandos reunidas em um arquivo de texto simples. Esse scripts geralmente são usados para automatizar tarefas que são realizadas mais de uma vez, sendo usados até mesmo pelos sistemas Unix em atividades administrativas e de manutenção do sistema.

Uma das vantagens destes *shell script*s é que eles não precisam ser compilados, ou seja, basta apenas criar um arquivo texto qualquer, e inserir comandos à ele. Para dar à este arquivo a definição de *shell script*, teremos que incluir uma linha no começo do arquivo (#!/bin/bash) e torná-lo "executável", utilizando o comando chmod[17].

6.1.4 SJPHONE

Para realizar as ligações VoIP utilizamos o SJphone, que é um *softphone* capaz de realizar ligações para qualquer outro softphone rodando em PC/PDA, qualquer telefone IP stand-alone, ou telefone móvel ou fixo utilizando o provedor de serviços de telefonia e Internet[18]. O Sjphone possui suporte para SIP e H.323, roda em vários sistemas operacionais (como o Windows e o Linux), além de interoperabilidade com maioria dos produtos VoIP no mercado, sendo por tudo isso o software VoIP ideal para ser utilizado nos nossos testes.

Foram utilizadas nos testes as seguintes configurações:

Protocolo de Sinalização: SIP;

Codec: G.711;

Tempo de empacotamento: 20 ms;

Tamanho do buffer do driver: 32 ms.

6.2 CONFIGURAÇÃO DO AMBIENTE DE TESTES

A configuração adotada para a rede está mostrada na figura 6.1:

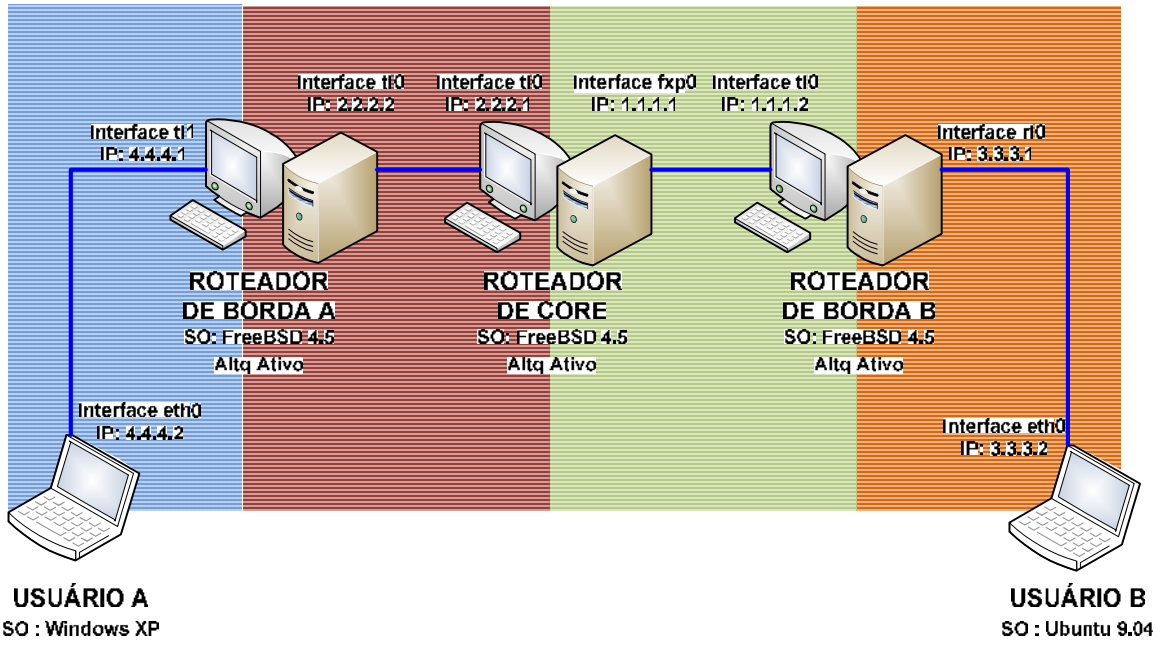

Figura 6.1 – Configuração da Rede de Testes

As duas máquinas das pontas (usuários A e B) são reponsáveis por realizar a ligação VoIP através do SJphone. Durante as chamadas, a qualidade das ligações é medida através de um analisador de protocolos de rede para que se possam tomar as decisões em relação à marcação do campo ToS dos pacotes VoIP. O TShark é usado por ser capaz de capturar, decodificar e extrair de uma stream RTP os parâmetros como jitter, delta máximo e porcentagem de perda de pacotes, como demostrado na figura figura 6.2.

```
======================== RTP Streams ========================
Src IP addr  Port     Dest IP addr  Port       SSRC          Payload
4.4.4.2      49164          3.3.3.2  49164   0x27D54524 ITU-T G.711 PCMA
3.3.3.2      49164          4.4.4.2  49164   0x060018A0 ITU-T G.711 PCMA
=============================================================
Pkts      Lost      Max Delta(ms)  Max Jitter(ms) Mean Jitter(ms) Problems?
3022   0 (0.0%)          139.07           13.57            8.21 X
3022   0 (0.0%)           41.50           15.74           14.96 X
```

Figura 6.2 – Estatísticas RTP obtidas pelo TShark

Uma observação deve ser feita nesse ponto: o delta máximo que o TShark nos retorna não é o mesmo que o *delay*. O *delay* é o atraso fim-a-fim dos pacotes, que geralmente deve ser abaixo de 150ms para que se possa ter uma boa qualidade de voz no receptor. O TShark não consegue calcular o atraso fim-a-fim pois a única informação disponível são os timestamps dos pacotes capturados. Se olharmos o cabeçalho de um pacote RTP (ou até mesmo cabeçalhos IP/UDP) veremos que não há outra informação temporal. Não há como saber quando o pacote foi enviado na rede, e se soubermos, não há garantia de que o transmissor e o receptor estão sincronizados. O delta máximo representa o máximo intervalo de tempo entre dois pacotes consecutivos. Em um caso ideal sem jitter, se usarmos o codec G.711 e um tempo de empacotamento de 20 ms, isso resultará em pacotes chegando ao receptor em intervalos de tempo de 20 ms, assim como o delta máximo. Iremos utilizar o delta máximo como parâmetro temporal nos nossos testes pois é um dos parâmetros dados pelo sniffer utilizado, além de ser a melhor aproximação que temos do *delay*.

Nos roteadores de borda ocorre a marcação do campo ToS dos pacotes VoIP, sendo que os pacotes que não são VoIP passam com a própria marcação, ou seja, não há a alteração do ToS (Type of Service) desses pacotes. Geralmente esses pacotes possuem o ToS setados como 0 decimal. Os parâmetros escolhidos para se filtrar os pacotes VoIP dos demais pacotes foram os campos do cabeçalho IP referentes ao protocolo e à porta de origem e porta de destino. O protocolo é o UDP (User Data Protocol) e as portas utilizadas pelo SIP (Session Initiation Protocol) e o RTP (Real Time Protocol) são específicas. O SIP geralmente usa a porta 5060 e o RTP usa portas acima da 49000. A ferramenta utilizada para se marcar os pacotes foi o Altq (Alternate Queueing), que é um sistema de enfileiramento e marcação de pacotes amplamente usado para implementação de QoS e roda em sistemas BSD UNIX. No Apêndice I mostramos como instalar o Altq no FreeBSD 4.5 e no Apêndice II detalhamos o arquivo de configuração do Altq nos roteadores de borda.

Com os pacotes já marcados, o papel de os enfileirar nas devidas classes DiffServ é feito pelo roteador de núcleo. A taxa de dados das duas placas de rede foi limitada em 200 Kbps para que se possa congestionar a rede mais facilmente, demonstrando com maior clareza os resultados dos testes. Nesse roteador de borda está configurado o Altq nas duas interfaces de rede com as regras mostradas na figura 6.3.

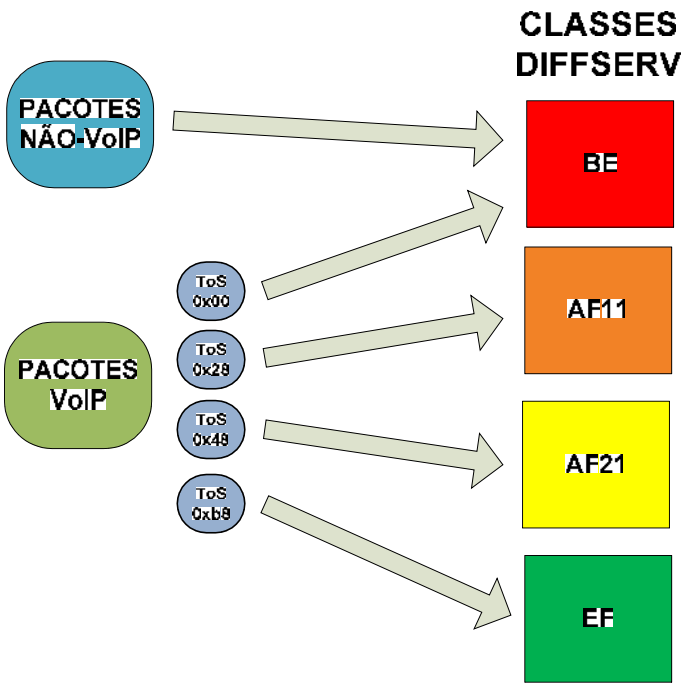

Figura 6.3 – Tomada de decisão do Altq no roteador de núcleo

O arquivo de configuração do Altq no roteador de núcleo está transcrito no Apêndice III.

7. TESTES E RESULTADOS

Com todo o cenário de testes configurado e funcionando em perfeitas condições, foram realizados testes para comprovar que o tráfego VoIP recebe um tratamento diferenciado em relação à outros tráfegos na rede utilizando a arquitetura DiffServ. Mais do que isso, assegurar que as tomadas de decisão são feitas de maneira adaptativa de acordo com o estado da rede, o que é feito através do script shell rodando nas máquinas dos usuários e que está transcrito no Apêndice V.

7.1 TESTES

Os seguintes testes foram realizados:

- **Teste A**: Ligação VoIP entre os usuários A e B sem outro tipo de tráfego na rede;
- **Teste B**: Ligação VoIP entre os usuários A e B com tráfego icmp na rede;
- **Teste C**: Ligação VoIP entre os usuários A e B com o script shell rodando na máquina dos usuários;

Os testes A e B têm a finalidade de demonstrar o comportamento do tráfego VoIP quando enquadrado nas diferentes classes DiffServ de acordo com o tráfego da rede. Já o teste C visa demonstrar a adaptabilidade do sistema proposto, assim como a manutenção da qualidade do serviço VoIP, independente do tráfego da rede.

