Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Avaliação de Mecanismos para a Provisão de Qualidade de Serviço às Aplicações VoIP em Redes de Computadores Utilizando o Simulador NS-2

Tomaz Ferreira de Aguiar Neto

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. André Costa Drummond

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenadora: Prof.ª Dr.ª Maristela Terto de Holanda

Banca examinadora composta por:

Prof. Dr. André Costa Drummond (Orientador) — CIC/UnB
Prof. MsC. João José Costa Gondim — CIC/UnB
Prof. Dr. Jacir Luiz Bordim — CIC/UnB

CIP — Catalogação Internacional na Publicação

Neto, Tomaz Ferreira de Aguiar.

Avaliação de Mecanismos para a Provisão de Qualidade de Serviço às Aplicações VoIP em Redes de Computadores Utilizando o Simulador NS-2 / Tomaz Ferreira de Aguiar Neto. Brasília : UnB, 2014.
215 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2014.

1. DiffServ, 2. IntServ, 3. NS-2, 4. Qualidade de Serviço, 5. VoIP

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Avaliação de Mecanismos para a Provisão de Qualidade de Serviço às Aplicações VoIP em Redes de Computadores Utilizando o Simulador NS-2

Tomaz Ferreira de Aguiar Neto

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. André Costa Drummond (Orientador)
CIC/UnB

Prof. MsC. João José Costa Gondim
CIC/UnB

Prof. Dr. Jacir Luiz Bordim
CIC/UnB

Prof.ª Dr.ª Maristela Terto de Holanda
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 7 de Março de 2014

Dedicatória

Dedico este trabalho a Deus que me dá forças para enfrentar os obstáculos que aparecem por todo o caminho. Dedico também à minha esposa, filhos, pais, amigos e professores.

Agradecimentos

Agradeço a Deus por sempre se fazer presente em minha vida e por ouvir as minhas orações. Agradeço à minha esposa, Emmanuelle Silva Coutinho, pelo brilho sem igual da sua presença, pela compreensão, apoio, conselhos e paciência. Agradeço aos meus filhos, Mateus Coutinho de Aguiar e Letícia Coutinho de Aguiar, que me incentivaram com seus abraços e sorrisos. Agradeço aos meus pais, Manoel Galba Vasconcelos de Aguiar e Lúcia Maria Rodrigues, por incentivarem meus estudos, ensinarem-me a ser integro, perseverante, honesto e acreditar em Deus. Agradeço também aos meus irmãos, em especial à minha irmã Valdeída de Sá Vasconcelos pelo aprendizado adquirido em sua instituição de ensino durante todos os meus anos escolares.

Agradeço aos professores do Departamento de Ciências da Computação da Universidade de Brasília por transmitirem seus conhecimentos com empenho e dedicação. Agradeço ainda ao Professor João José da Costa Gondim que me ajudou a entender redes de computadores de forma prática e objetiva.

Em especial agradeço ao meu orientador, Professor André Costa Drummond, por acreditar no desenvolvimento deste trabalho e na minha capacidade, pela orientação simples e precisa, dedicação, compromisso e apoio necessários à conclusão deste trabalho.

Agradeço a todos os meus chefes e amigos de trabalho, principalmente ao Cel Wolski que se mostrou prestativo ao me liberar do trabalho para que eu pudesse me dedicar à vida acadêmica.

Resumo

As redes de computadores atuais, inclusive a Internet foram projetadas sem prever a necessidade de Qualidade de Serviço (QoS), mas logo viu-se a necessidade de mudança na arquitetura das redes como um todo, pois novas aplicações com requisitos cada vez mais exigentes foram surgindo. Na tentativa de melhorar o desempenho dessas aplicações modelos de QoS para a Internet foram propostos pela IETF (*Internet Engineering Task Force*) como o de Serviços Integrados e o de Serviços Diferenciados. Apesar de ainda hoje a Internet ser incapaz de prover QoS para suas aplicações, o que leva a uma baixa qualidade dos serviços VoIP existentes, em ambientes controlados, como nas redes corporativas, é possível implementar mecanismos que garantam os requisitos de QoS destas aplicações. Esta possibilidade permitiu a implantação de soluções VoIP em ambientes corporativos de forma a substituir completamente as soluções de telefonia tradicionais. Neste contexto, esta monografia apresenta uma avaliação dos mecanismos DiffServ para provisão de QoS para as aplicações VoIP utilizando-se o simulador de redes de computadores NS-2 e o cenário de rede corporativa da empresa *Freedom Network* na qual serão implantados 2000 telefones VoIP.

Palavras-chave: DiffServ, IntServ, NS-2, Qualidade de Serviço, VoIP

Abstract

The current computer networks, including the Internet were designed without the need to provide Quality of Service, but soon it has been seen the need for change the architecture as a whole, as new applications with increasingly demanding requirements have emerged. In an attempt to improve the performance of these applications QoS models have been proposed by the IETF (Internet Engineering Task Force) as the Integrated Services and Differentiated Services. Although the Internet is still unable to provide QoS for its applications, which leads to a low quality of the existing VoIP services in controlled environments, such as in corporate networks, it is possible to implement mechanisms to guarantee the QoS requirements of these applications. This possibility allows the deployment of VoIP solutions in enterprise environments that completely replace traditional telephony solutions. In this context, this monograph presents a review of the mechanisms for DiffServ QoS provisioning for VoIP applications using the network simulator NS-2 considering the network scenario of the company Freedom Network in which 2000 VoIP phones will be deployed.

Keywords: DiffServ, IntServ, NS-2, Quality of Service, VoIP

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Internet foi pensada, inicialmente, para dar suporte ao mundo acadêmico, conectando universidades. Ao passar dos anos, a Internet evoluiu e passou a ser usada comercialmente e hoje alcança cerca de 2 bilhões de usuários no mundo, proporcionando a estes acesso a informações, serviços, entretenimento e, principalmente comunicação. Ao mesmo tempo em que ocorre essa expansão, a Internet vem sofrendo transformações tornando-se uma plataforma robusta e confiável. Para consolidar esses objetivos, ainda existem algumas barreiras a serem vencidas, em particular, Segurança e Qualidade de Serviço.

Qualidade de Serviço (do inglês, *Quality of Service*-QoS) refere-se a capacidade da rede em prestar um melhor serviço, permitindo o transporte de pacotes com requisitos especiais. Quando surgiu o conceito de QoS, foram reservados no cabeçalho do datagrama IP bits para indicar o tipo do serviço usado, mas esses bits indicativos não foram respeitados e entraram em desuso porque os roteadores e *switches* convencionais que operam no serviço de melhor esforço são mais baratos, menos complexos e mais rápidos do que os que implementam os mecanismos de QoS [16].

Com o aumento de usuários também houve um aumento dos serviços prestados na Internet. Um destes novos serviços, chamados de aplicações multimídia, diferem das aplicações elásticas, como acesso ao email, *downloads*, compartilhamento de arquivos, porque "são muito sensíveis ao atraso ponto a ponto e a variação de atraso, mas podem tolerar casuais perdas de informações" [17]. Tendo em vista esse novo cenário, a discussão da implantação de QoS na Internet no final dos anos 90 ganhou força e grupos como o IETF propuseram arquiteturas como a IntServ e a DiffServ.

Com a possível implantação de QoS, a Internet sofrerá transformações em sua infraestrutura e poderá suportar diversas aplicações que são sensíveis a taxa de transferência de dados, perda de pacotes, pacotes corrompidos, latência, variação de atraso ou *jitter*, entrega de pacotes fora de ordem, etc. Como exemplo de aplicações desta natureza pode-se citar: Voz sobre IP, videoconferência e telemedicina [29].

1.1 Problema

Este trabalho considera o cenário de rede da empresa *Freedom Network*. O nome da instituição é fictício, pois não foi autorizada sua divulgação. A rede da empresa *Freedom Network* usa o *Best-Effort* e possui uma Central de Atendimento ao Usuário com 100 telefones VoIP em perfeito funcionamento. A empresa irá expandir sua rede com

a construção de dois prédios onde serão implantados 2000 novos telefones VoIP. Com o aumento de tráfego na rede espera-se que o serviço *Best-Effort* atualmente implementado não consiga atender os requisitos mínimos (atraso, perda de pacotes e *jitter*) de funcionalidade das aplicações VoIP. Neste caso, uma hipótese considerável é que implementando o modelo DiffServ poderá se garantir QoS para as aplicações VoIP na rede em questão. Para isto, considera-se a largura de banda, atraso dos enlaces e as características de tráfego observados na própria rede. Dessa forma, o problema estudado nesta monografia é: Considerando o incremento no volume de telefones VoIP na instituição *Freedom Network* de até 2000 dispositivos, a implementação de mecanismos de controle de QoS tradicionais será capaz de atender aos requisitos mínimos de qualidade exigidos para o funcionamento do sistema VoIP?

1.2 Motivação

A implantação de novas tecnologias e protocolos na infraestrutura das redes de computadores apresenta grandes desafios visto que demandaria mudanças em milhares de equipamentos de rede já instalados. Devido a isso, as soluções propostas podem ser testadas em redes de pequeno porte, mas a grande maioria é avaliada através de ferramentas de simulação, que permitem considerar o ambiente real ao qual a solução se destina.

O simulador de redes NS-2 é uma ferramenta amplamente utilizada no meio acadêmico e industrial que permite a criação de diversos cenários de tráfego em uma rede de computadores, viabilizando e simplificando a avaliação de qualquer solução de provisão de QoS.

Outro fator motivador deste trabalho é o uso da tecnologia VoIP. Com essa tecnologia pode-se reduzir os gastos com ligações telefônicas e com a compra de aparelhos convencionais. Finalmente, a tecnologia de Serviços Diferenciados está implementada em quase todos os roteadores e *switches* do mercado, aumentando as chances da implantação desta tecnologia em cenários reais sem a necessidade de *upgrades* na rede.

1.3 Objetivos

Este trabalho tem como objetivo principal avaliar o desempenho do modelo DiffServ como solução para provisão de QoS para aplicações VoIP na rede corporativa da *Freedom Network* usando o simulador NS-2. Este trabalho tem como objetivos específicos:

- Configurar o NS-2 e implementar os cenários de rede e tráfego de interesse;
- Desenvolver simulação sem tratamento de QoS para os pacotes VoIP no cenário de rede da instituição *Freedom Network* com o NS-2;
- Desenvolver simulação com tratamento de QoS para os pacotes VoIP no cenário de rede da instituição *Freedom Network* usando o módulo DiffServ disponível no simulador NS-2;
- Desenvolver scripts para gerar os resultados a partir do traço de eventos gerado pelo simulador NS-2;

- Mostrar, através de simulações uma melhora significativa do desempenho de aplicações VoIP usando a solução DiffServ.

1.4 Metodologia

O presente trabalho fará uma revisão teórica dos principais conceitos de Qualidade de Serviço abordando seu conceito, parâmetros, mecanismos de escalonamento e regulação usados. Além de apresentar os modelos DiffServ e IntServ mostrando suas características e protocolos usados.

Serão desenvolvidas simulações através do simulador NS-2 usando o modelo *Best-Effort* e o modelo DiffServ em um cenário de rede corporativa submetendo os cenários à vários tipos de tráfego e carga com o intuito de avaliar o comportamento do tráfego VoIP. Para avaliar os parâmetros de QoS serão desenvolvidos scripts que recebem os traços de uma simulação de 1 hora e 30 minutos para gerar as medidas estatísticas de mínimo, médio, máximo e desvio padrão para atraso, perda e *jitter*. Após coletar os resultados das simulações, serão gerados gráficos para comparar a eficiência da solução DiffServ em relação ao modelo *Best-Effort*.

1.5 Estrutura do Trabalho

O Capítulo 2 do trabalho traz a definição de Qualidade de Serviço, os parâmetros necessários para analisar o desempenho de QoS, caracterização dos diversos tipos de aplicações, além dos mecanismos usados nos equipamentos de rede para melhorar o desempenho no tratamento dos pacotes que necessitam de QoS em redes IP. O Capítulo 3 abrange os modelos DiffServ e IntServ mostrando suas limitações, protocolos e definições, além dos problemas relacionados ao uso de cada um desses modelos. Ao fim do capítulo é apresentado um quadro comparativo relacionando os modelos abordados. O Capítulo 4 apresenta os protocolos VoIP, sua arquitetura e funcionamento, além de apresentar suas principais características, mostrando os parâmetros que servem de base para verificar a performance de uma conversação VoIP. O Capítulo 5 mostra como foi elaborada a simulação, a instalação e configuração do NS-2, as principais estruturas utilizadas nas simulações, o cenário de rede da empresa *Freedom Network* e os resultados numéricos da simulação. O Capítulo 6 discuti os resultados do trabalho e apresenta as conclusões, mostrando a importância do uso de técnicas para melhorar o desempenho de aplicações VoIP em redes corporativas.

Capítulo 2

Qualidade de Serviço

2.1 Definição de QoS

Qualidade de Serviço ou QoS pode ser visto como o conjunto de tecnologias que possibilitam à rede oferecer garantias de que os requisitos mínimos de serviço e tráfego podem ser satisfeitos, permitindo melhor uso dos recursos de infraestrutura de rede existente [32, 34]. Do ponto de vista das aplicações multimídia, as quais são sensíveis ao atraso, variação de atraso e ao descarte de pacotes [17], podemos defini-lo como um método de reserva de recursos ou tratamento especial para um fluxo de dados individual ou para um conjunto de fluxos de dados [5]. Em outra definiçao, QoS é o conjunto de características de um sistema necessário para atingir uma determinada funcionalidade. Embora as definições de QoS tenham características particulares, nota-se que todas tem um requisito em comum: "Capacidade de diferenciar entre tráfegos e tipos de serviços, para que o usuário possa tratar uma ou mais classes de tráfego diferentes das demais." [16]. Vale a pena frisar, que a qualidade de áudio e vídeo fornecido pela aplicação pode ser classificada também como um parâmetro variável e subjetivo, quando é determinada pelos seus usuários.

2.2 Parâmetros de QoS

Existem parâmetros numa rede que podem definir o comportamento do tráfego e representar QoS. Esses parâmetros determinam a garantia de serviço que as arquiteturas oferecem à rede. Os principais parâmetros que representam QoS do ponto de vista da rede são [4, 17]:

- Largura de banda: É a capacidade que os enlaces de comunicação tem disponível para a aplicação ao longo do caminho do fluxo de dados.
- Descarte de pacotes: Ocorre, normalmente, devido as filas de roteadores estarem lotadas e os pacotes serem descartados. Isso é uma característica do congestionamento na rede.
- Atraso fim a fim: Refere-se ao tempo que os pacotes gerados pela aplicação levam para chegar ao destino.
- Variaçao de atraso ou *jitter*: O tempo que os pacotes levam para serem gerados, enviados pela fonte e recebidos pelo receptor variam de pacote a pacote. Isso ocorre

devido os pacotes passarem pelas filas dos roteadores e até mesmo seguirem rotas diferentes. Dependendo da quantidade de pacotes existentes na fila o pacote pode seguir seu destino mais rápido ou levar um pouco mais de tempo para ser enviado. Se o receptor ignorar a variação de atraso, corre um sério risco da qualidade da aplicação ficar comprometida. Para eliminar a variação de atraso são utilizados mecanismos chamados de números de sequência, marcas de tempo e atrasos de reprodução, sendo estes definidos abaixo:

- Número de sequência:

  Cada pacote gerado pela fonte é precedido por um número de sequência que é incrementado toda vez que um novo pacote é gerado.
- Marcas de tempo:

  Cada pacote gerado é marcado com o instante em que foi gerado.
- Atrasos de reprodução no receptor:

  Esse atraso é necessário pois os pacotes devem chegar ao receptor antes de seus tempos de reprodução. Se os pacotes não chegarem ao receptor no tempo programado eles serão descartados.