Esses testes serão detalhados a seguir.

7.1.1 TESTE A

O Teste A foi realizado unicamente para demonstrar que, quando a rede não está congestionada, a diferença entre o tráfego VoIP tratado e o tráfego VoIP não tratado é praticamente nula.

Uma ligação VoIP de um minuto foi feita através do SJphone entre os usuários A e B sem nenhum outro tipo de tráfego na rede. Os pacotes VoIP são marcados nos roteadores de borda com o ToS 0x00. Desta maneira todos os pacotes VoIP irão obrigatoriamente entrar na classe BE, ou seja, sem nenhum tratamento DiffServ. O TShark captura os pacotes durante esse minuto para colher informações como jitter médio, delta máximo e porcentagem de perda de pacotes.

Posteriormente, o mesmo procedimento é repetido, só que dessa vez com o ToS marcado como 0x28, 0x48 e 0xb8 para que os pacotes sejam encaminhados para as classes AF11, AF21 e EF respectivamente. Os dados obtidos são os seguintes:

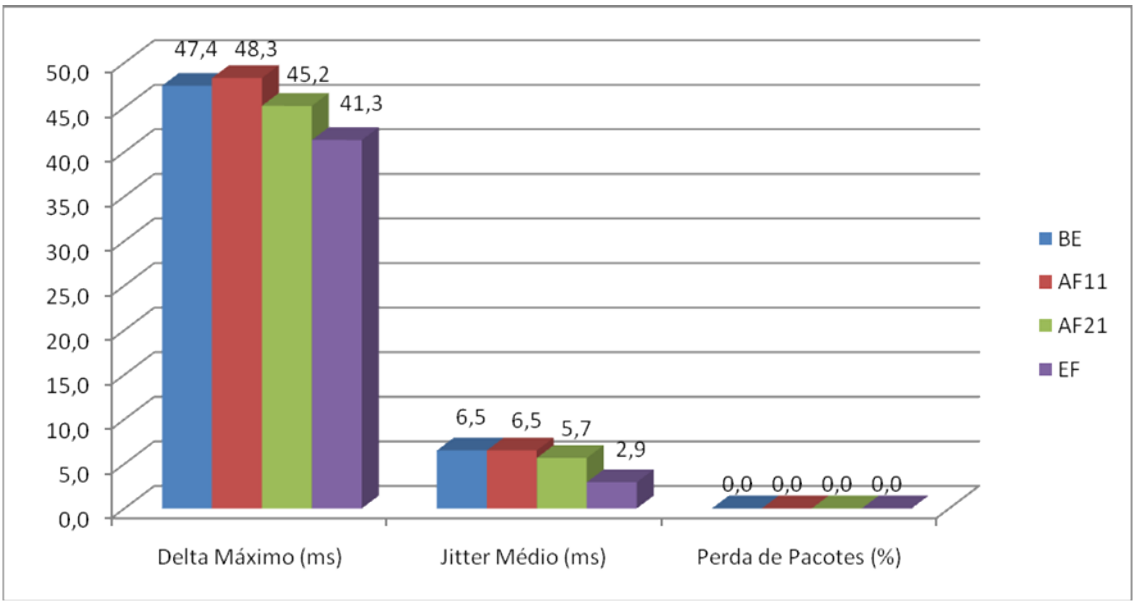

Figura 7.1 – Valores dos parâmetros obtidos no Teste A

Observa-se na figura 7.1 que a diferença entre os parâmetros dos pacotes VoIP sem tratamento (classe BE) e os pacotes VoIP com tratamento (classes AF11, AF21 e EF) é praticamente imperceptível nesse primeiro teste, sendo a qualidade de experiência auditiva semelhante em todos os casos. Os valores coletados nos três parâmetros estão bem abaixo dos limites inferiores mostrados na tabela 6.1. Isso acontece pois o tráfego VoIP possui em média uma taxa de 85 kbps quando utiliza o codec G.711, e como a rede não esta congestionada com outros tipos de tráfegos, toda a banda de 200 kbits pôde ser utilizada pelo tráfego VoIP, independente da classe escolhida. Mesmo assim, já podemos perceber pequena vantagem quando é utilizada a classe EF, principalmente em relação ao jitter.

Isto fica mais claro observando o Apêndice III. Quando utilizada a classe BE, como não há outro tipo de tráfego, há 200 kbits de banda livre que pode ser utilizado pelo VoIP. Na classe AF11 temos a garantia de 15 kbits de banda e há disponibilidade de se pegar emprestado da classe default o restante de banda necessária. A classe AF21 é semelhante à AF11, mudando somente a quantidade de banda garantida de 15 Kbits para 45 Kbits. Já na EF não há possibilidade de se pegar emprestado banda da classe default, mas como é garantido uma taxa de 90 Kbps, e o VoIP usa em torno de 85 Kbps, o VoIP passa tranquilamente. Essa garantia de 90 Kbps explica a pequena melhora nos valores da classe EF.

7.1.2 TESTE B

O Teste B tem como finalidade a análise do comportamento do tráfego VoIP quando tratado nas diferentes classes DiffServ e compartilhando a rede com outro tipo de tráfego, nesse caso o tráfego ICMP.

O tráfego ICMP foi escolhido para realizar o Teste B devido ao comportamento uniforme apresentado e a fácil manipulação dos parâmetros quando gerado pelo comando **ping**:

```
ping –s 5000 3.3.3.2
```

No exemplo acima, pacotes ICMP de 5 kbytes são mandados para o IP 3.3.3.2 (usuário B). Logo abaixo está traçada a relação entre o tamanho dos pacotes ICMP e a taxa ocupada na rede:

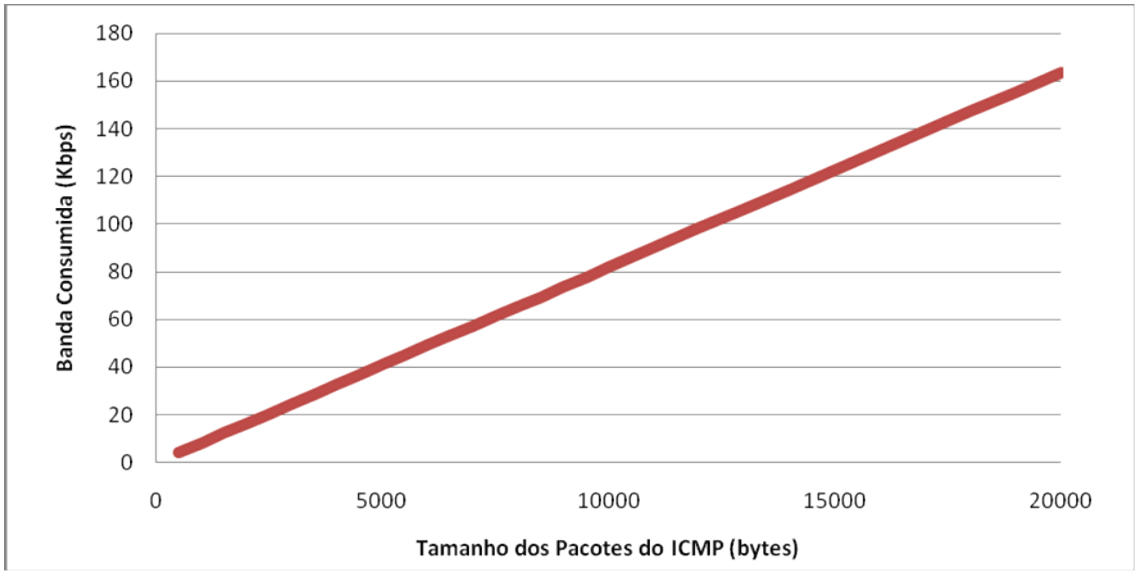

Figura 7.2 – Tamanho dos Pacotes x Banda Consumida

Vemos na figura 7.2 que para cada 5 kbits de tamanho do pacote ICMP uma taxa de aproximadamente 40 Kbps é ocupada na rede.

Foi feita a coleta de valores dos mesmos parâmetros do teste A dentro de 4 cenários com perfis de tráfego diferentes:

- 1º Cenário: Tráfego ICMP com taxa de 40 kbps. Taxa de ocupação total da banda da rede (tráfego VoIP + ICMP): 62,5%;
- 2º Cenário: Tráfego ICMP com taxa de 80 kbps. Taxa de ocupação total: 82,5%;
- 3º Cenário: Tráfego ICMP com taxa de 120 kbps. Taxa de ocupação total: 102,5%;
- 4º Cenário: Tráfego ICMP com taxa de 160 kbps. Taxa de ocupação total: 12,5%.

Como o tráfego VoIP possui uma taxa de aproximadamente 85 kbps e temos uma banda de 200 Kbits, teremos o 1º e o 2º cenário com ocupação abaixo do limite da banda. No 3º cenário estaremos no limiar do total da banda. Já no 4º cenário a taxa de dados na rede será maior que a banda disponível, exemplificando uma situação em que a rede esteja totalmente congestionada.

Em cada cenário foi realizada uma ligação VoIP de um minuto com SJphone entre usuário A e usuário B. Como no Teste A, há um primeiro momento onde marcamos os pacotes VoIP nos roteadores de borda com o ToS 0x00 afim de que eles sejam encaminhados para a classe BE e os dados de jitter médio, delta máximo e percentual de perda de pacotes são coletados. Posteriormente, o mesmo procedimento é feito com marcações de ToS em 0x28, 0x48 e 0xb8, para serem encaminhados para as classes AF11, AF21 e EF.

Os resultados coletados são mostrados a seguir.

1º CENÁRIO: TAXA DE OCUPAÇÃO TOTAL – 62,5%

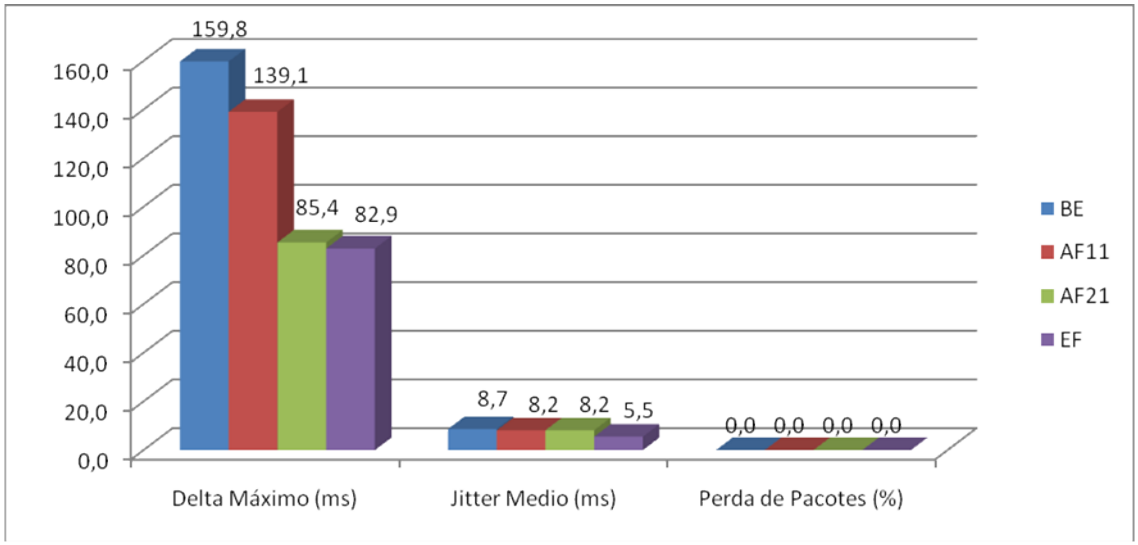

Figura 7.3 – Valores dos parâmetros obtidos no 1º Cenário do Teste B

Nota-se na figura 7.3 que os valores de delta máximo e jitter médio são maiores que os obtidos no Teste A, em que não havia tráfego concorrente com o VoIP, mas em todos os casos os valores não superam os limites máximo descritos na tabela 6.1. Também não houve perda de pacotes, pois o tráfego ICMP não foi suficiente para congestionar a rede. A qualidade da experiência auditiva foi satisfatória, como era de se esperar pelos valores obtidos no gráfico acima. Fica claro também a melhoria obtida nos valores dos parâmetros a medida que o tráfego é encaminhado para classes DiffServ de melhor disciplina de tratamento.

Neste cenário o uso da classe BE é suficiente para uma qualidade de experiência satisfatória.

2º CENÁRIO: TAXA DE OCUPAÇÃO TOTAL – 82,5%

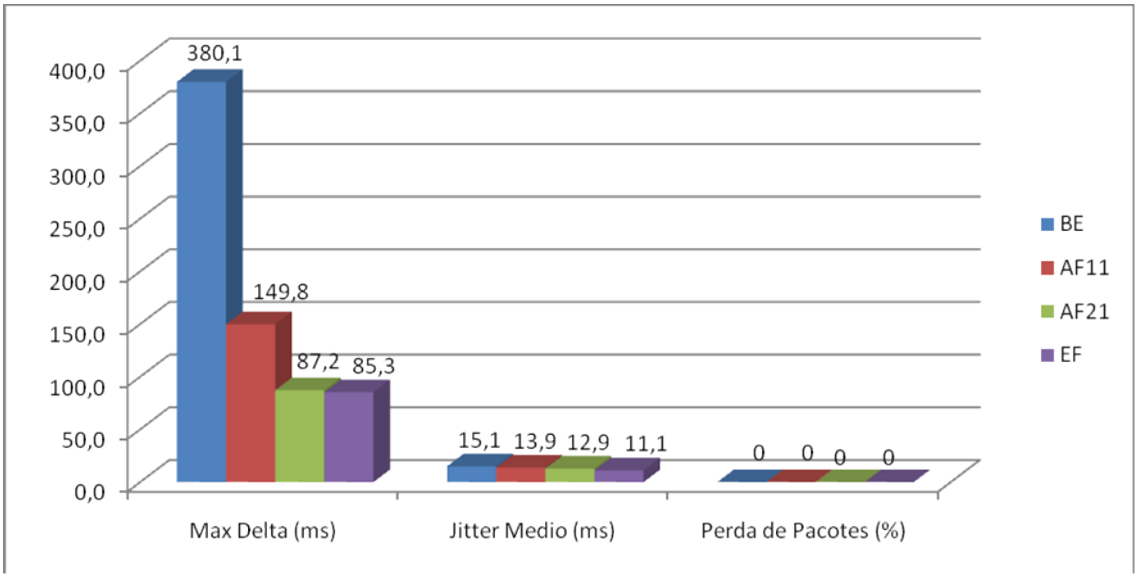

Figura 7.4 – Valores dos parâmetros obtidos no 2º Cenário do Teste B

Percebemos que a qualidade de experiência começa ser afetada quando não há tratamento do tráfego VoIP. Isto é confirmado pelos valores dos parâmetros obtidos neste teste, mostrados na Figura 7.4. Quando utilizada a classe BE, o delta máximo extrapola em muito o limite superior de 160 ms.O jitter se mantêm entre os limites e não há perda de pacotes, pois o tráfego total na rede não supera 200 Kbps. Na classe AF11 os valores de delta máximo e jitter médio começam a se aproximar do limite superior, e apesar de ser inteligível, a qualidade de experiência não é satisfatória. Para as classes AF21 e EF a qualidade de experiência melhora, apresentando parâmetros semelhantes entre eles e abaixo dos limites propostos na tabela 6.1.

Neste cenário o uso da classe AF21 é suficiente para uma qualidade de experiência satisfatória.

3º CENÁRIO: TAXA DE OCUPAÇÃO TOTAL – 102,5%

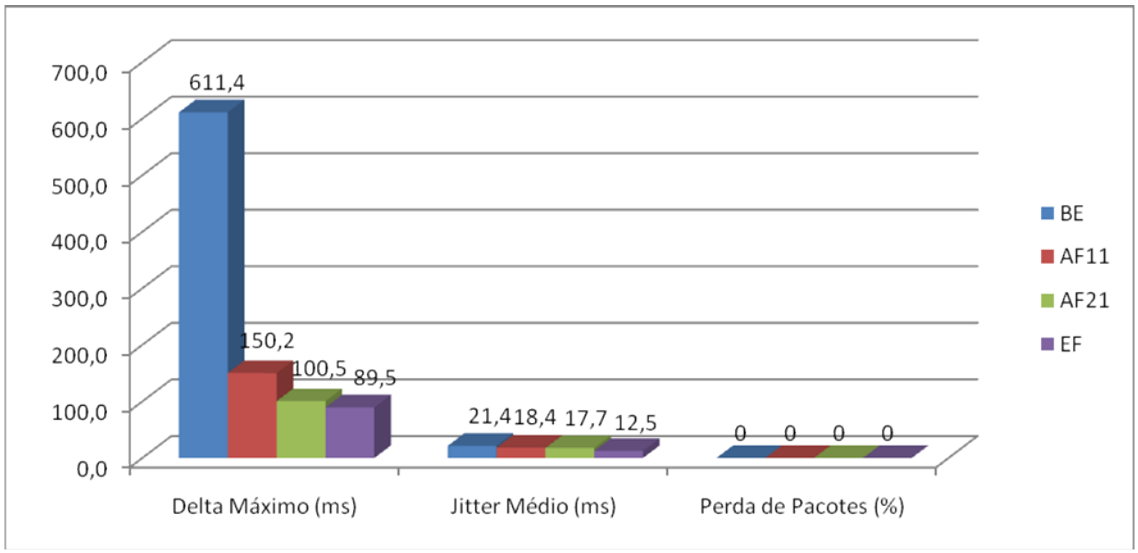

Figura 7.5 – Valores dos parâmetros obtidos no 3º Cenário do Teste B

Como temos um tráfego ICMP com taxa próxima de 120 Kbps e um tráfego VoIP com aproximadamente 85 Kbps, estamos no limiar da capacidade da rede, mas ainda não há perda de pacotes. Como mostra a figura 7.5, o delta máximo da classe BE é quase quatro vezes maior do que o limite superior definido. Para as outras classes, esse parâmetro apresenta valores abaixo do limite superior, e no caso das classes AF21 e EF, até mesmo abaixo do limite inferior de 130 ms. Os valores de jitter médio estão próximos ou acima do limite superior definido, exceto no caso da classe EF.

A qualidade de experiência observada foi satisfatória apenas quando utilizada a classe EF, mostrando o quanto é decisivo o parâmetro de jitter na percepção auditiva.

4º CENÁRIO: TAXA DE OCUPAÇÃO TOTAL – 122,5%

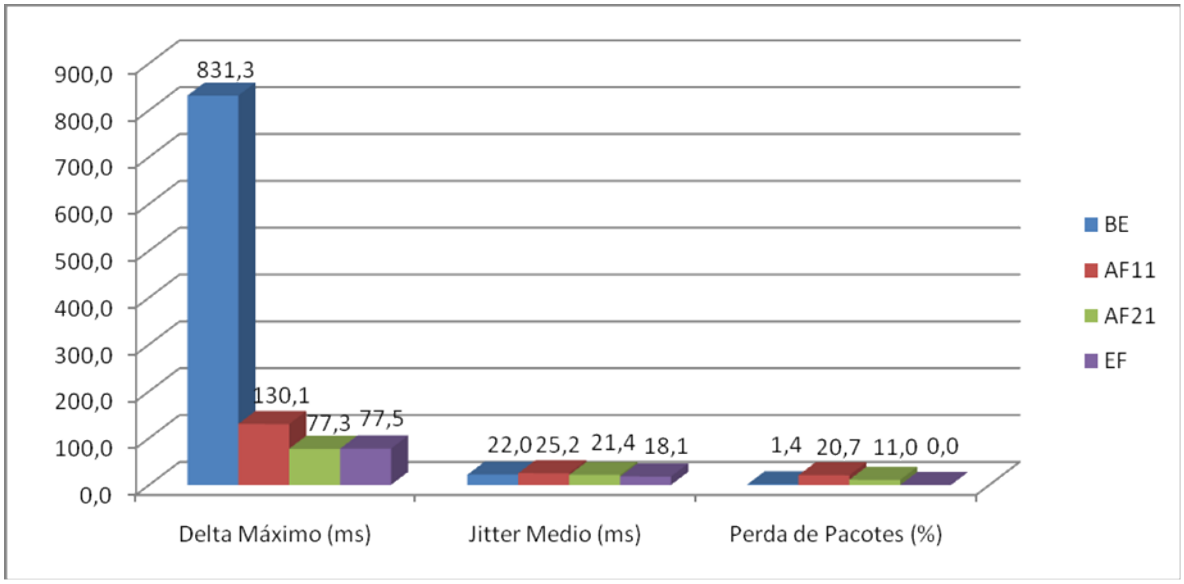

Figura 7.6 – Valores dos parâmetros obtidos no 4º Cenário do Teste B

Temos neste cenário o pior caso, onde a rede encontra-se totalmente congestionada. Como a taxa de bits supera os 200 Kbps, temos pela primeira vez a presença de perda de pacotes. Como mostrado na figura 7.6, os valores de jitter de todas as classes superam o limite superior de 18 ms. No caso do delta máximo, somente a classe BE supera o limite máximo.