Existem também parâmetros de QoS da perspectiva da aplicação, a saber [19]:

- Vazão: É a capacidade efetiva de largura de banda de uma rede .
- Taxa de perda: Ocorre, em geral, devido ao descarte de pacotes nas filas dos roteadores ao longo do fluxo de dados.
- Latência: Podemos descrevê-lo como a soma dos atrasos da rede com a dos equipamentos utilizados para a comunicação. Ou simplemente como atraso fim a fim que é percebido pela aplicação.
- *Denial of service* ou DoS: O ataque de DoS gera uma interrupção do serviço prestado pela rede. Logo, os requisitos da aplicação não são atendidos.

2.3 Tipos de Aplicações

Um importante recurso que deve ser usado para construir um modelo de QoS é conhecer quais os tipos de tráfego gerados pelas aplicações utilizadas e qual o comportamento da rede para que funcionem corretamente. Com base nos tipos de tráfego, pode-se classificar as aplicações em [3, 17, 23]:

1. Interativas em tempo real ou não elásticas

   Essas aplicações permitem que as pessoas se comuniquem entre si em tempo real e podem se subdividir em:

   Tolerantes: São aquelas que mesmo com variação de atraso, causadas pela rede, ainda são reproduzidas com qualidade aceitável.

Não tolerantes: São aquelas que com variações de atraso são reproduzidas com qualidade inaceitável. Um exemplo desse tipo de aplicação é o telefone por Internet.

2. Elásticas: São aquelas que o importante é a recepção correta dos dados. Assim, elas não necessitam de nenhum recurso em especial. Um exemplo desse tipo de aplicação é o FTP, aplicações Web e outras aplicações cliente/servidor.

Pode-se citar, ainda, duas classes mais abrangentes, especificamente, de aplicações multimídia [17]:

1. Fluxo contínuo armazenado

   São aplicações onde os arquivos que serão exibidos no cliente estão armazenados em um servidor. Essas aplicações podem ser:

   - Mídia Armazenada

     É aquela onde o conteúdo que será distribuído foi pré-gravado e está armazenado num servidor. Pelo fato de estar gravado num servidor, o cliente pode interagir com o conteúdo da gravação.
   - Fluxo contínuo

     É aquela onde o conteúdo a ser reproduzido não precisa ter sido descarregado totalmente no cliente. Depois de começar a receber o arquivo do servidor, o cliente espera alguns segundos e então passa a reproduzi-lo. Essa técnica é também chamada de *streaming*.
   - Reprodução contínua

     Como o conteúdo a ser reproduzido no cliente deve ser igual à temporização original da gravação, os dados devem ser recebidos pelo cliente a tempo de serem reproduzidos sem que ocorram atrasos de *buffer*.

2. Fluxo contínuo ao vivo, que tem as seguintes características:

   - O cliente não pode adiantar o conteúdo que está recebendo.
   - Pode ser feita por transmissões *broadcast* ou *multicast IP*.
   - São iguais as transmissões de TV e rádio, só que são transmitidas pela Internet.

O termo Qualidade de Serviço muitas vezes é utilizado com um sentido mais específico quando oferece serviços com garantias mais estritas relacionadas a certos parâmetros como mostrado em 2.2. Dessa forma, podemos classificar QoS quanto o nível de garantia de serviço oferecido [16]:

- QoS baseada em prioridades: Ocorre quando se oferece garantias para cada tipo de fluxo individualmente.
- QoS baseada em reserva de recursos: Nesse, caso as garantias que são oferecidas são para classes ou grupos que tem a mesma caracterítica. Esse tipo de QoS é mais fácil de ser implementada e barato.

2.4 Mecanismos de Escalonamento e Policiamento de Filas

Para que ocorra a implementação de QoS em redes de computadores é necessário a existência de mecanismos de controle de admissão, política de filas, condicionamento de tráfego e reserva de recursos [6, 16].

O condicionamento do tráfego está relacionado com as garantias que o contrato de serviço pode garantir. Para verificar essas garantias, o tráfego entre o cliente e o provedor de serviço é policiado. Assim, o condicionamento do tráfego está relacionado com classificação de pacotes, medição de tráfego e a ação a ser tomada se o pacote não estiver dentro do perfil especificado no contrato.

2.4.1 Mecanismos de Escalonamento de Filas

Nos roteadores, os pacotes pertencentes a fluxos diferentes são multiplexados e enfileirados nos *buffers* de saída. Para que os pacotes que estão nos *buffers* sejam enviados existe a disciplina de escalonamento do enlace, que faz a seleção dos pacotes que serão transmitidos pelo enlace.

O gerenciamento das filas é um mecanismo fundamental para prover QoS, além de ser necessário para diferenciar os níveis de serviço. A dificuldade está em mensurar o tamanho da fila, pois se a fila for pequena pode ocorrer descarte de pacotes, por outro lado se a fila for grande poderá ser introduzida uma quantidade inaceitável de latência e *jitter*, fazendo com que os protocolos de transporte fim-a-fim e aplicações multimídia não funcionem corretamente. Para que se possa melhorar o repasse de pacotes são usados mecanismos de enfileiramento. A seguir, serão mostrados alguns mecanismos de enfileiramento e suas principais características [11].

First In Fisrt Out - FIFO

Um dos mecanismos de enfileiramento de pacotes é chamado de FIFO - *First In First out* - primeiro a chegar/primeiro a sair. Na Figura 2.1 é mostrado o funcionamento desse mecanismo. Os primeiros pacotes que chegam são transmitidos primeiro, ou seja, os pacotes são transmitidos na ordem que chegam à fila de saída do enlace. Quando os pacotes chegam à fila de saída de um enlace, pode ocorrer os seguintes eventos [17]: Encontrar o enlace ocupado com a transmissão de outro pacote, então aquele espera o pacote que está na fila ser transmitido, se houver espaço no *buffer*; não havendo espaço no *buffer* para o pacote que está chegando, dependendo da política de descarte de pacotes da fila, pode ser que o pacote que está chegando seja descartado ou outro pacote ser retirado da fila para dar espaço ao novo pacote. A política FIFO é usada na ausência de um algoritmo específico e por esse motivo, é normalmente ultilizada em redes de melhor esforço.

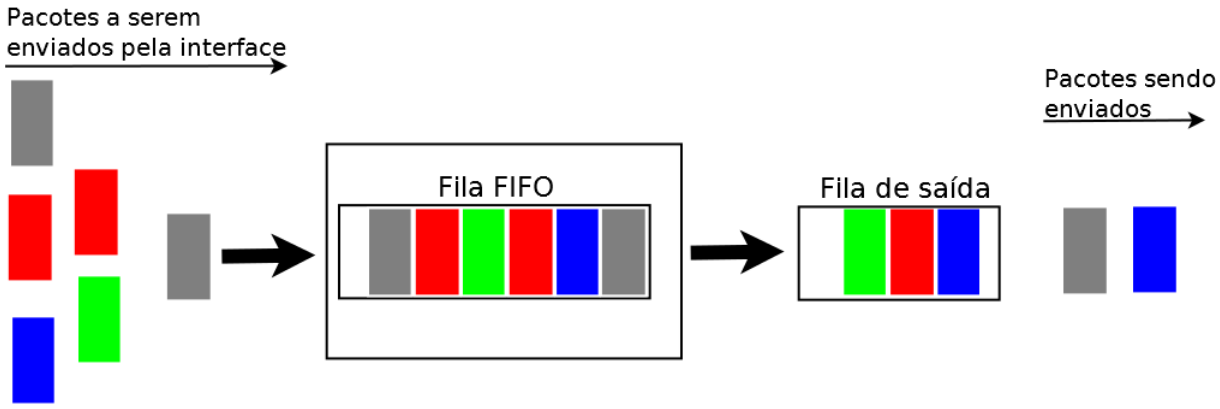

Figura 2.1: Processamento de pacotes através do mecanismo *Fist In First Out*.

Round Robin - RR

No enfileiramento *Round Robin* a classificação dos pacotes é igual a do Enfileiramento Prioritário, a diferença é que neste mecanismo há uma alternância na trasmissão dos pacotes. Assim, se o pacote de uma classe que acabou de ser transmitido, esta classe só terá outro pacote transmitido quando um pacote de cada uma das outras classes forem transmitidos. Por esse motivo esse método também é chamado de Varredura Cíclica. Nesse mecanismo, existe uma disciplina de varredura cíclica de conservação de trabalho que verifica a existência de pacotes de uma determinada classe, se não encontrar nenhum ele imediatamente verifica qual a próxima classe a trasmitir pacotes e verifica se existe algum pacote nesta para ser transmitido, se houver o pacote é transmitido, caso contrário, faz outra busca até encontrar uma classe com pacote a ser trasmitido [21].

Enfileiramento Prioritário

O mecanismo de Enfileiramento Prioritário é aquele no qual os pacotes tem algum tipo de marca no seu cabeçalho e não requer nenhum mecanismo especial de alocação de banda. Assim, ao adicionarmos uma nova fila, é criada uma nova classe de tráfego e o agendador distribui a largura de banda entre todas as filas automaticamente.

A marca que é considerada no cabeçalho é chamada de Campo de Serviços Diferenciados, do inglês *Differentiated Services* (DS).

Cada classe de prioridade é definida pelo campo DS e normalmente tem sua própria fila. Desta forma, o primeiro pacote a ser transmitido é aquele cuja fila não está vazia e tem a mais alta prioridade. Assim, os pacotes de prioridade mais baixa só serão enviados após todos os pacotes das filas com maior prioridade serem enviados. Por essa razão, este algoritmo é mais indicado quando o tráfego com maior prioridade é pouco e deve ser tratado o mais rápido possível.

Para os pacotes dentro da mesma fila pode ser usado o mecanismo FIFO.

Os problemas desse algoritmo são [21]:

1. Se as filas com alta prioridade sempre tem pacotes para serem enviados, as filas com prioridade mais baixa não conseguem enviar pacotes logo, observa-se que o

mecanismo não distribui a largura de banda de acordo com os requisitos de largura de banda de cada classe;

2. O outro problema está no tamanho do pacote, pois se uma fila tem pacotes maiores que o das outras, ela receberá um percentual maior da largura de banda.

Weighted Round Robin (WRR)

O *Weighted Round Robin* (WRR) é usado para evitar a negação completa dos recursos para qualquer classe de serviço. Para isso, divide a largura de banda do enlace de saída entre as classes de tráfego de entrada observando as necessidades de cada classe.

Na Figura 2.2 é mostrado o *Weighted Round Robin*. Este mecanismo é uma variante do *Round Robin*, embora possam ser definidas várias filas de saída, a prioridade de cada fila e a quantidade de tráfego, em bytes, que será transmitido por cada fila a cada passagem na rotação do serviço. Dessa forma, os pacotes são processados até o limite que cada fila pode transmitir ou até que a fila fique vazia. Dessa forma, os pacotes que foram categorizados e classificados para uma fila específica têm uma boa chance de serem transmitidos sem induzir uma latência significativa no sistema evitando o enchimento do *buffer* e consequentemente o descarte de pacotes. Contudo, o WRR é um mecanismo de prioridade de enfileiramento mais justo, pois há um aumento dos recursos para as filas de maior precedência e uma diminuição de recursos para filas com menor precedência [6, 11].

O argumento, é que negar recursos é pior do que a redução destes, pois quando há a negação de recursos, também não se pode sinalizá-lo. Dessa forma havendo uma redução de recursos, esta pode ser percebida pelos sistemas finais os quais poderão adaptar suas taxas de transmissão em conformidade com o estado atual da rede, diminuindo ou até mesmo evitando o congestionamento na rede.

O problema do WRR é que ele não é escalável, por causa da sobrecarga computacional e pelo impacto causado pela reordenação no encaminhamento dos pacotes em redes com links de alta velocidade.

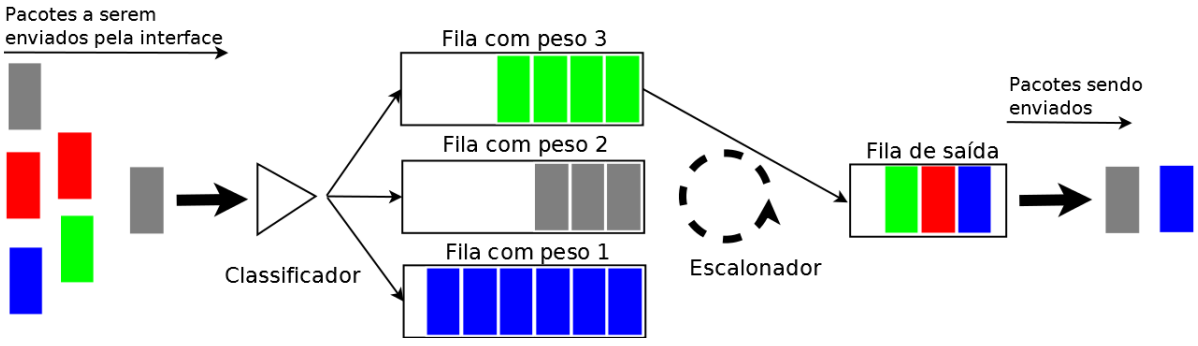

Figura 2.2: Processamento de pacotes através do mecanismo *Weighted Round Robin*.

Weighted Fair Queuing - WFQ

O mecanismo de Enfileiramento Justo Ponderado é uma generalização do *Round Robin* muito utilizado na arquitetura de QoS, tendo um papel fundamental nessas arquiteturas [17]. Como deriva da *Round Robin* possui atendimento às classes de modo cíclico e também a disciplina de varredura cíclica de conservação de trabalho. A diferença é que,

a qualquer momento, o WFQ atribui um peso a cada classe i. Com isso, é garantido a uma determinada classe i que em qualquer intervalo de tempo ela receberá uma fração da largura de banda do enlace, mesmo no pior caso. Com isso, é reservada uma largura de banda mínima para a classe i dando assim uma vazão mínima aos pacotes dessa classe.

2.4.2 Mecanismos de Policiamento

Nos algoritmos anteriores qualquer tráfego tem o mesmo tratamento, mas em QoS exitem tráfegos que precisam ser policiados de acordo com o tempo. Assim, podemos classificar o tráfego de acordo com três critérios [17, 31]:

1. Taxa Média: A rede pode desejar limitar a quantidade de tráfego que será enviado por um fluxo para dentro da rede durante um período de tempo longo. Para calcular a taxa média de uma fonte é necessário especificar o intervalo de tempo em que esta será calculada. De posse do intervalo de tempo, pode-se calcular a taxa média dividindo-se o número de bits gerados na fonte pela duração do intervalo.
2. Taxa de Pico: É o maior valor médio medido em todos os intervalos com a duração especificada, ou seja, é o limite máximo para a taxa média.
3. Taxa de Rajada: A rede pode limitar a quantidade máxima de pacotes que podem ser enviados para dentro da rede em um intervalo de tempo extremamente curto.

Algoritmo do Balde Furado

O algoritmo do Balde Furado busca policiar a taxa de entrada de pacotes na rede através do controle da frequência com que os pacotes são encaminhados. Essa técnica funciona da seguinte maneira: Cada *host* está conectado a rede por uma porta do enlace que implementa o algoritmo do Balde Furado. Se chegar um pacote e a fila de saída do enlace estiver cheia o pacote será descartado, caso contrário o pacote entrará na fila e irá esperar sua vez para ser transmitido. Esse algoritmo garante que a vazão dos pacotes será uniforme, atenuando as flutuações, além de reduzir a possibilidade de congestionamento na rede. Esse algoritmo tem um bom desempenho quando todos os pacotes tem o mesmo tamanho; se os pacotes tem tamanhos diferentes, a melhor opção é permitir a passagem de um número fixo de bytes por unidade de tempo [31].

Time Sliding Window with 2 Color Marking - (TSW2CM)

Este policiador utiliza as informações das taxas recebidas (CIR) e duas precedências de descarte. A menor precedência de descarte é utilizada quando a CIR é excedida.

Algoritmo do Balde de Fichas

Em alguns casos é necessário um aumento na vazão dos pacotes. Como o algoritmo do Balde Furado tem uma vazão constante, para contornar esse problema temos o algoritmo do Balde de Fichas. Na Figura 2.3 é mostrado o funcionamento do Balde de Fichas. O algoritmo do Balde de Fichas guarda no balde fichas que são geradas a cada pulso de *clock*. Ao chegar um pacote na fila de saída do enlace é verificado se contém alguma ficha

dentro do balde, se houver, o pacote é enviado e a ficha é descartada. Com isso se o tamanho do balde é n e chega uma rajada de pacotes repentina de tamanho n o algoritmo do Balde de Fichas consegue dar vazão a esses pacotes simultaneamente, mas por tempo determinado. Um outro fator importante desse algoritmo é que não ocorre o descarte de pacotes, mas sim de fichas quando o balde excede a sua capacidade máxima.

Os algoritmos de Balde Furado e Balde de Fichas podem ser usados em *hosts*, para controlar a saída, e para atenuar o tráfego entre roteadores [31]. Essas ferramentas servem para modelar o tráfego da rede de forma mais gerenciável para que se possa atender aos requisitos de QoS.

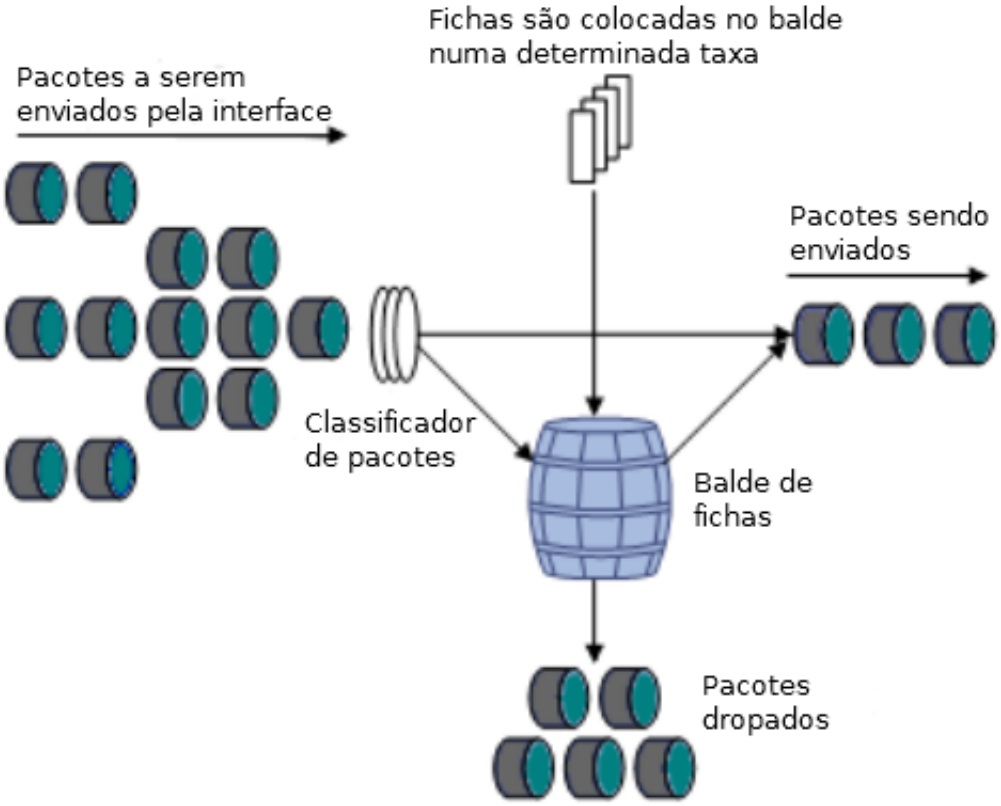

Figura 2.3: Modelagem de tráfego usando o policiador *Token Bucket*. Figura original em [6].

Combinando os algoritmos Balde de Fichas com WFQ

Uma outra forma de uso do algoritmo do Balde de Fichas é com o algoritmo do Enfileiramento Justo Ponderado (WFQ). Se o balde de fichas estiver cheio, ou seja com n fichas, e chegar ao enlace uma rajada com n pacotes, os pacotes retiram todas as fichas do balde e passarão para a fila WFQ onde serão separados por classe de tráfego e posteriormente serão transmitidos pelo enlace. Com a combinação desses algoritmos podemos ter um maior controle sobre os pacotes que deverão ter maior prioridade de transmissão [17].

2.5 Controle de Congestionamento

As técnicas de controle de congetionamento monitoram a rede com o intuito de antecipar e evitar a ocorrência de congestionamento. Duas técnicas que operam dessa maneira são a *Random Early Detection* (RED) e a *Weighted Random Early Detection* (WRED).

2.5.1 Random Early Detection

Quando um pacote TCP é descartado pela rede dá-se início ao processo conhecido como partida lenta. Se houve algum congestionamento na rede devido a essa perda, o descarte irá aliviar o congestionamento. O problema que ocorre é quando várias fontes tem seus pacotes descartados e dão início ao processo de partida lenta. Isso configura um cenário em que todas as fontes irão aumentar sua janela de congestionamento ao mesmo tempo e ocorrerá novamente o descarte de pacotes das mesmas fontes. Para evitar esse problema, o algoritmo RED atua de forma preventiva, tentando evitar que o congestionamento ocorra [36].

Quando o algoritmo RED nota que uma possível situação de congestionamento vai ocorrer, inicia-se um processo de descarte de pacotes aleatório, onde a probabilidade de descarte é função da taxa de ocupação da fila. Para funcionar adequadamente o RED necessita de protocolos que sejam robustos quanto à perda de pacotes, caso contrário o RED não terá efeito positivo.

2.5.2 Weighted Random Early Detection

No algoritmo WRED, a probabilidade de descarte de um pacote depende da taxa de ocupação da fila e por um peso associado ao fluxo ao qual o pacote pertence. O objetivo do WRED é priorizar os pacotes de fluxo com menor probabilidade de descarte ou que fizeram reserva de recursos [36].

2.6 Considerações Finais

Qualidade de Serviço é o conjunto de tecnologias que possibilitam à rede oferecer garantias de que requisitos mínimos de serviço e tráfego podem ser satisfeitos [32, 34]. Para prover QoS é necessário: Classificar, Controlar e Policiar o tráfego e Gerenciar e Evitar o congestionamento nas filas.

Para prover QoS na rede da instituição *Freedom Network* serão utilizados policiadores *Token Bucket*, filas *Weight Round Robin* (WRR) para gerenciar o tráfego e o mecanismo *Random Early Detection* (RED) para evitar congestionamento nas filas. O policiador *Token Bucket* será usado, pois consegue dar vazão aos pacotes de um tráfego em rajada onde o tamanho máximo da rajada é determinado pelo tamanho do balde, sendo uma resposta rápida ao envio de pacotes. Se for usado o policiador *Leaky Bucket* seria necessário especificar a taxa máxima que o tráfego poderia alcançar para dar vazão a um tráfego em rajada, pois o *Leaky Bucket* faz com que o tráfego de saída se mantenha em uma taxa constante. Logo, pode-se notar que o *Token Bucket* utiliza de forma mais eficiente os recursos da rede. O mecanismo de enfileiramento *Weight Round Robin* é uma variante do mecanismo de enfileiramento *Round Robin* onde cada fila possui um peso.

Quanto maior o peso da fila maior é a sua prioridade e consequentemente maior será a utilização da largura de banda por essa fila. A utilização do *Weight Round Robin* deve-se a sua capacidade de prover diferentes mecanismos de transmissão de pacotes podendo assegurar aos pacotes VoIP o Encaminhamento Expresso, discutido na Seção 3.2.3, além de fornecer o serviço de melhor esforço para os pacotes com baixa prioridade. O mecanismo *Random Early Detection* será usado devido ao fato de conseguir detectar quando uma possível situação de congestionamento irá ocorrer. Para tal finalidade são definidos dois parâmetros: Limites mínimo e máximo da fila. Caso a quantidade de pacotes atinja o valor médio de pacotes suportados na fila, pacotes são marcados aleatoriamente de acordo com uma função probabilística. Se a quantidade de pacotes ultrapassar o valor médio, os pacotes marcados anteriormente serão descartados primeiros. Caso a fila atinja seu limite máximo, pacotes são descartados indiscriminadamente.

Capítulo 3

Serviços Integrados e Serviços Diferenciados

Como a Internet oferece o modelo de melhor esforço e a cada dia estão sendo desenvolvidas aplicações multimídia este modelo se torna inadequado para ser utilizado com esse tipo de aplicação. Com a finalidade de oferecer algum tipo de garantia para o tráfego dessas aplicações foi criado pela IETF um grupo para desenvolver uma rede de Serviços Integrados (IntServ). Ao passo que o modelo IntServ foi apresentando falhas, foi proposta uma nova arquitetura para oferecer QoS na Internet, o qual foi chamado de Serviços Diferenciados (DiffServ).

3.1 Serviços Integrados (IntServ)

O modelo Serviços Integrados (*Integrate Services*) foi definido por um grupo da *Internet Engineering Task Force* (IETF) e divide o tráfego da Internet no tráfego de melhor esforço e no tráfego do fluxo de dados de aplicativos que necessitam de reserva de largura de banda com QoS garantida, além de poder ser usado em *unicast* e *multicast* [3]. Para que os roteadores da Internet sejam capazes de suportar o modelo dos Serviços Integrados devem propiciar uma QoS apropriada para cada fluxo de dados. A essa função do roteador dá-se o nome de Controle de Tráfego e consiste dos seguintes elementos [3, 22]:

Programador de Pacotes

: Gerencia e fiscaliza o direcionamento de fluxo de pacotes diferentes em *hosts* e roteadores de acordo com suas classes de serviço através de um conjunto de filas [3], além de policiar e moldar o tráfego. Deve garantir que os pacotes sejam entregues em conformidade com os parâmetros de QoS de cada fluxo e também que um *host* não viole suas características de tráfego. Normalmente é implementado no *drive* de saída e corresponde ao protocolo de camada de enlace.

Classificador de Pacotes

: Identifica e controla os pacotes em um *host* ou roteador que deverá receber um nível de serviço específico. Assim, cada pacote de entrada é mapeado pelo classificador em uma classe específica e os pacotes que são classificados numa mesma classe recebem

o mesmo tratamento por parte do Programador de Pacotes. A escolha da classe de um fluxo de dados no modelo de Serviços Integrados baseia-se no [21]:

- Identificação do protocolo
- Endereço IP de origem
- Endereço IP de destino
- Porta de origem
- Porta de destino

Controle de Admissão

É usado pelo roteador para determinar se existem recursos de roteamento suficientes para aceitar o novo fluxo de dados com a QoS solicitada e verifica se um determinado *host* pode fazer uso da reserva requisitada. Assim, ao chegar um novo fluxo no roteador este pode aceitar o novo fluxo, se houver recursos de roteamento livres e suficientes, ou pode rejeitar se não houver. Se for aceito, o roteador solicita ao Classificador de Pacotes e ao Programador de Pacotes para reservarem a QoS para esse fluxo. Cada roteador ao longo do caminho deve chamar o Controle de Admissão para verificar se pode ou não aceitar o novo fluxo [13, 22].

3.1.1 Classes de Serviço

O grupo de trabalho da IETF definiu também Classes de Serviço que propiciam limites nos controles de QoS dependendo do aplicativo usado. Para cada fluxo é usado um Descritor de Fluxo que define as características de tráfego e de QoS para um determinado fluxo. O Descritor de Fluxo consiste de [22, 27]:

Especificação de Filtro

É usado no Classificador de Pacotes. Usa o endereço IP e a porta de origem do emissor para identificar os pacotes que pertencem a um determinado fluxo.

Especificação de Fluxo

Contém as informações de chamada que são usados no Programador de Pacotes e podem ser divididas em:

- Especificação de Tráfego (Tspec): Descreve as características do tráfego que o remetente espera gerar assim como os parâmetros de QoS para a qual a reserva deve ser aplicada. Na especificação do modelo de Serviços Integrados é usado o algoritmo do Balde de Fichas (mostrado na Seção 2.4). Assim, os parâmetros que devem ser especificados são: a que taxa as fichas serão colocadas no balde $\mathbf{r}$, a capacidade do balde $\mathbf{b}$, a unidade policiada mínima $\mathbf{m}$ e o tamanho máximo do pacote M. Isto posto, o tráfego que é policiado pelo Balde de Fichas em todo instante $\mathbf{T}$, não deve ser superior a $\mathbf{rT+b}$.
- Especificação de Requisição (Rspec): Depende do tipo de serviço e das necessidades do aplicativo. Serve para definir QoS para um fluxo específico. Pode especificar os seguintes parâmetros: largura de banda específica, *jitter* máximo, taxa de perda de pacotes.

O Serviço Garantido e o Serviço de Carga Controlada são importantes classes de serviço e fazem parte do modelo dos Serviços Integrados. Abaixo será apresentada uma breve definição de ambos.

Serviço Garantido

O Serviço Garantido faz com que os pacotes de um fluxo que estão de acordo com as Especificações de Tráfego cheguem ao destino até o tempo de retardo máximo. Dessa forma as aplicações têm mais controle sobre o retardo de seus pacotes. É importante ressaltar que o Serviço Garantido não minimiza o *jitter*, ele controla o retardo de enfileiramento máximo [26, 32].