O valor da porcentagem de perda de pacotes no caso do BE é interessante, pois é menor do que no caso do AF11 e AF21. Isso acontece pois na classe BE o tráfego VoIP e ICMP concorrem pela mesma banda, sendo tratados de forma igual, e os pacotes são descartados independentemente do tipo. Já nas classes AF, somente uma parcela da banda é garantida para o tráfego VoIP, sendo que o restante é pego emprestado caso haja disponibilidade na classe default. Como a classe default está saturada com o tráfego ICMP, a parcela de banda que as classes AF conseguem pegar emprestado é menor, descartando maior quantidade de pacotes VoIP.

Neste cenário somente a classe EF consegue prover uma qualidade de experiência satisfatória.

7.1.3 TESTE C

O Teste C possui o objetivo de demonstrar a adaptabilidade que o *shell script* desenvolvido oferece ao tráfego VoIP frente a um ambiente com tráfego concorrente variável.

Em um período de nove minutos é realizada uma ligação VoIP entre os usuários A e B. O *shell script* mostrado no Apêndice IV é executado no usuário B e faz a avaliação a cada 30 segundos do fluxo RTP vindo do usuário A. Os valores da tabela 6.1 são usados na avaliação dos parâmetros de jitter médio, delta máximo e porcentagem de perda de pacotes. Após essa etapa, é decidido se a classe DiffServ utilizada deve ser alterada. Em caso positivo, um comando ssh (secure shell) executa o script do **Apêndice V** no roteador de borda A, alterando o arquivo de configuração do Altq (altq.conf). O teste começa sem outro tipo de tráfego e a cada minuto é introduzido um fluxo ICMP com uma taxa de transmissão diferente. O teste foi executado da seguinte forma:

- 1º minuto: Sem tráfego;
- 2º minuto: Tráfego icmp de 40kbps;

- 3º minuto: Tráfego icmp de 80kbps;
- 4º minuto: Tráfego icmp de 120kbps;
- 5º minuto: Tráfego icmp de 160kbps;
- 6º minuto: Tráfego icmp de 120kbps;
- 7º minuto: Tráfego icmp de 80kbps;
- 8º minuto: Tráfego icmp de 40kbps;
- 9º minuto: Sem tráfego.

O tráfego ICMP foi gerado do usuário A para o usuário B, afim de concorrer com o tráfego VoIP dado no mesmo sentido.

A ligação VoIP é toda monitorada no TShark e as características do delta e do jitter são observadas na figura 7.7:

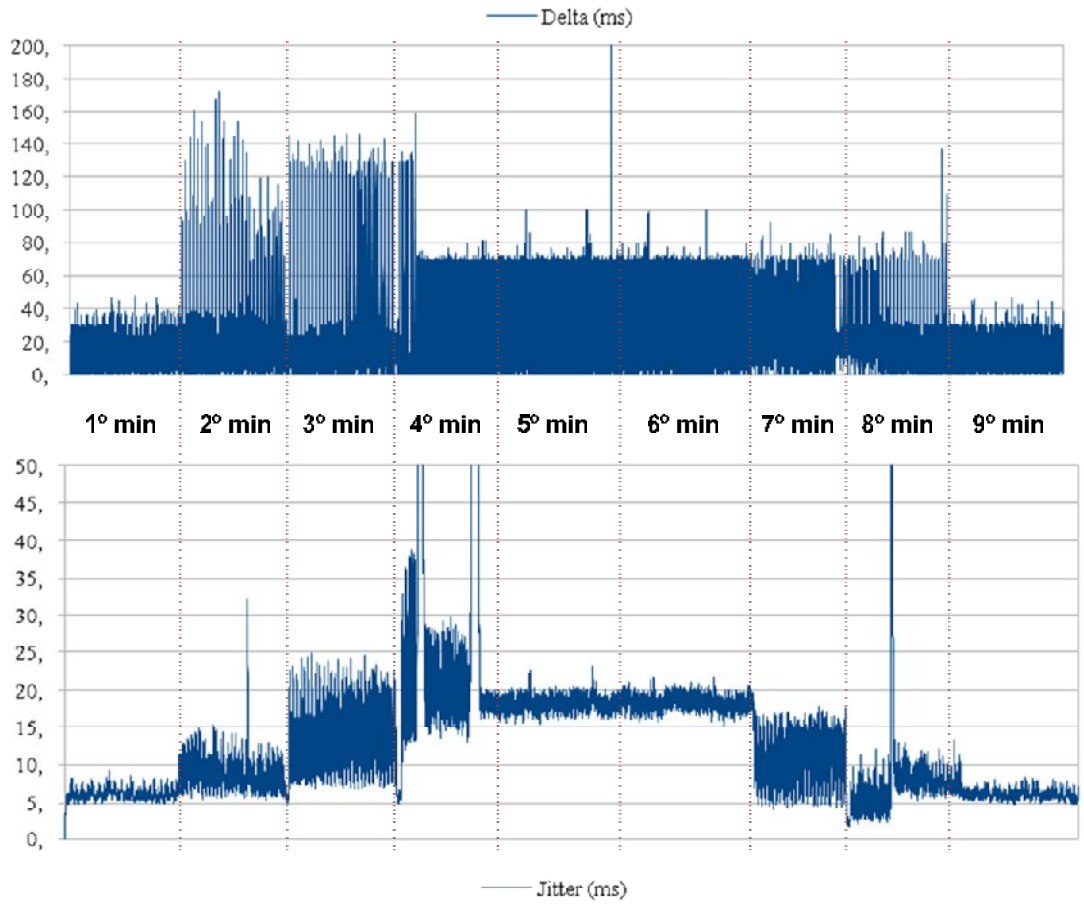

Figura 7.7 – Valores de jitter e delta obtidos no Teste C

A porcentagem de perda de pacotes observada em cada minuto é demostrada na figura 7.8:

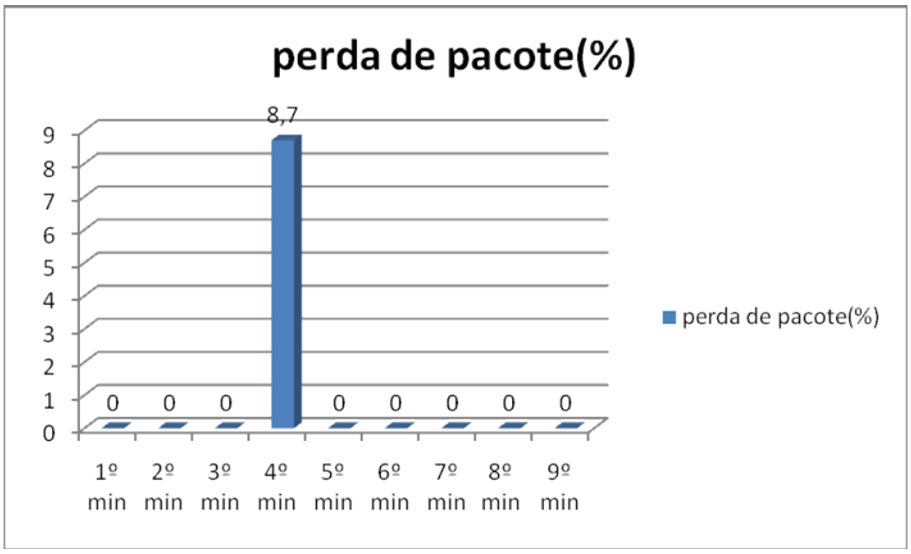

Figura 7.8 – Valores de porcentagem de perda de pacotes obtidos no Teste C

Percebemos que somente no quarto minuto da ligação há a presença de perda de pacotes VoIP na rede. Provavelmente a causa desse pico seja dado pelo momento de transição que o quarto minuto representa: é quando chegamos no limiar da taxa de transmissão da rede (beirando os 200 Kbps somando o tráfego VoIP e ICMP) apesar de a classe de encaminhamento do tráfego VoIP ainda estar configurada como AF11, não conseguindo priorizar completamente o tráfego de voz quando um tráfego intenso utiliza a classe *default*.

Uma análise detalhada de cada minuto é dada abaixo:

1º Minuto: Por padrão, os pacotes VoIP são encaminhados para classe BE, mas como não há tráfego concorrente o delta máximo se mantém por volta de 45 ms e o jitter por volta de 6 ms. Durante este tempo o script não altera a classe de encaminhamento DiffServ do tráfego VoIP, pois os valores medidos estão abaixo do limite máximo estipulado para estes parâmetros. A saída do script shell é mostrada abaixo:

```
Capturing on eth0
218 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2461
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    0.0        Bom
   Max Delta       44.96      Bom
   Mean Jitter     6.16       Bom
   Max Jitter      8.27
--------------------------------------------------------------------
ACAO
Diminuir Prioridade
--------------------------------------------------------------------
Marcacao atual 0x00
```

```
Marcacao nao alterada
```

2º Minuto: Com a introdução de um tráfego de 40kbps, começamos a observar um aumento dos valores de jitter e delta máximo. Apesar deste aumento, somente o delta máximo extrapola o limite superior de 160 ms, o que é suficiente para que o script altere a classe DiffServ do tráfego VoIP de BE para AF11.

```
Capturing on eth0
222 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2664
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    0.0         Bom
   Max Delta       164.09      Ruim
   Mean Jitter     9.04        Bom
   Max Jitter      15.42
--------------------------------------------------------------------
ACAO
Aumentar Prioridade
--------------------------------------------------------------------
Marcacao atual 0x00
Marcacao alterada para 0x28
```

Ainda no segundo minuto, uma outra avaliação é feita, e podemos perceber que com a nova classe de encaminhamento DiffServ a qualidade dos três parâmetros se enquadram dentro dos limites satisfatórios.

```
Capturing on eth0
231 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2679
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    0.0         Bom
   Max Delta       142.75      Ok
   Mean Jitter     9.48        Bom
   Max Jitter      23.76
--------------------------------------------------------------------
ACAO
Nada a fazer
--------------------------------------------------------------------
```

3º Minuto: Quando o terceiro minuto começa e o tráfego icmp passa para uma taxa de 80kbps, os valores de jitter e delta apresentam um novo aumento. Entretanto, a medida de delta máximo e jitter não foi suficientemente grande para classificar a qualidade destes parâmetros como RUIM, não alterando a classe DiffServ. A qualidade de experiência neste momento piora, mas não chega a compremeter o inteligibilidade da comunicação.

```
Capturing on eth0
225 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2798
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    0.0         Bom
   Max Delta       146.11       Ok
   Mean Jitter     14.24        Ok
   Max Jitter      36.15
--------------------------------------------------------------------
ACAO
Nada a fazer
--------------------------------------------------------------------
```

4º Minuto: Um tráfego icmp de 120kbps é lançado na rede, e apesar de o delta máximo diminuir significamente, o valor de jitter extrapola o limite máximo de 18 ms, além de ocorrer perda de pacotes VoIP. Baseado na qualidade desses parâmetros, o script altera a classe de encaminhamento VoIP para AF21, como podemos abservar a seguir:

```
Capturing on eth0
198 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2906
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    8.4          Ruim
   Max Delta       76.49        Bom
   Mean Jitter     20.37        Ruim
   Max Jitter      25.70
--------------------------------------------------------------------
ACAO
Aumentar Prioridade
--------------------------------------------------------------------
Marcacao atual 0x28
```

```
Marcacao alterada para 0x48
```

Com a nova alteração, os valores dos parâmetros diminuem um pouco, porém a porcentagem de perda de pacotes e o jitter médio ainda se encontram acima dos valores máximos estipulados. A qualidade de experiência auditiva melhora, mas ainda apresenta certa deficiência. O script altera novamente a classe de encaminhamento VoIP para EF, como descrito abaixo:

```
Capturing on eth0
213 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2863
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    7.1          Ruim
   Max Delta       70.24        Bom
   Mean Jitter     19.97        Ruim
   Max Jitter      24.45
--------------------------------------------------------------------
ACAO
Aumentar Prioridade
--------------------------------------------------------------------
Marcacao atual 0x48
Marcacao alterada para 0xb8
```

5º Minuto: O tráfego icmp de 160kbps congestiona a rede. Porém, com o VoIP na classe DiffServ EF, a banda garantida de 90kbits faz a qualidade de experiência auditiva manter um nível bom, apesar do jitter médio extrapolar o limite máximo. O script nada faz, pois não existe classe melhor para enquadrar o VoIP.

```
Capturing on eth0
264 packets captured
--------------------------------------------------------------------
Porta RTP = 49154
--------------------------------------------------------------------
Capturing on eth0
2731
--------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
--------------------------------------------------------------------
   Lost packets    0.0          Bom
   Max Delta       76.11        Bom
   Mean Jitter     18.34        Ruim
   Max Jitter      20.60
--------------------------------------------------------------------
ACAO
Aumentar Prioridade
```

```
------------------------------------------------------------------
Marcacao atual 0xb8
Marcacao nao alterada
```

6º Minuto: O tráfego icmp é reduzido novamente para 120kbps, mas os valores de jitter e delta máximo se mantém praticamente inalterados. A qualidade de experiência auditiva também mantém o mesmo padrão, explicado pela garantia de 90kbits da banda. O script não altera a classe, pois os valores do parâmetros coletados não permitem alteração.

```
Capturing on eth0
250 packets captured
------------------------------------------------------------------
Porta RTP = 49154
------------------------------------------------------------------
Capturing on eth0
2713
------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
------------------------------------------------------------------
   Lost packets    0.0         Bom
   Max Delta       77.30       Bom
   Mean Jitter     18.27        Ruim
   Max Jitter      20.97
------------------------------------------------------------------
ACAO
Aumentar Prioridade
------------------------------------------------------------------
Marcacao atual 0xb8
Marcacao nao alterada
```

7º Minuto: O tráfego icmp é novamente de 80kbps e valor do delta máximo permanece inalterado, por volta de 75 ms. O jitter, porém, apresenta diminuição para um valor classificado como OK. O script não faz nada, pois o valor do jitter ainda não foi considerado suficientemente bom para diminuir a classe de enfileiramento do VoIP.

```
Capturing on eth0
258 packets captured
------------------------------------------------------------------
Porta RTP = 49154
------------------------------------------------------------------
Capturing on eth0
2801
------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
------------------------------------------------------------------
---------
   Lost packets    0.0         Bom
   Max Delta       76.09       Bom
   Mean Jitter     11.08       Ok
   Max Jitter      17.40
```

```
------------------------------------------------------------------
ACAO
Nada a fazer
```

8º Minuto: Com nova redução do tráfego icmp para 40kbps, jitter e delta apresentaram valores abaixo do limite mínimo estipulado. O script então faz a alteração para a classe AF21, o que não acarreta alteração na qualidade de experiência auditiva, que permanece boa.

```
Capturing on eth0
218 packets captured
------------------------------------------------------------------
Porta RTP = 49154
------------------------------------------------------------------
Capturing on eth0
2655
------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
------------------------------------------------------------------
   Lost packets    0.0         Bom
   Max Delta       76.18       Bom
   Mean Jitter     5.47        Bom
   Max Jitter      17.10
------------------------------------------------------------------
ACAO
Diminuir Prioridade
------------------------------------------------------------------
Marcacao atual 0xb8
Marcacao alterada para 0x48
```

9º Minuto: No nono minuto não existe mais tráfego e os valores dos parâmetros coletados tendem aos valores observados no primeiro minuto. O script faz a alteração da classe do VoIP para AF11.

```
Capturing on eth0
213 packets captured
------------------------------------------------------------------
Porta RTP = 49154
------------------------------------------------------------------
Capturing on eth0
2655
------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
------------------------------------------------------------------
   Lost packets    0.0          Bom
   Max Delta       106.84       Bom
   Mean Jitter     8.46         Bom
   Max Jitter      14.02
------------------------------------------------------------------
ACAO
Diminuir Prioridade
------------------------------------------------------------------
Marcacao atual 0x48
```

```
Marcacao alterada para 0x28
```

Após mais uma análise o script altera a classe de enfileiramento do tráfego VoIP novamente para BE, já que jitter e delta máximo apresentaram valores classificados como bons e a qualidade da experiência auditiva é boa.

```
Capturing on eth0
231 packets captured
------------------------------------------------------------------------
Porta RTP = 49154
------------------------------------------------------------------------
Capturing on eth0
2496
------------------------------------------------------------------------
AVALIACAO
   Parametro       Valor    Qualidade
------------------------------------------------------------------------
   Lost packets    0.0         Bom
   Max Delta       118.89      Bom
   Mean Jitter     6.48        Bom
   Max Jitter      13.21
------------------------------------------------------------------------
ACAO
Diminuir Prioridade
------------------------------------------------------------------------
Marcacao atual 0x28
Marcacao alterada para 0x00
```

Ao final dos nove minutos, pudemos perceber que a qualidade de experiência foi mantida em um nível satisfatório durante toda a ligação. Os parâmetros de jitter, delta máximo e porcentagem de perda do tráfego VoIP foram controlados. Quando apresentaram degradação devido ao tráfego icmp concorrente foram todos atenuados para valores aceitáveis com a alteração da classe DiffServ do tráfego RTP. O sistema apresentou a adaptabilidade desejada, sendo flexível a realidade momentânea da rede e reservando os recursos de forma inteligente e sem desperdício.