"No modelo de Serviço Garantido a Especificação de Tráfego contém os valores do algoritmo do Balde de Fichas, já a Especificação de Requisição contém a largura de banda da reserva de fluxo." [22].

Serviço de Carga Controlada

O Serviço de Carga Controlada garante que mesmo quando a Internet, com seu serviço de melhor esforço, estiver sofrendo sobrecarga os aplicativos que fazem uso dessa classe de serviço recebem o serviço de melhor esforço equivalente ao da Internet em condições de sobrecarga leve [22, 32].

Para que o Serviço de Carga Controlada funcione, os roteadores que aceitam essa classe de serviço devem garantir que a largura de banda adequada e os recursos de processamento de pacotes estarão disponíveis para processar as solicitações de reserva de QoS [35]. No caso de ocorrer rajadas de tráfego por um longo prazo, o sistema deve ter, no mínimo, uma reserva de largura de banda para garantir que o fluxo chegue ao destino até o tempo de retardo máximo especificado no Descritor de Fluxo.

A classe de Serviço de Carga Controlada não aceita ou não faz uso de reservas de valores específicos de parâmetros de controle [35]. Sendo assim, os aplicativos que usam essa classe de serviço são aqueles que podem tolerar uma pequena perda e retardo de pacotes, como os aplicativos de áudio e vídeoconferência.

Para fazer a reserva de recursos, a aplicação de origem faz uma Especificação do Fluxo, que contém as características do tráfego que incluem: Taxa de pico, taxa média, tamanho da rajada e os parâmetros do Balde de Fichas. Já os requisitos do serviço para o fluxo incluem: Atraso, variação de atraso, largura de banda, taxa de perda de pacotes. O IntServ faz a reserva de recurso através do protocolo chamado Protocolo de Reserva de Recursos (RSVP).

3.1.2 Protocolo RSVP

O protocolo RSVP está especificado na RFC 2205. É suportado no IPv4 e no IPv6, assim como é aplicável nos modos IP *unicast* e *multicast*. É um protocolo que serve para preparar e controlar as requisições de reserva no modelo IntServ. O RSVP necessita de um protocolo de roteamento para operar e é executado sobre os protocolos IP e UDP. Além disso, deve ser implementado em todos os roteadores ao longo do caminho. Os *hosts* terminais e roteadores ao longo do caminho se comunicam via RSVP e através deste criam e mantém estados de fluxos específicos [22]. Para fazer a reserva de recursos os

hosts de origem e destino trocam mensagens para estabelecer a classificação do pacote e o estado de encaminhamento de cada nó. A origem inicia a requisição da reserva de recursos, mas a disponibilidade e reserva de recursos começa no receptor [21]. O estado da reserva de recursos não é permanente, ela é mudado periodicamente. Tanto as mensagens RSVP como os pacotes percorrem o mesmo caminho e quem determina esse caminho são as tabelas de roteamento dos roteadores IP.

Como principais características do RSVP, pode-se citar [33, 37]:

- Faz as reservas de QoS somente em uma direção, por isso é chamado de protocolo simplex. Em conexões duplex, faz-se necessário o uso de duas sessões RSVP para cada *host*;
- É iniciado pelo receptor, o qual também mantém a reserva para o fluxo. Para iniciar a reserva de caminho, o emissor envia ao receptor uma mensagem de sinalização RSVP contendo a especificação de QoS que ele deseja reservar. De posse da mensagem, o receptor analisa e responde enviando uma mensagem RSVP de volta ao emissor com a QoS que deve ser reservada para o fluxo. A QoS especificada pelo emissor e pelo receptor leva em conta a capacidade de seus sistemas;
- Não considera a informação que está no Descritor de Fluxo, sendo estas repassadas ao Controle de Tráfego de cada roteador para o devido processamento;
- Fornecer reserva de recursos para fluxos *unicast* e *multicast*, fazendo as adaptações e as mudanças dos membros do grupo de rota automaticamente;
- Suporta tanto IPv4 quanto IPv6;
- Pode operar de forma transparente em roteadores que não oferecem suporte ao RSVP;
- Cada estado de reserva tem um tempo associado a ele. Quando o tempo expira, os recursos usados nesse estado são liberados para que outras reservas sejam feitas;
- Fornece vários estilos de reserva.

Cabeçalho RSVP

Uma mensagem RSVP consiste de um cabeçalho Comum, seguido por um corpo composto por objetos. O conteúdo e o número de objetos varia com o tipo da mensagem. A Figura 3.1 apresenta os campos do cabeçalho Comum do RSVP.

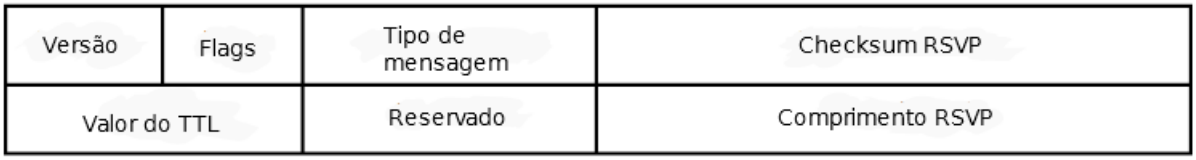

Figura 3.1: Cabeçalho Comum do RSVP. Figura original em [37].

A seguir serão descritos os campos do cabeçalho Comum [22, 37]:

- Versão: Campo de 4 bits. Contém o número da versão do protocolo.
- *Flags*: Campo de 4 bits de comprimento e é reservado para as *flags*. Nenhuma *flags* foi definida.
- Tipo de Mensagem: Campo de 4 bits e representa o tipo da mensagem, que pode ser:

  1. *Path* (Caminho)
  2. Resv (Requisição de Reserva de Recursos)
  3. PathErr (Erro no Caminho)
  4. ResvErr (Erro na Requisição de Reserva de Recursos)
  5. PathTear (Usada na Remoção de um Caminho)
  6. ResvTear (Usada na Remoção de Reserva de Recursos)
  7. ResvConf (Confirmação de Reserva de Recursos)

- *Checksum* RSVP: Campo de 16 bits. Pode ser usado pelo receptor de uma mensagem para detecção de erros na transmissão desta.
- Valor TTL: Campo de 8 bits que contém o valor do TTL IP da mensagem que foi enviada.
- Comprimento RSVP: Campo de 16 bits que contém o comprimento total (em bytes) da mensagem RSVP incluindo o cabeçalho Comum.

Problemas no uso do IntServ

- É um protocolo complicado, pois cada nó cuida do estado da reserva de recursos;
- A quantidade de informação de estado aumenta com o número de fluxos exigindo enorme espaço de armazenamento e gerando sobrecarga de processamento nos roteadores;
- Todos os roteadores, transmissores e receptores, devem implementar o protocolo RSVP, Controle de Admissão, Classificação e Escalonamento de Pacotes.

3.2 Serviços Diferenciados (DiffServ)

O objetivo dos Serviços Diferenciados é definir um pequeno conjunto de mecanismos que possam ser implementados nos nós da rede e que suportem uma grande variedade de serviços e aplicativos.

Para o DiffServ o tráfego de fluxos individuais são agrupados em classes, baseado no comportamento do fluxo. Diferente do IntServ, nessa abordagem os recursos da rede são alocados para a classe de fluxos ao invés de para cada fluxo individualmente e não mais existe a necessidade de estados por fluxo e nem sinalização por salto [2, 22]. Logo, a reserva de recursos é feita para as agregações de fluxo ou *Behavior Aggregate*. A preocupação do DiffServ está no domínio único e não mais no caminho fim-a-fim que os pacotes fazem.

Como o DiffServ provê um tratamento para diferentes tipos de classes de tráfego seu nível de QoS não é absoluto, faz-se necessário usar um Controle de Admissão de pacotes para controlar o tráfego de entrada na rede, o qual é feito nas bordas dos domínios únicos [21].

O serviço DiffServ é definido através de um Acordo de Nível de Serviço (SLA) entre o cliente e o provedor de serviços. Nele são especificados o serviço de encaminhamento e a classificação de tráfego que o cliente deve receber, além de determinar as garantias mínimas de QoS à aplicação cliente. Dessa forma, quando os pacotes passam de um domínio DiffServ para o outro eles são verificados nos roteadores de borda a fim de fiscalizar se o SLA está sendo mantido.

Para que a arquitetura Diffserv funcione corretamente, existem alguns componentes que são necessários: *Per-Hop Behavior* (PHB), Classificação de Pacotes, Acordo de Condicionamento de Tráfego (TCA) incluindo o Perfil do Tráfego, as Métricas de Desempenho, Marcação Adicional, Policiamento e Modelagem do Tráfego. Para que essa arquitetura seja escalável, a implementação das funções de classificação e condicionamento é feita somente nos nós de fronteira.

Na Figura 3.2 é mostrado o processamento dos pacotes do usuário em um roteador que suporta DiffServ. Ao chegar na entrada da rede provedora é verificado se existe algum SLA para os pacotes desse usuário. Dependendo do SLA contratado os pacotes são classificados pelo Classificar de Fluxos Agregados (BA) em diferentes filas de PHB de acordo com o DSCP. Por fim, o Escalonador de Pacotes seleciona uma fila de PHB, respeitando a prioridade de cada fila, que irá transmitir os pacotes pela porta de saída.

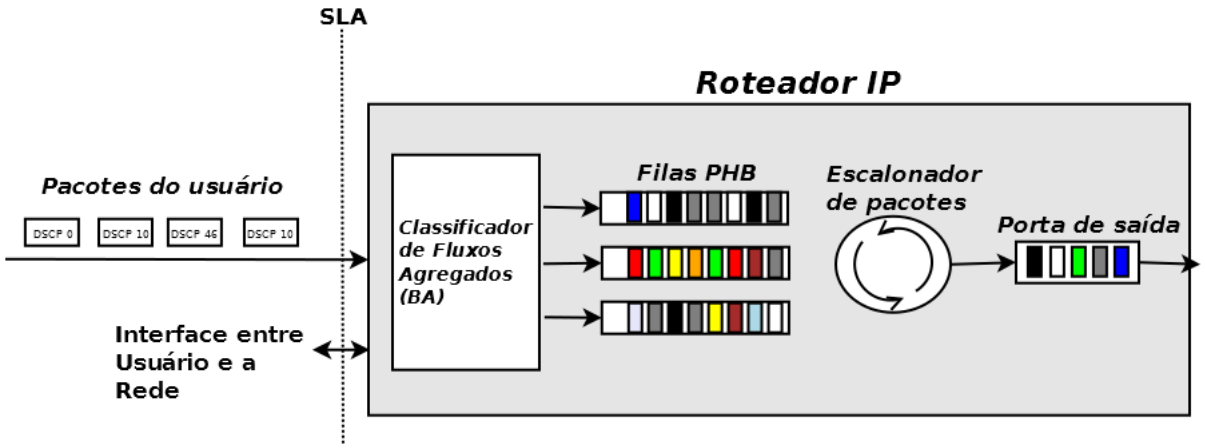

Figura 3.2: Processamento de pacotes em uma arquitetura DiffServ. Figura original em [21].

3.2.1 *Differentiated Services Code Point* (DSCP)

Quando um nó suporta DiffServ ele usa o campo DS do IPV4, mostrado na Figura 3.3, para fazer a marcação do pacote DiffServ. Dos oito campos do DS, os 6 primeiros bits são utilizados para marcação dos pacotes DiffServ e são chamados de *Differentiated Code Point* (DSCP), sendo os outros dois bits reservados para uso futuro. Com o propósito de atribuir e gerenciar os *Code Points* a RFC 2474 dividiu o DSCP em 3 grupos. O grupo 1 tem um conjunto de 32 *Code Points* sendo o último bit do DSCP igual a '0' e os outros

cinco bits podendo ser zero ou um. Esse grupo será padronizado pela IETF e reconhecido universalmente; o grupo 2 tem um conjunto de 16 *Code Points*, sendo os dois últimos bits fixados em '11' e os outros quatro bits podendo ser zero ou um. Esse grupo serve para uso experimental ou local. Não serão reconhecidos fora de uma Intranet; o grupo 3 tem um conjunto de 16 *Code Points*, sendo os dois últimos bits fixados em '01' e os outros quatro bits podendo ser zero ou um. Esse grupo também serve para uso experimental ou local. A diferença deste grupo para o grupo 2 é que este poderá ser usado para ações padronizadas.

Apenas os três primeiros bits do DSCP são usados efetivamente nas redes convencionais e são chamados de Pontos de Código Seletores de Classe, possibilitando até oito classes de DiffServ. Essas classes são reconhecidas no roteadores convencionais que executam IPv4 para designar os tipos de precedência do tráfego IP. O DSCP é tratado como um índice mapeando o DSCP para uma configuração de PHB [2, 18].

Para que os pacotes trafegem numa nuvem DiffServ, eles são marcados previamente no cliente. Essa marcação é feita preenchendo o campo DSCP no cabeçalho IP de acordo com a classe dos dados que estão sendo enviados. As classes com menores valores são as que tem maior precedência.

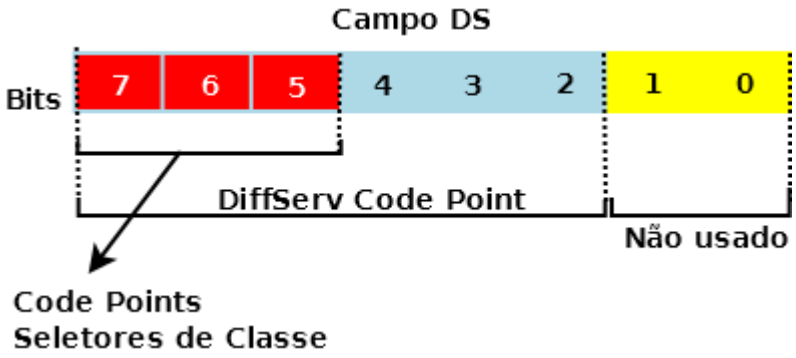

Figura 3.3: Estrutura do campo DS.

3.2.2 Arquitetura DiffServ

Um domínio DiffServ é definido como uma "porção contígua de nós DiffServ no qual opera um conjunto de planos de ação de serviços diferenciados e é administrado de forma coordenada." [22]. No domínio DiffServ dizemos que um nó é DS-capaz se ele tem capacidade de prover DiffServ. Se um nó pode suportar DiffServ ele é dito DS-compatível. Dentro de um domínio DiffServ têm-se os nós Internos DiffServ e na fronteira de um domínio DiffServ têm-se os nós DS-Fronteira.

Os nós Internos DiffServ só podem ser conectados a outros nós Internos, a um nó DS-Fronteira ou a um nó que não suporta DiffServ, mas nunca a um nó de outro domínio DiffServ, sendo a decisão de encaminhamento de pacotes feita com base no campo DSCP [20, 22]. Assim, o Condicionador de Tráfego em um nó Interno é basicamente um Classificador de Pacotes que seleciona os pacotes de acordo com seus valores de PHB e envia-os ao gerenciador de filas.