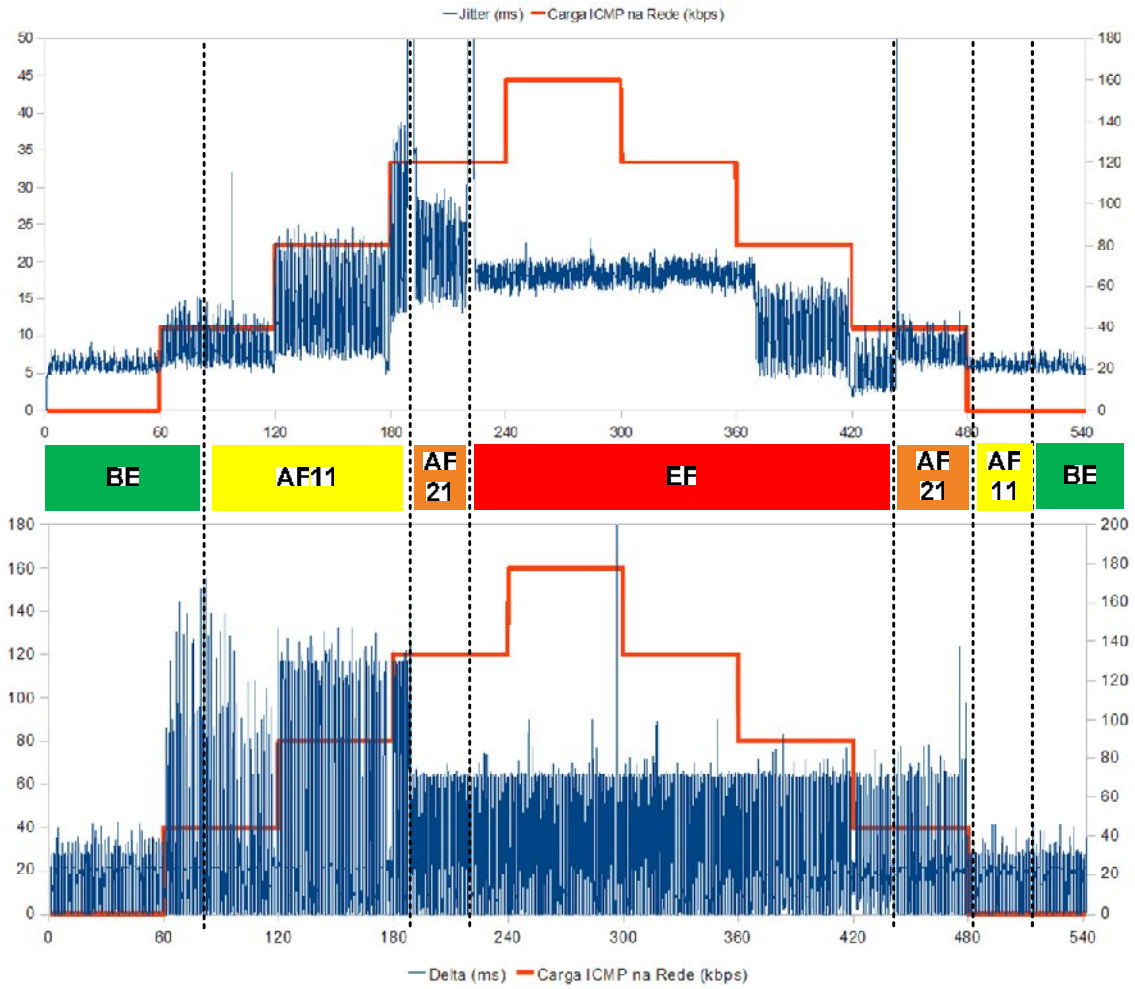

Figura 7.9 – Mudança das Classes DiffServ durante o Teste C

8. CONCLUSÃO

O objetivo deste trabalho foi abordar os principais conceitos que são envolvidos quando se trata de Voz sobre IP. Primeiramente foi explicado o que realmente é VoIP e como é feita a transformação da voz em pacotes, permitindo que o que é falado trafegue nas redes IP. Os principais protocolos envolvidos no processo de uma chamada e suas características foram citados e deram embasamento teórico para entender como os mecanismos funcionam em conjunto, dando o suporte necessário que o VoIP necessita.

Após entender a tecnologia, partiu-se para o conceito de Qualidade de Serviço. Exemplificou-se as três principais arquiteturas utilizadas atualmente para se garantir qualidade de serviço (Best Effort, Integrated Services e Differentiated Services) e como os parâmetros usados por cada uma influenciam diretamente nos serviços VoIP. Analisando essas três opções, foi feita a escolha que melhor se encaixava na proposta, escolhendo-se o DiffServ.

Outra área essencial na proposta é a Gerência de Redes. Através dos conceitos de estruturas e áreas funcionais foi passado o que as normas e padrões recomendam para se estruturar essa área, que tem importância central na nossa proposta, a de propiciar adaptabiliadade ao nosso sitema.

Esmiuçando cada tema acima, houve a possibilidade de ter uma visão de como o VoIP é tratado atualmente, vendo os pontos positivos e negativos das arquiteturas utilizadas. Partindo disto, foi feita a proposta de uma arquitetura que busca melhorar o tratamento dos pacotes VoIP na rede através de um enfoque adaptativo do sistema.

Ficou claro que o atual modelo de enfileiramento de melhor esforço praticado na maioria das redes não é capaz de prover a qualidade de serviço necessária para o tráfego VoIP. Quando esse modelo é utilizado em ambientes com tráfegos concorrentes, como os realizados nos testes da sessão 7, a qualidade de experiência auditiva fica muito abaixo dos níveis considerados como aceitáveis. Sendo assim, é evidente a necessidade de um novo modelo de tratamento do tráfego VoIP.

O DiffServ foi escolhido como modelo de tratamento de tráfego por apresentar escalabilidade e facilidade de implementação. Os resultados obtidos quando utilizadas classes de encaminhamento DiffServ foram superiores aos obtidos no modelo de melhor esforço, mostrando eficiência quando tratamos de tráfego em tempo real e de grande sensibilidade a perdas.

Dentre as muitas ferramentas de implementação do modelo DiffServ, o Altq apresentou-se como o mais simples, por não necessitar de ferramentas complementares para o seu uso. Após recompilar o kernel do FreeBSD, sua configuração limita-se em alterar um arquivo texto com as regras DiffServ desejadas. Essa poderosa ferramenta é capaz de manipular bem os parâmetros DiffServ, além de monitorar em tempo real as mudanças ocorridas na rede.

A adaptabilidade, principal foco deste projeto, foi alcançado por meio do *shell script*, que conseguiu reunir e interagir as várias ferramentas citadas durante todo o documento. No Teste C ficou demonstrado que a proposta de adaptabilidade foi eficiente em prover qualidade de serviço para o tráfego VoIP pois durante toda a ligação a qualidade de experiência observada foi satisfatória, não havendo uso desnecessário de recursos da rede.

Todos os testes apresentaram resultados dentro dos valores esperados na teoria, e a proposta criada conseguiu atingir os objetivos estabelecidos inicialmente.

8.1 PROJETOS FUTUROS

Encontramos certa dificuldade para implementar a marcação dos pacotes VoIP nos roteadores de borda e enfileiramento dos mesmos no roteador de núcleo utilizando o Altq nas versões mais recentes do FreeBSD. Isso porque nessas novas versões desse sistema operacional o Altq funciona em

conjunto com outras ferramentas, como o PF (Packet Filter) e o IPF (IPFILTER Firewall). Uma implementação utilizando esse conjunto de ferramentas em versões mais recentes do FreeBSD seria interessante para acompanhar as mudanças feitas no sistema operacional.

Outro ponto a ser mudado seria conseguir o parâmetro de *delay* dos pacotes VoIP ao invés do delta máximo. Como explicado anteriormente, há diferença entre esses parâmetros, e utilizando o *delay* podemos ter uma avaliação mais fiel do status da rede.

Além dessa troca do delta pelo *delay*, pode-se adicionar outros parâmetros na avaliação da rede. Quanto mais informações estiverem incluídas na análise da rede, mais exatas serão as mudanças tomadas.

Por fim, mostramos nesse projeto um algorítmo para a avaliação da rede baseado nos três parâmetros dos pacotes VoIP, sendo aumentada a classe DiffServ se um deles estiverem RUIM, e diminuindo a classe se todos estiverem BOM. Esse algorítmo pode ser mudado visando maximizar os efeitos da mudança de classe.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Integração Profissional Brasil, Revista IP (28/05/2009): www.revistaip.com.br/article.php?a=211

[2] Qualidade de Serviço em VoIP (01/06/2009): www.rnp.br/newsgen/0005/qos_voip1.html

[3] MA, Angus. *Voice over IP*. Spirent Communications, 2001

[4] BATES, Regis J. *Broadband Telecommunications Handbook*. MacGraw-Hill, 2002

[5] VoIP Signaling Protocols (12/06/2009): //voip-facts.net/h323.php

[6] The Session Initiation Protocol (13/06/2009): //voip-facts.net/sip.php

[7] Visão Geral sobre QoS (9/06/2009): technet.microsoft.com/pt-br/library/cc757887(WS.10).aspx

[8] FERGUSON, Paul; HUSTON, Geoff. *Quality of Service in the Internet: Fact, Fiction or* Compromise. John Wiley & Sons, 1998

[9] WANG, Zheng. *Internet QoS: architectures and mechanisms for Quality of Service*. ISBN 1-55860-608-4, Academic Press, 2001.

[10] BARRETO, Carlos; SOARES, Osnildo. *Telecommunication Management Network*, 1999

[11] DAVIDSON, Jonathan; PETERS, James; BHATIA, Manoj; KALIDINDI, Satish; MUKHERJEE, Sudipto. *Voice over IP Fundamentals*. Second Edition: Cisco Press, 2006

[12] UDUPA, Divakara K.. *NetworkManagement Systems Essentials*. Estados Unidos: McGraw-Hill, 1996

[13] MULLER, Nathan J.. *Network Planning, Procurement & Management*. McGraw-Hill, 1996

[14] Instalando o Altq 3.1 (30/10/2009): http://src.scbd.net.id/FreeBSD/contrib/altq/altq-3.1/INSTALL

[15] Tshark - Dump and analyze network traffic (05/11/2009): http://www.wireshark.org/docs/man-pages/tshark.html

[16] ALTQ: Alternate Queueing BSD UNIX (08/11/2009): http://www.sonycsl.co.jp/~kjc/software.html

[17] Programando em shell-script (08/11/2009): http://www.devin.com.br/shell_script/

[18] SJLabs: Voice over IP Software (20/11/2009): http://www.sjlabs.com/sjp.html

APÊNDICE I – INSTALANDO O ALTQ

Primeiramente, é importante ressaltar que a escolha do FreeBSD 4.5 como sistema operacional não foi feita por acaso. De acordo com os documentos da última versão do Altq (versão 3.1), o FreeBSD é a última versão do FreeBSD que dá suporte ao Altq de forma independente. A partir do FreeBSD 4.6 o Altq passou a ser usado juntamente com outras ferramentas, o PF (Packet Filter) ou o IPFW (IPFirewall). Escolhemos usar o Altq de forma independente por se mostrar mais simples e eficiente em alterar o campo ToS dos pacotes.

Faça o download da versão 3.1 do Altq, que pode ser encontrada no link:

```
http://www.rofug.ro/projects/freebsd-altq/altq-3.1.tar.gz
```

Com o arquivo compactado do altq em seu computador, podemos continuar com os procedimentos para a instalação. Como iremos recompilar o kernel, recomendamos que se crie uma cópia dos arquivos fontes em uma nova pasta (onde iremos manipular os arquivos) para que se possa preservar o kernerl atual[14]:

```
# cd /usr/src

# mkdir sys-altq

# cd sys

# tar cvf - . | (cd ../sys-altq; tar xf -)
```

Descompacte o arquivo baixado anteriormente nessa nova pasta:

```
# tar –zxvf altq-3.1.tar.gz
```

Aplique o patch:

```
# cd /usr/src/sys-altq

# patch –p0 < altq-3.1/sys-altq/sys-altq-freebsd-4.5.patch
```

Copie os arquivos altq para o diretório sys-altq/altq:

```
# mkdir altq
# cp altq-3.1/sys-altq/altq/* altq/
```

Agora iremos reconfigurar o novo kernel:

```
# cd i386/conf
# ee ALTQ
```

Será aberto o editor de texto com as configurações do novo kernel. Nele devemos descomentar as opções do Altq que iremos utilizar. Um exemplo:

```
#ALTQ

options ALTQ

#options ALTQ_RED

#options ALTQ_RIO
```

```
#options ALTQ_CBQ

#options ALTQ_CDNR

#options ALTQ_PRIQ

#options ALTQ_HFSC
```

Caso você queira usar a função Conditioner do Altq basta descomentar a linha:

```
options ALTQ_CDNR
```

Selecionada as funções que serão usadas pelo Altq, continuemos com a configuração:

```
# config ALTQ

# cd ../../compile/ALTQ

# make depend

# make clean

# make

# make install

# shutdown –r now
```

Após reiniciado o computador, iremos instalar os dispositivos do Altq:

```
# cd altq-3.1

# sh MAKEDEV.altq all
```

Para instalar as ferramentas como altqd e altqstat:

```
# cd altq-3.1

# make

# make install

# shutdown –r now
```

Novamente o computador será reiniciado, e se nenhum erro ocorreu o Altq e suas ferramentas estarão instaladas e prontas para uso.