Os nós DS-Fronteira podem estar conectados a outros nós de fronteira, conectados a nós Internos do mesmo domínio DiffServ ou a um nó que não suporta DiffServ. Esses

nós também podem ser de entrada ou de saída, dependendo do sentido do tráfego. Os nós de entrada servem para garantir que os pacotes de entrada irão receber a mesma QoS do domínio anterior. Já os nós de saída usam as funções do Condicionador de Tráfego baseado no Contrato de Condicionamento de Tráfego(TCA). No TCA são estabelecidas as caracteríticas do tráfego contratado para realizar o encaminhamento dos pacotes a um domínio "diretamente conectado", as quais devem estar disponíveis em todos os nós DS-Fronteira para garantir que os pacotes de um mesmo fluxo receberão o mesmo tratamento em diferentes domínios DiffServ [2].

Embora os nós que não suportam DiffServ possam estar presentes em um domínio DiffServ, quanto maior for a quantidade de nós que não suportem DiffServ em um domínio menor será a eficiência desse domínio [21].

Perfil de Tráfego

O Perfil de Tráfego especifica a propriedade temporal de um tráfego selecionado pelo Classificador. Para tanto, fornece regras para determinar se um pacote está ou não de acordo com o perfil [2]. O conceito de dentro ou fora do perfil pode ser estendido para vários níveis de conformidade.

Aos pacotes dentro do perfil pode ser permitido entrar no domínio DiffServ sem estar condicionado ou ter seu DSCP alterado. Já os pacotes fora do perfil podem ser mapeados para um BA com menor prioridade de repasse do que o BA de um pacote dentro do perfil.

O Perfil de Tráfego é um componente opcional do TCA e o seu uso depende da oferta específica de serviços e política de provisionamento pelo domínio [2].

Classificação e Condicionamento de Tráfego

Um domínio Diffserv é extendido através dos nós de fronteira que estabelecem um SLA entre os domínios Diffserv que estão interligados. O SLA pode especificar regras de marcação de pacotes, além de poder especificar também o Perfil de Tráfego e as ações a serem tomadas com o tráfego que está dentro ou fora do especificado [2]. Abaixo são definidas as funções do Classificador e do Condicionador de Tráfego.

A função do Classificador é classificar os pacotes de entrada em múltiplos grupos com base no cabeçalho do pacote. Existem dois tipos de classificadores especificados pelo modelo DiffServ:

- Comportamento Agregado (BA)

  O Classificador BA seleciona os pacotes levando em consideração apenas o valor do campo DSCP. Normalmente, o BA é usado quando o pacote é marcado antes de chegar ao Classificador de Tráfego.

- Classificadores de Multicampos (MF)

  O Classificador de MF utiliza a combinação de um ou mais campos do cabeçalho do pacote, como endereço IP de origem, endereço IP de destino, DSCP, protocolo, portas de origem e destino. Pode ser usado para apoiar políticas de alocação de recursos mais complexas.

O Condicionador de Tráfego consiste dos componentes apresentados na Figura 3.4.

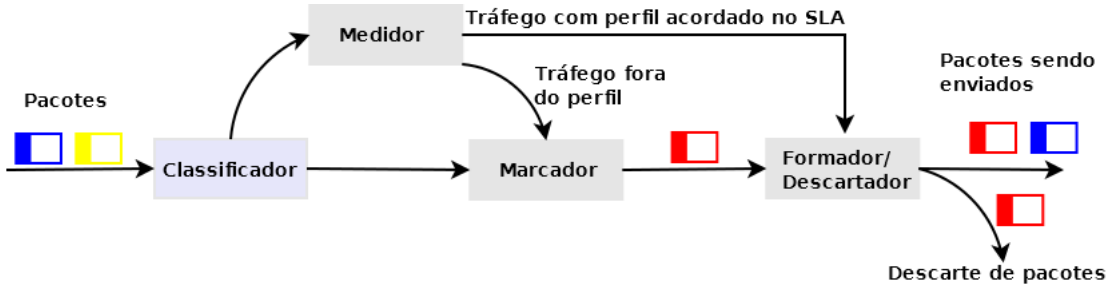

Figura 3.4: Componentes funcionais do Condicionador de Tráfego. Figura original em [28].

A seguir são descritos os componentes do Condicionador de Tráfego [2, 21, 22]:

1. Medidor: É a base do Condicionador de Tráfego. É usado para medir se o fluxo de pacotes está de acordo com o que foi acordado no SLA. É o medidor que passa informação de estado para outras funções do Condicionador de Tráfego. Com base na informação passada pelo medidor são tomadas as ações pertinentes a cada pacote.

2. Marcador: Faz a marcação do campo DSCP do pacote de entrada. A marcação pode ocorrer na aplicação do cliente ou no primeiro roteador da rede do cliente, caso a rede do cliente suporte *Differentiated Service Marking*. Caso contrário, o pacote pode ser marcado no nó DS-Fronteira. Há pacotes que podem ser marcados com um DSCP especial que indica a não conformidade com o que foi acordado no SLA. Esses pacotes serão descartados primeiro se houver um congestionamento na rede.

   Se os pacotes passarem de um domínio DiffServ para outro, os pacotes podem ter seus campos DSCP remarcados pelo Marcador de Tráfego por causa de uma violação no Perfil do Tráfego ou por que o outro domínio DiffServ usa outro tipo de DSCP para a QoS desse tráfego.

3. Formador: Faz a função de fiscalização do plano de ação do fluxo de pacotes. Para executá-la, o Formador tem um *buffer* que atrasa os pacotes que estão fora do perfil contratado, ou seja, é responsável pela adaptação do tráfego de entrada, fazendo com que este fique em conformidade com o que foi acordado no SLA.

4. Descartador: Pode ser implementado como um tipo especial de Formador. É responsável por descartar alguns ou todos os pacotes de um fluxo com o objetivo de fazer com que os pacotes do fluxo fiquem em conformidade com o especificado no SLA.

Os pacotes que não tem marcação de QoS em uma rede que usa Diffserv também tem a marcação dos seus pacotes definida pelo PHB. Com isso os nós que não contrataram nenhum serviço especial de QoS podem trafegar normalmente no domínio DiffServ.

3.2.3 Comportamento por Enlace (PHB)

O DiffServ usa o método *Behavior Aggregate* (BA) classificando os pacotes de acordo com o campo DSCP para selecionar a classe de tráfego que um pacote irá receber em cada nó DiffServ. Assim, todos os pacotes com o mesmo DSCP recebem tratamento

igual. O DiffServ é baseado em enlaces individuais e é através do *Per-Hop Behavior* (PHB) que se sabe o comportamento que cada nó DiffServ deve ter para que exista um serviço diferenciado para o cliente. O PHB é definido como um conjunto de parâmetros usados para controlar o envio de pacotes pela interface de saída [21]. Existem várias formas de implementar o PHB, em geral a implementação ocorre dentro dos nós através de algum gerenciamento de *buffers* e de mecanismos de escalonamento de pacotes, mas não é padronizada. Dentro de um nó DiffServ pode se ter tratamento distinto para comportamentos agregados distintos quando esses competem por recursos dentro do nó [2, 33].

A forma como os pacotes são tratados nos roteadores depende do comportamento do repasse de pacotes fora do roteador e da implementação interna do roteador. Com isso, nota-se que o DiffServ é baseado em saltos e na definição de PHBs individuais para cada roteador. Além disso, ao passar por um domínio DiffServ, um PHB para roteadores individuais também pode definir um caminho global fim-a-fim para prover uma melhor QoS.

Um conjunto de PHBs formam um grupo PHB. Um grupo PHB pode descrever uma alocação de recursos em termos relativos e compartilhá-los entre si. A arquitetura DiffServ define três grupos de PHB, a saber [18, 21]:

- PHB padrão

  Esse PHB é usado quando um pacote é marcado com '000000' no campo DSCP, ou seja, o pacote deve ser encaminhado pelo serviço de melhor esforço. Com isso os usuários que não contrataram nenhum serviço especial podem usar normalmente a infraestrutura do domínio DiffServ.

- PHB-EF ou Encaminhamento Expresso

  O valor recomendado de DSCP para esse grupo é: '101110'. Com o PHB-EF os pacotes são encaminhados com baixos *jitter*, atraso e descarte de pacotes. É utilizado, geralmente, para identificar e encaminhar tráfego de aplicações multimídia como voz e vídeo em tempo real. A implementação do PHB-EF pode ser feita usando um mecanismo de escalonamento de filas, como o mecanismo de Fila Prioritária (mostrado na Seção 2.4.1).

- PHB-AF ou Encaminhamento Assegurado

  O PHB-AF é usado quando os parâmetros de *jitter* e atraso não são tão importantes como a perda de pacotes. Esse grupo é recomendado para aplicações que não são baseadas em tempo-real, pois oferece apenas uma expectativa de serviço quando a rede passar por momentos de congestionamento. Para tal finalidade é usado um mecanismo de controle de tráfego que atua nas fronteiras do domínio DiffServ marcando os pacotes em função da sua prioridade. O PHB-AF visa diminuir os congestionamentos de longa duração. Como admite o congetionamento de pequena duração, um mecanismo de gerenciamento ativo de filas bastante usado é o *Random Early Detection* (RED) (mostrado na Seção 2.5.1). O PHB-AF define quatro classes de repasse com recursos próprios atribuídos e encaminhamento independente. Cada classe é dividida em 3 subclasses de precedência de descarte correspondendo a uma maior ou menor probabilidade de descarte dentro da classe.

O PHB-AF garante que os pacotes serão encaminhados com alta probabilidade de entrega, desde que o tráfego esteja de acordo com o SLA.

3.2.4 Problemas no uso do DiffServ

- Os Serviços Diferenciados dão segurança ao desempenho das aplicações em termos relativos.
- Dependendo do tráfego na rede, as aplicações podem ter uma performance abaixo de suas reais necessidades.

3.3 Comparação entre Serviços Integrados e Serviços Diferenciados

A tabela 3.1 apresenta um quadro comparativo dos modelos IntServ e DiffServ, abordando suas principais características e diferenças.

Tabela 3.1: Comparação entre IntServ e DiffServ.

Serviço	**IntServ**	**DiffServ**
Granularidade de Diferenciação do Serviço	Fluxo individual	Agregação de fluxos
Classificação de Tráfego	Garantia estatística	Garantia absoluta ou relativa
Controle de Admissão	Obrigatório	Obrigatótio para diferenciação absoluta
Protocolo de Sinalização	RSVP	Não requer para esquemas relativos
Coordenação para Diferenciação do Serviço	Fim-a-fim	A cada salto (local)
Escalabilidade	Limitado pelo número de fluxos	Limitado pelo número de classes de serviço
Gerenciamento da Rede	Similar a rede de comutação por circuito	Similar a rede IP

O modelo DiffServ provê QoS em redes de computadores pela diferenciação do tráfego enquanto que o IntServ provê QoS pela construção de um circuito virtual através da técnica de reserva de recursos. Devido a necessidade de reserva, o IntServ requer que cada nó da rede guarde a informação de cada fluxo.

A opção para o uso do modelo DiffServ ocorre naturalmente devido ao fato deste fornecer maior escalabilidade devido à menor granularidade na alocação de recursos, pois não é necessário efetuar uma reserva específica para cada fluxo. Os fluxos são agregados em classes de serviços Diffserv de modo que não há controle de estado individual para cada fluxo. Também não é necessária a sinalização entre cada roteador da rede. Se ao invés do modelo DiffServ fosse usado o IntServ teria-se uma maior carga de processamento

nos roteadores de núcleo implicando em um diminuição considerável do desempenho da rede.

Capítulo 4

Voz sobre IP - VoIP

O transporte de voz sobre IP pode reduzir drasticamente o custo de chamadas telefônicas à longa distância, isso porque a tecnologia IP tem um custo baixo e a banda passante na Internet é livre. Assim, aplicativos em tempo real estão cada vez mais presentes no mercado. O problema é que muitos desses aplicativos usam protocolos e algoritmos proprietários, o que prejudica a interoperabilidade entre eles.

Protocolos de sinalização, como a recomendação H.323, definida pelo ITU-T e o Protocolo de Iniciação de Sessão (SIP), especificado pela IETF na RFC 3261 são responsáveis pelo controle das chamadas, devendo especificar a codificação de voz, o modo de transporte de dados e de autenticação, formas de segurança, entre outros. Como protocolo de transporte, pode-se utilizar o *Real-Time Transport Protocol* (RTP) que transporta dados em tempo real, ou o *Secure Real-time Transport Protocol* (SRTP) o qual define um encapsulamento seguro para o RTP e que foi especificado na RFC 3711 [9].

4.1 Pilha de Protocolos VoIP

O VoIP consiste de um número de protocolos trabalhando em conjunto para realizar a função completa. Isto é feito de forma hierárquica e pode ser representado por uma pilha chamada Pilha de Protocolos VoIP [1, 22]. A Pilha de Protocolos VoIP é motrada na Figura 4.1.

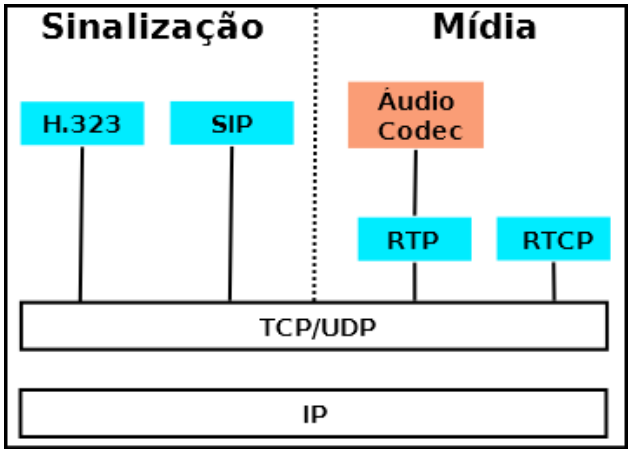

Figura 4.1: Pilha de Protocolos VoIP.

Tais protocolos são divididos em três categorias [9, 22]:

1. Protocolos de Sinalização:

   Recomendação H.323-ITU-T
   : Compreende um conjunto de especificações que definem entidades, protocolos e procedimentos para o estabelecimento, controle e término de comunicações multimídia sobre redes de comutação por pacotes. Por ser mais antigo e complexo, tem sido pouco utilizado em aplicações VoIP.

   SIP (*Session Initiation Protocol*)-IETF
   : Apresenta as mesmas funcionalidades do H.323. Como é menos complexo e mais recente, é mais utilizado em aplicações VoIP.

2. Protocolos de Controle de *Gateway*:

   MGCP (*Media Gateway Control Protocol*)-IETF
   : Protocolo mestre-escravo utilizado para controlar as conexões nos *gateways* de aplicações VoIP, definindo eventos, sinais, mapas de dígitos, transações, entre outros. Como principais capacidades do MGCP, têm-se [22]:

     - Determinar a localização do destinatário;
     - Determinar a capacidade de mídia do destinatário;
     - Determinar a disponibilidade do destinatário;
     - Estabelecer conexão entre origem e destino.

   MEGACO/H.248 (*Media Gateway Control Protocol*)-IETF/ITU-T
   : Do ponto de vista de arquitetura e funcionamento é similar ao MGCP. Porém, apresenta diferenças importantes, como um melhor suporte a conferências de áudio e vídeo multiponto.

3. Protocolos de Mídia (Transporte de voz):

   RTP (*Real-Time Transport Protocol*)-IETF
   : Responsável pelo transporte do fluxo de mídia em tempo real entre computadores e *gateways*.

   RTCP (*Real-Time Transport Control Protocol*)-IETF
   : Responsável pelo controle de transpote do fluxo de mídia realizado pelo RTP.

4.2 *Session Iniciation Protocol* (SIP)

É um protocolo de aplicação e sinalização usado para estabelecer, manter e terminar sessões ou chamadas multimídia utilizando o modelo requisição-resposta, similar ao *Hiper Text Protocol* (HTTP) [24]. O SIP suporta sessões *unicast* e *multicast*. Como principais características do SIP pode-se citar [24]:

- É um protocolo "fora de banda";

- As mensagens são recebidas e enviadas em portas diferentes das portas usadas para receber e enviar dados de mídia;
- Mensagens SIP são escritas em ASCII;
- Pode rodar sobre TCP ou UDP;
- Os endereços SIP podem ser informados em páginas Web, email, grupos ou diretórios LDAP.

O protocolo SIP é capaz de prover [1, 31]:

- Mecanismos para estabelecer chamadas entre dois interlocutores através de uma rede IP;
- Mecanismos que permitem ao chamador determinar o endereço IP corrente de quem é chamado;
- Mecanismos para gerenciamento de chamadas.

Os principais componentes do SIP são descritos abaixo [17]:

User Agents

Os *User Agents* podem ser um terminal SIP ou um software na estação final [24]. Para fazer as chamadas, utilizam endereços parecidos com o de email, chamados de SIP URL.

Os *User Agents* podem ser do tipo:

- Cliente ou UAC, os quais são capazes de iniciar requisições SIP.
- Servidor ou UAS, os quais são capazes de receber e responder requisições SIP.

Dessa forma, temos que um *User Agent* funciona como cliente quando solicita inicialização de uma sessão e como servidor quando responde a um pedido de sessão. Além disso, o *User Agent* armazena e gerencia estados de sessões.

Network Servers

Os *Network Server* são divididos três grupos [31]:

Servidor *Proxy*

É usado quando um cliente remetente SIP deseja obter o endereço IP do dispositivo usado pelo cliente receptor e não consegue obtê-lo diretamente. Para isso, o remetente envia uma mensagem INVITE para o servidor *Proxy*. A mensagem INVITE contém os campos [24]:

- Versão do SIP usado;
- Cabeçalho Via onde é colocado o endereço IP de cada dispositivo SIP em que a mensagem passa;
- *From* e *To*, contendo um endereço SIP do remetente e do receptor, respectivamente;

- *Call-ID* ou identificador de chamadas;
- *Content-Type* ou Tipo do Conteúdo, que serve para definir o formato usado para descrever o conteúdo da mensagem SIP;
- *Content-Length* ou Tamanho do Conteúdo, que informa o comprimento, em bytes, da mensagem;
- Informação sobre o endereço IP do remetente e tipo de áudio que o remetente quer receber.

Servidor de Redirecionamento

O Servidor de Redirecionamento serve, como o próprio nome diz, para redirecionar as mensagens do Servidor *Proxy* ao longo do caminho até encontrar o endereço IP do destinatário da mensagem.

Servidor de Registro

Cada usuário SIP tem um Servidor de Registro associado. Ao iniciar uma aplicação SIP em um dispositivo é enviada uma mensagem de Registro SIP contendo o endereço IP do dispositivo ao Servidor de Registro. Ou seja, o dispositivo usado pelo usuário de uma aplicação SIP deve registrar-se obrigatoriamente em um Servidor de Registro antes de efetuar qualquer chamada. Quando um usuário SIP deseja interagir com outro usa o Servidor *Proxy* para enviar uma mensagem INVITE ao Servidor de Registro SIP do destinatário. Este, por sua vez, deve encaminhar a mensagem INVITE recebida do Servidor *Proxy* para o dispositivo do destinatário que executa a aplicação SIP. Assim, o destinatário pode enviar uma mensagem de resposta SIP ao remetente.

Mensagens SIP

As mensagens SIP são as unidades básicas de comunicação SIP e contêm uma sequência estruturada de octetos formando uma sintaxe definida. Tantos as mensagens de requisição quanto de resposta possuem uma linha de início (*Start-Line*), um ou mais cabeçalhos (*Headers*), uma linha vazia (*Carriage-Return Line Feed*-CRLF) que indica o fim dos cabeçalhos e o corpo da mensagem (*Message Body*), o qual é opcional. A RFC 2543 [24] define seis tipos de métodos que podem ser usados em requisições e a IETF adicionou o método INFO que é um mecanismo para carregar informações de controle durante uma sessão. Na tabela 4.1 são descritos os métodos das Requisições SIP.

Tabela 4.1: Métodos de Requisições SIP [15].

Mensagem SIP	**Descrição**
INVITE	Convida um usuário a participar de uma chamada
ACK	Usado para facilitar a troca confiável de mensagens INVITE
OPTIONS	Solicita informações sobre as funcionalidades do servidor
BYE	Termina uma conexão entre usuários ou rejeita uma chamada
CANCEL	Termina uma requisição ou procura por um usuário
REGISTER	Registra a localização atual de um usuário
INFO	Usado para troca de sinalização durante a sessão

Depois de receber e interpretar as requisições, o receptor responde através de mensagens de resposta. A primeira linha da mensagem de reposta é a *Status-Line* a qual consiste da versão do protocolo, seguida por um código numérico de 3 dígitos inteiros chamado de *Status-Code* ao qual é associado um texto e que serve para indicar a tentativa de entender e satisfazer uma requisição [24]. A tabela 4.2 contém a descrição das Mensagens de Resposta SIP.

Tabela 4.2: Mensagens de Respostas SIP [24].

Mensagem SIP	**Descrição**
1xx	Informativo: requisição recebida, continuando o processo de requisição
2xx	Bem sucedido: a ação foi recebida com sucesso, entendida e aceita
3xx	Redirecionado: uma ação mais adiante precisa ser tomada para que a requisição possa ser completada
4xx	Erro do cliente: a requisição contém sintaxe falha ou não pôde ser executada pelo servidor
5xx	Erro do servidor: o servidor falhou em executar uma requisição aparentemente válida
6xx	Falha global: a requisição não pôde ser executada pelo servidor

4.2.1 *Real-Time Protocol* (RTP)

O RTP é um protocolo que proporciona mecanismos de transporte usados para sincronização de sequências de dados multimídia. Para tal finalidade, usa uma estrutura de pacote padronizada incluindo campos para dados de áudio/vídeo, números de sequência e marcas de tempo. Logo, o receptor pode sincronizar os dados multimídia enviados

pelo emissor usando o protocolo RTP [12, 22]. O RTP permite que para cada fonte seja atribuída uma corrente RTP independente. Com isso, numa videoconferência entre dois participantes podem ser abertas duas correntes RTP: uma para o áudio e outra para vídeo. Este protocolo é complementar aos protocolos interativos em tempo real como o SIP e o H.323. A Figura 4.2 mostra os dados multimídia sendo encapsulados pelo protocolo RTP.

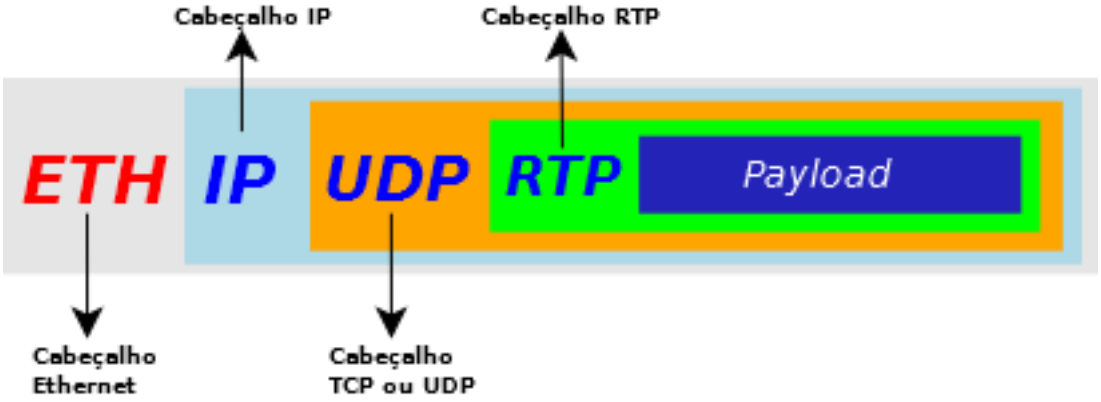

Figura 4.2: Conteúdo da mensagem sendo encapsulado com o RTP.

Como os roteadores não distinguem datagramas IP que carregam RTP dos que não o carregam, o RTP não pode fornecer mecanismos que assegurem a entrega de dados a tempo, QoS e nem mesmo impedir a entrega de pacotes fora de ordem [17].

Pacotes RTP podem ser usados em aplicações *unicast*, *multicast* um-para-muitos e muitos-para-muitos. No último caso, os remetentes e fontes da sessão *multicast*, em geral, usam o mesmo grupo *multicast* para enviar suas correntes RTP. Quando correntes *multicast* RTP tem o mesmo destino formam um grupo e pertencem a uma mesma sessão RTP. A Figura 4.3 apresenta os campos do cabeçalho RTP.

Figura 4.3: Formato do Cabeçalho RTP. Figura original em [12].

A seguir serão descritos os campos do cabeçalho do protocolo RTP [9, 12, 22]:

V (*Version*): Identifica a versão do protocolo;

P (*Padding*): Bit de Preencimento . Serve para informar se o pacote foi completado com um ou mais bits que não fazem parte do *payload*;

X (*Extension*): Bit de Extensão. Indica a presença de cabeçalho de extensão;

CC (*Contributing Source Counter*): É um contador que serve para indicar o número de identificadores CSRC após o cabeçalho fixo;

M (*Marker Bit*): Bit de Marcação. Usado para marcar o começo de um quadro de vídeo ou outro elemento reconhecido pela aplicação;

PT (*Payload Type*): Especifica o formato do conteúdo no pacote RTP e determina como será interpretado pela aplicação;

Número de sequência (*Sequence Number*): Cada pacote RTP tem um número que serve para o receptor restaurar a ordem original dos pacotes e detectar uma possível perda de pacotes;

Marca de tempo (*Timestamp*): Contém o momento em que os dados no pacote RTP foram amostrados;

Identificador SSRC (*Syncronization Source*): Indica a qual fonte o fluxo de dados pertence. Cada pacote RTP tem um único identificador SSRC para que o receptor possa agrupar os pacotes por origem e reproduzí-los;

Lista CRSC (*Contributing Source*): Identifica as fontes que contribuem para a formação do pacote RTP. Os identificadores de CRSC são inseridos nos misturadores usando os identificadores SSRC que contribuíram para a geração do pacote RTP.

Como principais características do RTP pode-se destacar [17]:

- Roda normalmente sobre UDP;
- Pacote RTP é formado por cabeçalho e *payload*;
- Encapsulamento do RTP só é visto nos sistemas finais;
- Pacotes RTP podem ser enviados também para árvores *multicast* um-para-muitos ou muitos-para-muitos;
- Permite a cada fonte ter sua corrente independente de pacotes RTP.

Conversores e Misturadores

O uso de conversores e misturadores é aceito no protocolo RTP os quais servem para modificar o fluxo RTP na presença de *Firewall* e/ou de um enlace de baixa largura de banda [12].

Os conversores, normalmente, não são detectados pelos receptores e servem para encaminhar os pacotes RTP sem modificar o identificador de origem. Assim, os receptores podem identificar o remetente original do pacote RTP apesar de pacotes partindo de vários remetentes passarem pelo mesmo conversor e carregarem o endereço daqueles [12, 22].

Os misturadores servem para combinar fluxos de áudio RTP provenientes de remetentes diferentes em um só fluxo de áudio RTP [22]. Como existe uma grande probabilidade dos remetentes que contribuíram não estarem em sincronismo, o misturador torna-se a fonte de sincronização dos pacotes uma vez que gera o tempo de sincronismo para o fluxo combinado. Por este motivo, o misturador coloca o seu identificador SSRC em todos os

pacotes do fluxo de dados combinados que serão enviados. Para manter a identidade dos remetentes originais, o misturador inclui os identificadores SSRC dos remetentes na lista de identificadores CSRC [12].

4.2.2 Protocolo de Controle do RTP (RTCP)

Aplicativos em tempo real necessitam de um retorno acerca da qualidade de recepção dos pacotes RTP que foram enviados. Este retorno é feito através de pacotes RTCP os quais também servem para monitorar a entrega de pacotes RTP de forma escalável em redes *multicast* e para fornecer um controle sobre os pacotes RTP transmitidos. Dentre as funções do RTCP pode-se citar [12]:

1. Fornecer *feedback* sobre a qualidade da distribuição dos dados. Isto é feito através dos relatórios de emissor e receptor;
2. Carregar um identificador de nível de transporte chamado CNAME para que o receptor possa monitorar cada participante. Isto é necessário, pois existe a possibilidade do SSRC ser trocado em caso de descoberta de conflito com outra corrente multimídia RTP usando o mesmo SSRC;
3. Calcular a taxa de envio de relatórios de controle RTCP para não sobrecarregar a rede com envio de relatórios RTCP.

Para executar suas funções, o RTCP produz pacotes que podem ser [12, 17, 22]:

Relatórios de Emissor ou de Receptor: contém dados estatísticos sobre o fluxo e a contagem de pacotes. Os relatórios de emissor ou SR(*Sender Report*) são usados para informar aos receptores sobre o que eles devem ter recebido pelos emissores no fluxo de pacotes RTP; Já os relatórios de receptor ou RR(*Receiver Report*) servem para informar aos emissores a qualidade de recepção dos pacotes RTP, por exemplo: pacotes perdidos, *jitter*, atraso etc.;

Source Description Items ou SDES: Itens de Descrição do Emissor. São usado pelo emissor dos pacotes RTP para distribuir informações sobre o emissor, incluindo o CNAME;

BYE: é usado para informar o fim da participação em uma conferência. Quando é usado por misturadores, este envia todas as origens que contribuem para a formação do pacote RTP;

APP: uso experimental e serve para funções específicas dos aplicativos.

Para cada corrente RTP que um remetente enviar é criado e transmitido um relatório de remetente RTCP o qual é enviado em uma porta diferente da que é usada para enviar a corrente RTP. Este relatório serve para informar os receptores o que devem ter recebido na corrente RTP enviada, por exemplo, quantitativo de pacotes enviados na corrente. Dessa forma, quando o remetente envia uma corrente RTP para o receptor, ao recebê-la, este deve gerar e enviar para o remetente um relatório de recepção. Ao receber o relatório do receptor, o remetente saberá como estão sendo recebidos os pacotes RTP e poderá

tomar alguma providência acerca do envio da corrente RTP. Outra forma do receptor enviar pacotes RTCP é agregando os relatórios de recepção em um único pacote RTCP e enviá-lo para a árvore *multicast* que conecta todos os participantes da sessão. Um único pacote RTCP também pode concatenar relatórios de remetente e de receptor sendo este pacote encapsulado em um segmento UDP e enviado para a árvore *multicast* [17].