APÊNDICE II – ARQUIVO ALTQ.CONF DOS ROTEADORES DE BORDA

```
1.interface tl1

2.conditioner tl1 rtp <mark 0xb8>

3.    filter tl1 rtp 3.3.3.2 0 4.4.4.2 (porta rtp) 17

4. conditioner tl1 ssh <mark 0xb8>

4.    filter tl1 ssh 3.3.3.2 0 4.4.4.1 22 0
```

Este arquivo deve estar na pasta /etc. Detalharemos cada linha deste arquivo:

```
1. interface tl1
```

A primeira linha ativa o Altq na interface em questão.

```
2. conditioner tl1 rtp <mark 0xb8>
```

O conditioner é o formatador do tráfego que passa pela interface, tendo várias opções de ações a serem tomadas. Uma das opções é marcar os pacotes, feito através da opção "mark". No nosso projeto, alteramos essa linha de acordo com a prioridade que queremos dar aos pacotes VoIP. Por exemplo, se quisermos dar a prioridade de uma classe AF21 para os pacotes VoIP, bastar remarcá-lo com <mark 0x48>.

```
3.    filter tl1 rtp 3.3.3.2 0 4.4.4.2 (porta rtp) 17
```

A terceira linha é o filtro, onde os pacotes que se encaixarem no perfil dado serão encaminhados para o conditioner para serem marcados. As opções do filter são:

```
(endereço de destino) [máscara de rede] (porta de destino) (endereço
de origem) [máscara de rede] (porta de origem) (protocolo) [valor do
tos [máscara do tos]]
```

Os parâmetros entre colchetes são opcionais. A porta especificada nessa linha é obtida através de um script executado nas máquinas dos usuários. O número que especifica o protocolo usado pelo UDP é o 17 por padrão.

APÊNDICE III – ARQUIVO ALTQ.CONF DOS ROTEADORES DE CORE

```
1. interface fxp0 bandwidth 200k hfsc

2. class hfsc fxp0 default root pshare 10 default

3 .class hfsc fxp0 ef root grate 90k

4.   filter fxp0 ef 0 0 0 0 0 tos 0xb8

5. class hfsc fxp0 af21 root grate 40k pshare 6 rio

6.   filter fxp0 af21 0 0 0 0 0  tos 0x48

7. class hfsc fxp0 af11 root grate 15k pshare 5 rio

8.   filter fxp0 af11 0 0 0 0 0  tos 0x28

9. ########################################

10. interface tl0 bandwidth 200k hfsc

11. class hfsc tl0 default root pshare 10 default

12. class hfsc tl0 ef root grate 90k

13.  filter tl0 ef 0 0 0 0 0 tos 0xb8

14. class hfsc tl0 af21 root grate 40k pshare 6 rio

15.  filter tl0 af21 0 0 0 0 0  tos 0x48

16. class hfsc tl0 af11 root grate 15k pshare 5 rio

17.  filter tl0 af11 0 0 0 0 0  tos 0x28

18. conditioner tl0 ssh <mark 0xb8>

19.  filter tl0 ssh 3.3.3.2 0 4.4.4.1 22 0
```

A primeira linha ativa o Altq na interface fxp0 e limita essa interface a trabalhar com uma taxa de 200 Kbps utilizando a disciplina hfsc (Hierarchical Fair Service Curve). Escolhemos limitar a taxa de dados para 200 Kbps pois seria mais complicado saturar a rede utilizando todos os 100 Mbps que a placa de rede pode oferecer.Assim, podemos congestionar a rede com comandos simples, como por exemplo o comando ping.

Logo a seguir são configuradas as classes DiffServ que irão utilizar essa taxa de 200 Kbps.

Na linha 2 é definida a classe padrão. Somente uma classe padrão deve ser definida para cada interface. Essa classe padrão é usada quando os pacotes não se encaixam em nenhum critério de filtragem das outras classes. No nosso caso a classe padrão irá utilizar no máximo 10% da banda total de 200k (quando houver disponibilidade), ação conseguida através do comando pshare.

Na linha 3 é configurada a classe EF (Expedited Forwarding), onde garantimos 90k de banda (através do comando grate) para todo pacote que utilizar essa classe. Na linha 4 é realizado o filtro dos pacotes

que irão entrar na classe EF. Percebe-se que a filtragem é feita levando em consideração somente o campo ToS dos pacotes, ou seja, somente irá ingressar na classe EF os pacotes que estiverem com o ToS setado em 0xb8.

Seguindo temos a linha 5, que se refere à classe AF21. Parecido com a configuração da classe EF, a classe AF21 garante 40k de banda e se houver disponibilidade da classe default usará mais 6%. A opção rio é o mecanismo de red (random early detection: mecanismo de gerenciamento do buffer) para os pacotes de entrada e saida da fila.

As linhas de 6 à 8 são quase idênticas às linhas 4 e 5, mudando apenas os valores dos parâmetros utilizados, assim como as linhas de 10 à 17, que se diferenciam das linhas de 1à 8 apenas pelo nome da interface.

As duas últimas linhas são unicamente para que se dê prioridade aos pacotes ssh (secure shell), protocolo de rede que permite que se execute comandos em uma unidade remota. Isso porque quando a rede está congestionada temos que garantir que os comandos ssh sejam executados na máquina remota.

APÊNDICE IV – DIFFSERV ADAPTATIVO *SHELL SCRIPT*

```
1.#!/bin/bash

2.clear

3.echo "************************************************************"

4.echo "************************************************************"

5.echo "                  Diffserv Adaptativo                       "

6.echo "************************************************************"

7.echo "************************************************************"

8.echo ""

9.while [[ 1 == 1 ]]

10.do

#########################################################################
#Analisa a porta usada pelo RTP

11.sudo tshark -a duration:5 -i eth0 -f "dst portrange 49152-65535
and dst host 3.3.3.2" > /home/igor/Diffserv_Adaptativo/captura2

12.while read linha2

13.do

14.  pacote=$linha2

15.done < /home/igor/Diffserv_Adaptativo/captura2

16.porta=`echo $pacote | awk '{print $11}'`

17.echo "------------------------------------------------------------"

18.echo "Porta RTP = $porta"

19.echo "------------------------------------------------------------"

20.if [[ $porta -ne "" ]]

21.then

#########################################################################

#Leitura do arquivo gerado pela gravacao do trafego no tshark e
#geracao do arquivo com os parametros
```

```
22.sudo tshark -i eth0 -a duration:25 -w
/home/igor/Diffserv_Adaptativo/captura

23.echo `sudo tshark -d udp.port=="$porta",rtp -r
/home/igor/Diffserv_Adaptativo/captura -qz rtp,streams >
/home/igor/Diffserv_Adaptativo/arquivo`

#####################################################################
#Loop para leitura dos parametros

24.while read linha2

25.do

26.   src=`echo $linha2 | awk '{print $1}'`

27.   if [[ $src == "4.4.4.2" ]]

28.   then

29.         xx=$linha2

30.   fi

31.done < /home/igor/Diffserv_Adaptativo/arquivo

#####################################################################
#Separacao da string com os parametros

32.lost=`echo $xx | awk '{print $11}' | sed 's/(//' | sed 's/)//' |
sed 's/%//'`

33.maxdelta=`echo $xx | awk '{print $12}'`

34.maxjitter=`echo $xx | awk '{print $13}'`

35.meanjitter=`echo $xx | awk '{print $14}'`

#####################################################################
#Avaliacao dos parametros calculados com os limites maximos e
minimos de qualidade estipulados

36.teste=`echo "$maxdelta 160 130" | awk '{if ($1 > $2) print 1;
else {if ($1 < $3) print 2; else print 0}}'`

37.teste2=`echo "$meanjitter 15 10" | awk '{if ($1 > $2) print 1;
else {if ($1 < $3) print 2; else print 0}}'`

38.teste3=`echo "$lost 5 0.5" | awk '{if ($1 > $2) print 1; else {if
($1 < $3) print 2; else print 0}}'`

39.if [[ $teste = 1 ]]

40.then

41.   estadodelta="Ruim"
```

```
42.else
43.   if [[ $teste = 2 ]]
44.   then
45.         estadodelta="Bom"
46.   else
47.         estadodelta="Ok"
48.   fi
49.fi
50.if [[ $teste2 = 1 ]]
51.then
52.   estadojitter="Ruim"
53.else
54.   if [[ $teste2 = 2 ]]
55.   then
56.         estadojitter="Bom"
57.   else
58.         estadojitter="Ok"
59.   fi
60.fi
61.if [[ $teste3 = 1 ]]
62.then
63.   estadolost="Ruim"
64.else
65.   if [[ $teste3 = 2 ]]
66.   then
67.   estadolost="Bom"
68.   else
69.   estadolost="Ok"
70.   fi
```

```
71.fi

#####################################################################
#Escreve na tela os valores observados e sua avaliacao

72.echo "-------------------------------------------------------"

73.echo "AVALIACAO"

74.echo ""

75.echo "   Parametro       Valor    Qualidade"

76.echo "-------------------------------------------------------"

77.echo "   Lost packets    $lost         $estadolost"

78.echo "   Max Delta       $maxdelta     $estadodelta"

79.echo "   Mean Jitter     $meanjitter   $estadojitter"

80.echo "   Max Jitter      $maxjitter"

81.echo "-------------------------------------------------------"

#####################################################################
#Avaliacao da atitude a ser tomada

82.if [[ $estadolost == "Ruim" || $estadodelta == "Ruim" ||
$estadojitter == "Ruim" ]]

83.then

84.  estado="Aumentar Prioridade"

85.elif [[ $estadolost == "Bom" && $estadodelta == "Bom" &&
$estadojitter == "Bom" ]]

86.then

87.  estado="Diminuir Prioridade"

88.else

89.  estado="Nada a fazer"

90.fi

91.echo "ACAO"

92.echo ""

93.echo $estado

94.echo "-------------------------------------------------------"
```

```
#####################################################################
#Decisao de qual script ira executar no roteador

95.if [[ $estado == "Aumentar Prioridade" ]]

96.then

97.   ssh root@4.4.4.1 sh /script+

98.elif [[ $estado == "Diminuir Prioridade" ]]

99.then

100.  ssh root@4.4.4.1 sh /script-

101.fi

102.else

103.echo "Nao ha stream RTP na rede"

104.echo "-----------------------------------------------------------"

105.sleep 10

106fi

107done

108.exit 0
```

O script inicia com uma tela de apresentação. Nas linhas 2 a 8 a tela é limpa e mostra-se o nome do programa.