Escalabilidade da largura de banda do RTCP

O RTCP tenta usar um limite de 5% da largura de banda da sessão para a transmissão de seus pacotes. 25% dessa taxa é destinada aos emissores e os outros 75% é destinada aos receptores. Pode-se notar que dependendo do número de participantes da sessão o RTCP muda a taxa com a qual um participante envia pacotes RTCP para a árvore *multicast*. Como cada participante envia pacotes de controle (RTCP) para todos os participantes, pode-se estimar o número de participantes da sessão [12].

4.2.3 Protocolo de Fluxo Contínuo em Tempo Real (RTSP)

A RFC 2326 contém a especificação do RTSP. Este protocolo permite ao usuário controlar a reprodução de uma corrente de mídia. As mensagens RTSP podem ser enviadas por TCP ou UDP. Estas mensagens não são transmitidas no mesmo pacote que a corrente de mídia e por esse motivo o RTSP é chamado de protocolo fora de banda [25]. A função do RTSP é trocar informações entre o transdutor e o servidor. O transdutor informa ao servidor de fluxo contínuo em tempo real a solicitação do usuário e o servidor de mídia responde com outra mensagem RTSP de acordo com a solicitação [17]. O protocolo RTSP suporta as seguintes operações [25]:

Recuperação de mídia a partir do servidor de fluxo contínuo:

> O cliente pode solicitar uma descrição de apresentação de mídia via HTTP. Se a apresentação for em *multicast*, a descrição da apresentação contém os endereços *multicast* e as portas que serão utilizadas. Se for *unicast* o destino é fornecido pelo cliente.

Convidar um servidor para conferência:

> Um servidor de mídia pode ser convidado para participar de uma conferência existente para gravar toda ou partes dos meios de comunicação ou para reproduzir mídia em uma apresentação.

Existência de uma mídia disponível para apresentação:

> Em uma apresentação ao vivo, é útil o servidor poder informar ao cliente que existe uma mídia disponível.

Como principais características do RTSP podem ser citadas [25]:

- Novos métodos e parâmetros podem ser adicionados facilmente;
- Pode ser analisado por HTTP e MME;
- Reutiliza mecanismos de segurança na Web;

- Pode controlar dispositivos de reprodução e gravação, assim como alternar entre os dois;
- Pode ser manipulado facilmente por *Proxy* e *Firewall*;
- Reutiliza os conceitos do HTTP;
- Negocia o método de transporte antes de enviar a mídia;
- Monitora o estado do cliente para cada sessão em curso.

A Figura 4.4 apresenta um exemplo de conexão usando mensagens RTSP.

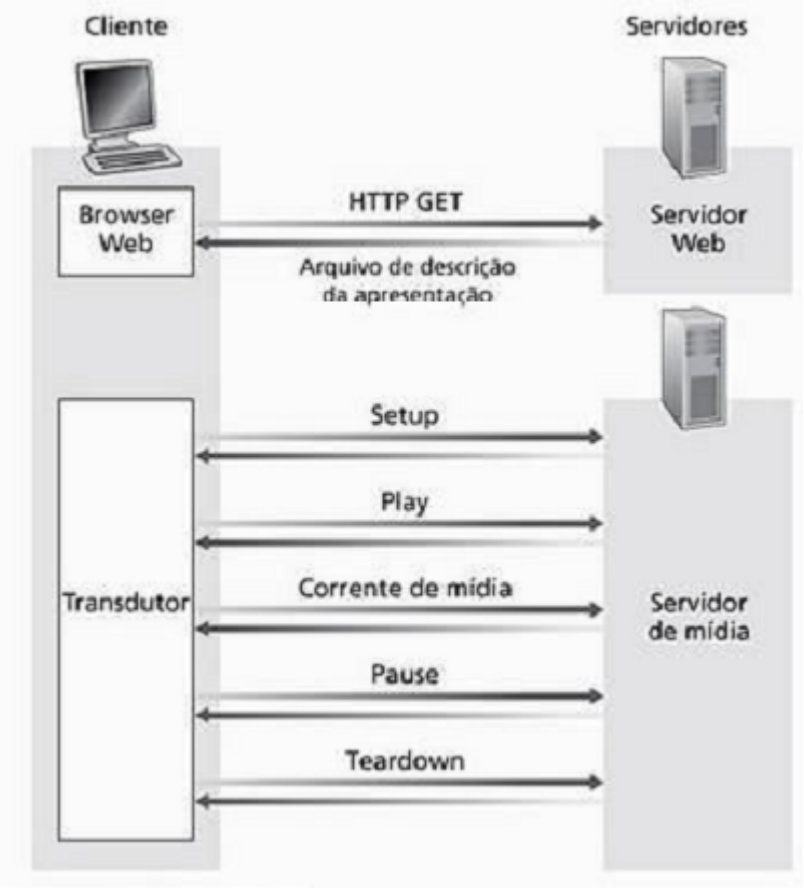

Figura 4.4: Interação entre Cliente e Servidor usando RTSP. Figura original em [17].

1. O browser do cliente encapsula uma solicitação do arquivo de descrição de apresentação e envia ao servidor Web;
2. O servidor Web encapsula o arquivo de descrição em uma mensagem de resposta HTTP e envia a mensagem ao browser;
3. Ao chegar a mensagem de resposta HTTP, o browser verifica o tipo de conteúdo da mensagem. De acordo com o conteúdo, chama um transdutor que usará o arquivo de descrição recebido do servidor Web que contém referências às correntes de mídia. Com isso, o transdutor e o servidor onde estão as correntes de mídia trocam uma

série de mensagens RTSP para que o transdutor possa reproduzir as correntes de mídia.

4.3 Requisitos de QoS para VoIP

O tráfego VoIP requer um serviço de prioridade explícita e garantias de largura de banda para a sinalização e para a chamada propriamente dita. Logo, mostramos uma lista de requisitos e recomendações mínimos para que o tráfego VoIP funcione corretamente [10, 30]:

- O tráfego VoIP pode ter o campo DSCP do pacote marcado baseando-se na RFC 3246 [8];
- Não podem ocorrer perdas de pacotes acima de 10%;
- A latência de ida ou volta não pode ser maior do que 150 ms;
- A média do *jitter* de ida ou volta não pode ser superior a 30 ms.

Abaixo foram relacionados os fatores que afetam uma chamada VoIP e os problemas decorrentes [30]:

Perda

Perda provoca períodos de silêncio em uma conversação. Para inibir as perdas decorrentes do atraso dos pacotes nos *buffers* existem técnicas que mascaram os efeitos dos pacotes VoIP perdidos ou descartados. Com essas técnicas pode-se mascarar a perda de até 20 ms de amostras.

Latência

Latência pode provocar a degradação da qualidade da voz se for maior que 150 ms em um caminho unidirecional fim a fim. Se a latência for superior a 150 ms a conversação parece estar mais lenta.

Jitter

O *jitter* causa degradação em sistemas de transmissão de alta velocidade. Seus efeitos fazem-se sentir ao afetar o processo de recuperação de dados causando perdas de dados durante a transmissão. Além disso, o *jitter* introduz distorção no processamento da informação na recepção, necessitando de mecanismos de controle e compensação que variam de aplicação para aplicação. Uma das soluções mais comuns para o problema é a utilização de *buffers jitter*.

Os *buffers jitter* servem para alterar as chegadas assíncronas em um fluxo síncrono. Se o *buffer jitter* for definido grande ou pequeno irá impor restrições desnecessárias sobre as características da rede. Um *buffer* grande gera um aumento no atraso fim a fim. Enquanto que um *buffer* pequeno pode gerar *underflows buffer* ou transbordamento de *buffer*.

4.4 Considerações Finais

Voz sobre IP é uma tecnologia que permite o tráfego de voz pelas redes de computadores. Esta tecnologia vem se tornando uma realidade em grandes instituições, pois através dela obtém-se benefícios como redução do custo de chamadas telefônicas à longa distância e gastos com aparelhos telefônicos convencionais. Para realizar a comunicação entre os equipamentos utilizados na tecnologia VoIP foi definida uma Pilha de Protocolos VoIP. Essa Pilha de protocolos trabalha em conjunto e de forma hierárquica sendo dividida em: Protocolos de Sinalização, Protocolos de Controle de *Gateway* e Protocolos de Transporte de Mídia. O SIP é um protocolo de Sinalização e Aplicação usado para estabelecer, manter e terminar sessões multimídia. O RTP é um protocolo de Transporte de Mídia usado para sincronização de sequências de dados multimídia. O protocolo RTCP é usado pelo RTP para monitoramento da qualidade da recepção e entrega dos pacotes enviados. Para que funcionem corretamente as aplicações VoIP requerem um serviço de prioridade explícita e garantias de largura de banda para sinalização e chamada. Os requisitos mínimos de QoS para VoIP são: Atraso fim-a-fim de até 150 ms, Perda de Pacotes de até 10% e *Jitter* médio de até 30 ms.

Capítulo 5

Experimentos e Resultados

O *Best-Effort* é um serviço de rede no qual não se provê garantias de entrega para os pacotes dos seus usuários. Além disso, todos os seus usuários compartilham a mesma largura de banda. Para que os pacotes de um fluxo cheguem ao destino, eles utilizam as rotas definidas e a largura de banda que estiver disponível. Quando ocorre congestionamento pacotes são descartados sem distinção.

Os testes realizados com o modelo de Serviços Diferenciados (DiffServ) foram desenvolvidos para atestar a eficiência do modelo Diffserv na rede corporativa da empresa *Freedom Network* em comparação ao *Best-Effort*. Para conseguir esse objetivo foram usadas as funcionalidades do simulador NS-2(*Network Simulador*) realizando simulações utilizando a topologia e características de tráfego da rede da empresa *Freedom Network*. Foi simulado um cenário tratando os pacotes VoIP com QoS (DiffServ) e outro cenário sem tratamento QoS aos pacotes VoIP (*Best-Effort*).

Este capítulo é organizado da seguinte maneira: na Seção 5.1 é apresentado o simulador NS-2 e na Seção 5.2 é descrito o ambiente usado para implementar as simulações. Na Seção 5.3 é descrito como foram implementadas as simulações e na Seção 5.4 é mostrado o cenário de avaliação, as características do tráfego e a topologia de rede da empresa *Freedom Network*. Por fim na Seção 5.5 temos a avaliação do modelo de Serviços Diferenciados em relação ao *Best-Effort*.

5.1 *Network Simulator-2*

O *Network Simulator v2* [14] é um simulador de eventos discretos resultante de um projeto conhecido como VINT (*Virtual InterNetwort Testbed*). Dentre outros compõe esse projeto a DARPA, USC/ISI, Xerox PARC, LBNL e a Universidade de Berkeley. Como principal vantagem do NS-2 tem-se o fato deste ser gratuito, ter código aberto e ser validado pela comunidade científica. O simulador oferece suporte à simulação de um grande número de tecnologias de rede com e sem fio, cenários baseados nos protocolos TCP e UDP, diversos escalonadores e políticas de fila e caracterização de tráfego com diversas distribuições estatísticas.

O NS-2 utiliza duas linguagens: C++ para estrutura básica (protocolos, agentes, etc) e *Object-oriented Tool Command Language*(OTCL) para uso como *frontend*. OTCL é uma linguagem interpretada, desenvolvida pelo *Massachusetts Institute of Technology*(MIT). Nesta linguagem serão efetivamente descritas as simulações. O motivo para se usarem

duas linguagens deve-se ao fato de usar uma linguagem mais robusta para a manipulação de bytes, pacotes e para implementar algoritmos que rodem um grande conjunto de dados. Para esse contexto C++, que é uma linguagem compilada e de uso tradicional, mostrou-se a ferramenta mais eficaz. Em contrapartida é fato que, durante o processo de simulação, ajustes são necessários com certa frequência. Assim haveria uma perda de tempo e um desgaste muito grande se, a cada mudança de parâmetro, houvesse a necessidade de se compilar o programa para testá-lo. Já o uso da linguagem OTCL, que é interpretada, evita esse desgaste por parte do usuário, pois há uma simplificação no processo interativo de mudar e re-executar o modelo.

A Figura 5.1 mostra a estrutura de uma simulação no NS-2. Para se construir uma rede conectam-se os nós por meio de enlaces. Eventos são escalonados para passar entre os nós através dos enlaces. Aos nós e enlaces podem ser associadas várias propriedades. Agentes podem ser associados aos nós, os quais são responsáveis por gerar diferentes tipos de tráfego. A cada agente em particular é associada uma aplicação. Além disso, os agentes necessitam de receptores específicos para receberem o tráfego que foi gerado. No caso do agente TCP (*Transmission Control Protocol*) esse receptor é chamado de *sink* e deve gerar os pacotes de reconhecimento ou ACK(*Acknowledge*).

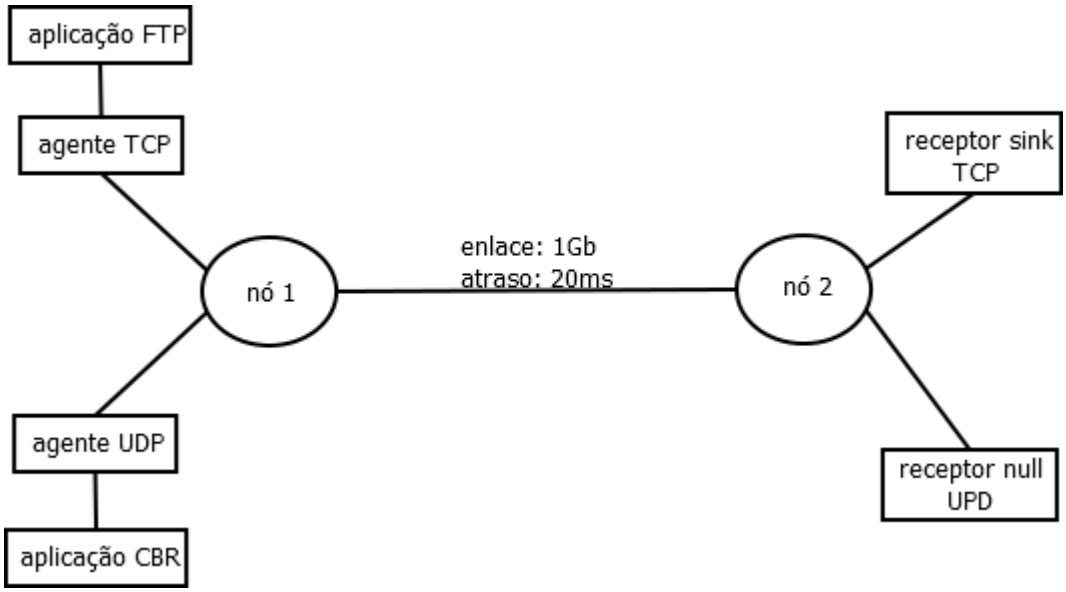

Figura 5.1: Estrutura de uma simulação no NS-2.