Seguindo na linha 9 é iniciado um loop infinito com o objetivo de fazer com que o script nunca deixe de executar até ser encerrado pelo usuário.

Na linha 11 o tshark é executado por 5 segundos fazendo uma captura com um filtro para pegar apenas pacotes com porta de destino no intervalo de 49152 a 65535, já que este é o grupo de portas usado pelo SJphone para o tráfego RTP. Esta captura é gravada no arquivo "captura2" que é lido pelo script nas linhas 11 a 15. A linha deste arquivo possui uma estrutura tabelada, e separa os parâmetros dos pacotes por espaços, sendo que a 11ª palavra da linha é a porta de destino do pacote. Na linha 16 passamos a ultima linha do arquivo captura2 já lido para o aplicativo awk, que separa a string pelos espaços e retorna a 11ª palavra (porta de destino). Dessa forma o script consegue identificar a porta usada para o tráfego RTP. A linha 20 observa se a porta do RTP foi encontrada e em caso negativo executará o programa a partir da linha 102.

Na linha 22, o tshark é novamente executado por 25 segundos gravando a captura no arquivo "captura". Após a gravação, o tshark é usado na linha 23 para ler o arquivo recém gerado com o comando para decodificar corretamente a porta RTP já identificada anteriormente e imprimir apenas as estatísticas (jitter médio, jitter máximo, delta máximo, perda de pacotes) dos fluxos de voz que são gravados no arquivo "arquivo".

Nas linhas 24 a 31, o arquivo com as estatísticas é lido e, nas linhas 32 a 35, estas estatísticas são separadas em variáveis com o aplicativo awk.

Devido a uma limitação do *shell script*, que considera variáveis do tipo float como string, nas linhas 36 a 38 passamos os parâmetros para o awk fazer comparação dos valores com os limites estipulados para cada característica do fluxo. Caso seja maior que o limite máximo, o teste retorna o valor 2, caso seja menor que o limite mínimo, o teste retorna o valor 1 e caso esteja entre os valores máximo e mínimo retorna o valor 3.

Com o resultado dos testes sendo dados como inteiros, o *Shell script* consegue fazer as comparações para tomada de decisão, e assim nas linhas 39 a 71 os testes são analisados e é retornado nas variáveis de estado as condições (boa, ok, ruim) da rede, medidas para jitter médio, delta máximo e perda de pacote.

Nas linhas 72 a 81 estão os comandos para escrever na tela as estatísticas observadas do fluxo RTP, assim como o julgamento do estado destas estatísticas.

Nas linhas 82 a 94, é tomada a decisão de qual ação deve ser executada diante das avaliações feitas do tráfego RTP. Se jitter médio, delta máximo e perda de pacotes forem considerados bons, a prioridade do tráfego RTP pode ser diminuída. Se um destes parâmetros for considerado ruim, a prioridade do trafego RTP dever ser aumentada. Em caso de nenhuma das duas situações anteriores acontecerem, nada deve ser feito.

Nas linhas 95 a 101 é tomada a decisão de qual script que deverá rodar no roteador de borda. Caso deva aumentar a prioridade, o script+ é executado através do comando ssh (Secure Shell). Caso deva diminuir a prioridade, o script- é executado. Estes scripts são descritos no Apêndice V.

Nas linhas 102 a 106 são executados os comandos caso não seja encontrada a porta RTP na primeira captura com o tshark. Nestas linhas é escrito na tela que não foi encontrado stream RTP na rede e o programa dorme por 10 segundos.

Na linha 107 o loop termina e então o script volta a ser executado a partir da linha 24.

Para o script funcionar corretamente, algumas configurações tiveram que ser feitas.

O TShark precisa de permissão de root para escutar a interface de rede. Porém quando usado o comando sudo ou su, ele perde a permissão de criar arquivo nas pastas do usuário. Sendo assim, esta permissão teve que ser liberada no sistema.

O ssh por padrão exige uma senha para que um terminal tenha acesso remoto ao outro. Isso dificulta a execução deste comando dentro de um script. Por isso o ssh foi configurado para não haver necessidade de se digitar uma senha. Isso é feito gerando um par de chaves digitas no usuário cliente que juntas fazem uma verificação de autenticidade das informações. Elas podem ser geradas através do seguinte comando .

```
# ssh-keygen -t rsa
```

Após ter gerado as chaves que ficam salvas na pasta /root/.ssh/ , a chave pública deve ser divulgada para o usuário servidor pelo comando.

```
# ssh root@servidor cat /root/id_rsa.pub >>
/root/.ssh/authorized_keys
(digitar senha)
```

Com as chaves geradas e a chave pública divulgada para o usuário servidor, deve ser configurado também no servidor a opção para permitir acesso ao usuário root. Isso é feito alterando uma linha do arquivo /etc/ssh/sshd_config. A linha onde se encontra *PermitRootLogin no* deve ser alterada para *PermitRootLogin yes*. O usuário cliente deve ser configurado para não pedir autenticação com senha. Basta alterar o arquivo /etc/ssh/ssh_config, descomentando a linha *password authentication no*. Desta forma o ssh pode ser feito sem senha. Apesar de não ser seguro, fica muito mais prático o seu uso dentro do script.

APÊNDICE V – SCRIPT+ / SCRIPT-

```
#!/bin/sh

#SCRIPT +

1.a=0

2.while read linha

3.do

4.a=`expr $a + 1`

5.if [ "$a" -eq "4" ]

6.then

7.y=$linha

8.fi

9.done < /etc/altq.conf

10.mark=`echo $y | awk '{print $5}' | sed 's/>//'`

11.echo "Marcacao atual $mark"

12.if [ "$mark" = "0x28" ]

13.then

14.cp /etc/altqaf21.conf /etc/altq.conf

15.echo "Marcacao alterada para 0x48"

16.elif [ "$mark" = "0x48" ]

17.then

18.cp /etc/altqef.conf /etc/altq.conf

19.echo "Marcacao alterada para 0xb8"

20.elif [ "$mark" = "0x00" ]

21.then

22.cp /etc/altqaf11.conf /etc/altq.conf

23.echo "Marcacao alterada para 0x28"

24.else

25.echo "Marcacao nao alterada"
```

```
26.fi
27.killall -9 altqd
28.altqd
29.exit 0
```

```
#!/bin/sh
#SCRIPT –
1.a=0
2.while read linha
3.do
4.a=`expr $a + 1`
5.if [ "$a" -eq "4" ]
6.then
7.y=$linha
8.fi
9.done < /etc/altq.conf
10.mark=`echo $y | awk '{print $5}' | sed 's/>//'`
11.echo "Marcacao atual $mark"
12.if [ "$mark" = "0x28" ]
13.then
14.cp /etc/altqbe.conf /etc/altq.conf
15.echo "Marcacao alterada para 0x00"
16.elif [ "$mark" = "0x48" ]
17.then
18.cp /etc/altqaf11.conf /etc/altq.conf
19.echo "Marcacao alterada para 0x28"
20.elif [ "$mark" = "0xb8" ]
21.then
22.cp /etc/altqaf21.conf /etc/altq.conf
```

```
23.echo "Marcacao alterada para 0x48"

24.else

25.echo "Marcacao nao alterada"

26.fi

27.killall -9 altqd

28.altqd

29.exit 0
```

Os script+ e script- são os scripts que são executados no roteador de borda para alterar o arquivo de configuração do altq e assim mudar as marcações do campo ToS dos pacotes VoIP.

Os dois scripts funcionam da mesma forma. Baseados na marcação atual lida no arquivo altq.conf, escolhem qual arquivo deve substitui-lo.

Na linha 1 é iniciada a variável "a" que serve como um contador. Na linha 2 um loop é iniciado para ler o arquivo altq.conf . Dentro deste loop, na linha 3, o contador "a" é incrementado e na linha seguinte é comparado com o número 4. Isso é feito para gravar na variável "y", a quarta linha do arquivo, que possui a marcação dos pacotes RTP. Pode se observar a gravação da variável "y" na linha 7.

A linha 10 passa a variável "y" para o aplicativo awk que faz a separação da string e retorna a 5ª palavra, referente a marcação usada para o tráfego RTP. Ainda nessa linha, esta palavra é passada para o aplicativo sed que retira o caracter ">" da palavra. A linha 11 escreve na tela a marcação atual lida.

Nas linhas 12 a 26 é feita a comparação com as marcações e tomada as decisões de qual deverá ser a nova configuração do altq. No script+, a proposta é melhorar a classe de serviço do tráfego VoIP. Assim se a marcação atual for 0x00, o script substituirá o arquivo altq.conf pelo arquivo altqaf11.conf que contém as mesmas regras, com exceção da regra de marcação do VoIP que é alterada para 0x28. Se a marcação for 0x28, o script substitui o altq.conf pelo arquivo altqaf21.conf que como o altqaf11.conf altera apenas a regra de marcação do VoIP para 0x48. Da mesma forma acontece quando a marcação atual for 0x48. O altq.conf é substituído pelo altqef.conf que altera a marcação para 0xb8. Se a marcação for 0xb8 o script nada faz, pois não existe classe melhor para substituir.

O princípio usado no script- é semelhante. O objetivo agora é diminuir a classe DiffServ do tráfego VoIP. Então quando a marcação atual for 0xb8, o altq.conf é substituído pelo altqaf21.conf. Caso a marcação seja 0x48, altq.conf é substituído por altqaf11.conf. Se a marcação for 0x28, o altq.conf é substituído por altqbe.conf. Por fim, se a marcação for 0x00 o script não faz nada, pois não existe classe menor que a best effort.

A linha 27 encerra o processo altqd (altq daemon) para que seja reiniciado com a nova regra na linha 28.