Uma das etapas mais importantes em uma simulação é definir os eventos que ocorrerão durante a execução da mesma. Assim, quando uma nova simulação é inicialiada, são executadas as seguintes operações:

- Definição do formato do pacote;
- Criação do escalonador de eventos.

O NS-2 fornece um conjunto de interfaces de configuração e quatro tipos de escalonadores de eventos para conduzir a simulação da melhor maneira possível, sendo o escalonador de eventos responsável por selecionar o evento mais antigo e executá-lo até o fim. Este

processo é realizado até que o último evento seja executado. Como o NS-2 é um simulador orientado a eventos, a definição da dinâmica da simulação pode ser realizada atribuindo-se um tempo total de simulação e, durante este período, invocar eventos específicos, como início de tráfegos, quedas ou perdas de dados nos enlaces, entre outros.

Ao fim da simulação o NS-2 gera um arquivo de log contendo todos os eventos ocorridos durante a simulação em um arquivo texto chamado de *Trace*. Na Figura 5.2 é apresentado o formato do *Trace*:

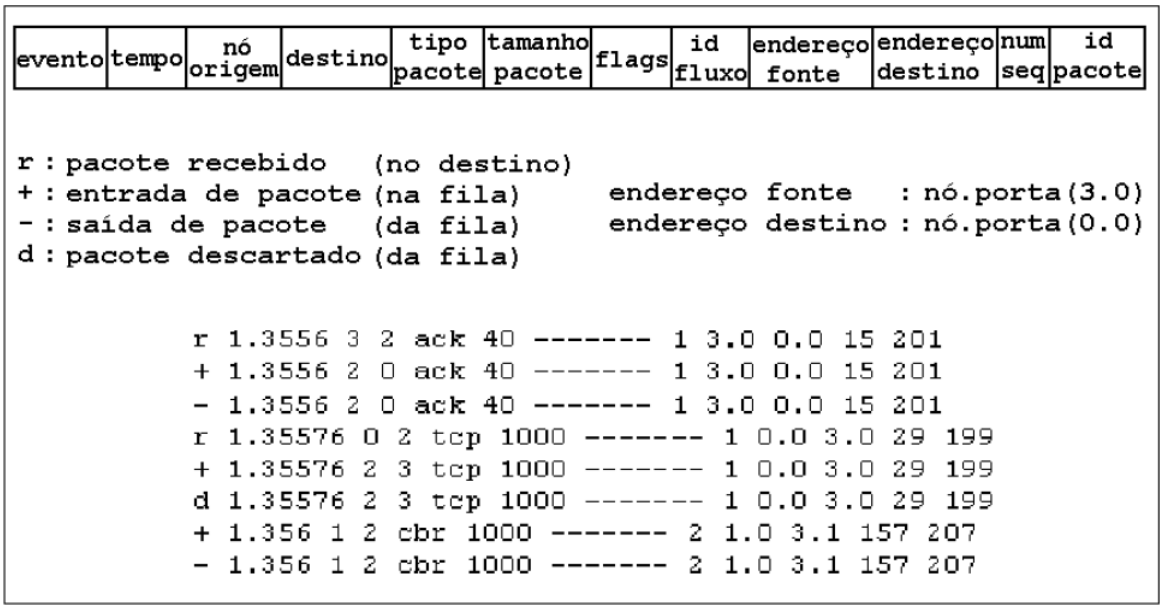

Figura 5.2: Formato do *Trace file*. Figura original em [7].

O primeiro campo diz respeito ao evento ocorrido. O campo seguinte é o tempo em que ocorreu o evento. Os próximos campos identificam os nós de origem e destino do enlace em que o pacote está. Logo após tem-se os campos tipo e tamanho (em bytes) do pacote. A seguir tem-se uma série de flags relacionadas a notificação de congestionamento e por fim tem-se os campos identificação do fluxo, endereços de fonte e de destino, número de sequência do pacote e um identificador único do pacote na rede.

5.2 Ambiente de Simulação

Para implementar as simulações foi usado um servidor HP ML Proliant com 1 processador Xeon de 3.1 GHz, 12 GB de memória RAM, HD de 1TB. O sistema operacional usado foi o Debian 7 64bits. Para instalar o NS-2 foram realizados os seguintes passos:

- Instalar as depedências necessárias através do comando:

  `apt-get install build-essential autoconf automake libxmu-dev`

- Download do NS-2.35 através do site:

  http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download

- Após efetuar o download, descompactar o arquivo na pasta home do usuário:

  tar zxvf ns-allinone-2.35.tar.gz

- Editar o arquivo **ls.h** que se encontra na pasta ns-allinone-2.35/ns-2.35/linkstate/ e modificar a linha 137, acrecentando o comando **this->** conforme mostrado abaixo.

  void eraseAll() { **this->**erase(baseMap::begin(), baseMap()::end()); }

- Ao terminar as alterações, pode-se instalar o NS-2 entrando na pasta ns-allinone-2.35/ e executar o seguinte comando:

  ./install

- Concluída a instalação, entre na pasta home do usuário e execute os seguintes comando para incluir as variáveis de ambiente do NS-2 no arquivo **.bashrc**:

  nano .bashrc

- Inclua as linhas abaixo no final do arquivo **.bashrc**, lembrando de trocando ‘‘`nome_usuario`’’ pelo nome da pasta do usuário do sistema:

```
# LD_LIBRARY_PATH
OTCL_LIB=/home/nome_usuario/ns/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/nome_usuario/ns/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/home/nome_usuario/ns/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/home/nome_usuario/ns/ns-allinone-2.35/bin:/home/nome_usuario/ns/ns-allinone-2.35/tcl8.5.10/unix:
/home/nome_usuario/ns/ns-allinone-2.35/tk8.5.10/unix

NS=/home/nome_usuario/ns/ns-allinone-2.35/ns-2.35/
NAM=/home/nome_usuario/ns/ns-allinone-2.35/nam-1.15/
PATH=\$PATH:$XGRAPH:$NS:$NAM
```

- Permaneça na pasta home do usuário e recarregue o arquivo **.bashrc** através do comando:

  source .bashrc

5.3 Experimentos

Para a realização dos experimentos, foram criados scripts no formato OTCL. Os scripts podem ser encontrados nos Apêndices e são organizados em módulos com funções descritas abaixo.

5.3.1 Módulos da Simulação *Best-Effort*

A simulação do cenário *Best-Effort* (sem QoS) é composta pelos seguintes módulos:

Módulo de Configuração (Apêndice D):

Responsável pela inicialização de variáveis da simulação.

Módulo de Criação de Cenário (Apêndice F):

Responsável pela criação da topologia da rede corporativa da empresa *Freedom Network*. Esse módulo contém as características de toda a topologia da rede, tais como:

- Criação de nós;
- Criação de enlaces;
- Tipo de filas usados pelos enlaces;
- Largura de banda dos enlaces;
- Atraso;
- Limite de pacotes suportados em cada enlace.

Módulo de Tráfego (Apêndice G):

Responsável por criar o tráfego de uma origem a um destino definindo quais os agentes responsáveis pela camada de transporte e camada de aplicação de cada tráfego criado. Contém ainda a caracterização do tráfego: taxa de transmissão, tamanho de pacote, identificador de fluxo e de classe do tráfego, tempo inativo e ativo de cada fonte.

Módulo de Criação de Fontes (Apêndice E):

Responsável por gerar os valores das variáveis aleatórias dos tráfegos usados, inicializar os contadores de fontes ativas e chamar os procedimentos para criação de cada tipo de tráfego.

Módulo de Procedimentos de Fontes (Apêndice H):

Responsável por conter todos os procedimentos para inicializar, verificar status, incrementar e decrementar a quantidade de fontes ativas de todos os tipos de fontes usados na simulação.

Módulo de Inicialização de Eventos (Apêndice I):

Responsável por chamar os procedimentos específicos para inicializar cada um dos tipos de fontes criadas pelo **Módulo de Criação de Fontes**.

Módulo de Execução (Apêndice J):

Responsável pela junção e chamada dos módulos componentes. Neste módulo são definidas quais filas serão monitoradas além de informar a quantidade de fontes ativas para cada tráfego durante toda a simulação. É o módulo chamado para executar a simulação.

5.3.2 Módulos da Simulação DiffServ

A simulação DiffServ (com QoS) é composta pelos mesmo módulos da simulação *Best-Effort* acrescentando-se os seguintes módulos:

Módulo de Criação de Políticas e PHB (Apêndice L):

Responsável pela criação do comportamento que cada nó deverá apresentar para o cumprimento do SLA que foi especificado.

Módulo de Criação de Cenário (Apêndice K):

Responsável pela criação da topologia da rede corporativa da empresa *Freedom Network*. Esse módulo contém as características de toda a topologia da rede, tais como:

- Criação de nós;
- Criação de enlaces;
- Tipo de filas usados pelos enlaces;
- Largura de banda dos enlaces;
- Atraso;
- Limite de pacotes suportados em cada enlace.

Módulo de Execução (Apêndice M):

Responsável pela junção e chamada dos módulos componentes. Neste módulo são definidas quais filas serão monitoradas além de informar a quantidade de fontes ativas para cada tráfego durante toda a simulação. É o módulo chamado para executar a simulação.

5.4 Cenário de Avaliação

A avaliação de performance foi obtida através do simulador NS-2 e do módulo DiffServ disponível no próprio NS-2. Para que fosse possível avaliar o desempenho do modelo DiffServ foram utilizados dois cenários usando a topologia de rede da empresa *Freedom Network* conforme a Figura 5.3. Está é uma topologia com 16 nós, largura de banda de 3 Gb no enlace principal com limite máximo de enfileiramento de 35000 pacotes e nos demais enlaces largura de banda de 1 Gb e limite máximo de enfileiramento de 30000 pacotes. Os enlaces que ligam o **Core1** ao **Core2** e que estão "diretamente conectados" aos **Core1** e **Core2** são de fibra multimodo. Enquanto que os outros enlaces são cabos Ethernet CAT5E.

Para o cenário usando a solução DiffServ diferenciou-se as classes através do mecanismo de enfileiramento *Weight Round Robin* (WRR) com duas filas de peso 8 e peso 2 usadas para o tráfego VoIP e para os demais tráfegos, respectivamente; policiadores *Token Bucket* (Balde de Fichas) com *Committed Information Rate* (CIR) 72 kbps e *Committed Burst Size* (CBS) 10 kB para o tráfego VoIP, CIR 1 Mbps e CBS 1 KB para o tráfego *Constant bit Rate* e CIR 400 kbps e CBS 1 KB para o tráfego Pareto (Modelos de tráfego definidos na Tabela 5.1). Por fim foram usadas também filas do tipo *Random Early Detection*

(RED) com limite mínimo de 5000 pacotes e limite máximo de 20000 pacotes para as filas que contém o tráfego VoIP e limite mínimo 2500 pacotes e limite máximo de 10000 pacotes para as filas que contém os demais tráfegos.

Para a implementação do cenário *Best-Effort* foram usadas somente filas do tipo *First In First Out* (FIFO). A topologia está organizada da seguinte maneira: dois roteadores de núcleo, chamados de **Core1** e **Core2**. No **Core1** são ligados os roteadores **Borda1** e **Core2**. Ao roteador de **Borda1** estão ligados 3 *switches* chamados de **SwA**, **SwB**, **Sw_Helpdesk**. Ao **Core2** estão ligados 5 roteadores chamados de **Borda2**, **Borda3**, **Borda6**, **Borda4**, **Borda5**. A cada roteador de borda está ligado um *switch*: ao **Borda2** está ligado o **Sw2**, ao **Borda3** está ligado o **Sw3**, ao **Borda4** está ligado o **Sw4** e ao **Borda5** está ligado o **Rot5**.

Para analisar o tráfego VoIP de interesse separadamente e levando em consideração a dinâmica da rede corporativa, inclui-se em cada fluxo VoIP as chamadas realizadas entre os novos prédios, a Central Telefônica e a Central de Atendimento aos Usuários, ou seja, os locais onde estão as aplicações VoIP. Assim, pode-se avaliar os parâmetros de QoS para as aplicações VoIP simuladas definindo 4 fluxos de tráfego VoIP, os quais foram distribuídos da seguinte maneira:

- **Fluxo 1:** Fluxo com 100 fontes VoIP. As origens deste fluxo são os *switches* ligados ao roteador **Core2** e o destino é o *switch* **Sw_Helpdesk**. O monitoramento deste fluxo serve para avaliar os parâmetros de QoS para as aplicações VoIP utilizadas na **Central de Atendimento ao Usuário**;
- **Fluxo 2:** Fluxo com 1000 fontes VoIP. A origem deste fluxo é o *switch* **Sw3** e os destinos são os **Sw2** e **Sw6**. O monitoramento deste Fluxo serve para avaliar os parâmetros de QoS para as aplicações VoIP que realizam chamadas do **prédio2** para o **prédio1** e a **Central Telefônica**;
- **Fluxo 3:** Fluxo com 1000 fontes VoIP. A origem deste fluxo é o *switch* **Sw2** e os destinos são os *switches* **Sw3** e **Sw6**. O monitoramento deste Fluxo serve para avaliar os parâmetros de QoS para as aplicações VoIP que realizam chamadas do **prédio1** para o **prédio2** e a **Central Telefônica**;
- **Fluxo 4:** Fluxo com 2000 fontes VoIP. A origem deste fluxo é o *switch* **Sw6** e os destinos são os *switches* **Sw2** e **Sw3**. O monitoramento deste Fluxo serve para avaliar os parâmetros de QoS para as aplicações VoIP que realizam chamadas da **Central Telefônica** para os **novos prédios**.

Ao fazer o mapeamento da rede corporativa da instituição *Freedom Network*, foram realizadas medições do tráfego e um levantamento dos tipos de tráfego gerados na rede corporativa. Logo, para simular o tráfego gerado na rede observou-se que os tipos de fontes descritos na Tabela 5.1 e criados no cenário das simulações reproduzem o que foi observado do ponto de vista de carga na rede.

Tabela 5.1: Modelagem do Tráfego.

Fonte	**Carga Útil**	**Tempo** *on*	**Tempo** *off*	**Taxa de transmissão**	**Tipo**	**Função**
Constat Bit Rate	1kb	-	-	1 Mbps	Tráfego de fundo	Simular *stream* de áudio e vídeo
Pareto	53 bytes	250 ms	250 ms	400 Kbps	Tráfego de fundo	Simular tráfego FTP e HTTP
Exponential	53 bytes	800 ms	1200ms	72 Kbps	Tráfego VoIP	Simular tráfego VoIP

A fonte *Constant Bit Rate* (CBR) é aplicada a conexões que necessitam de largura de banda fixa. Na simulação desenvolvida, esta fonte foi usada para simular *stream* de áudio e vídeo. A fonte Pareto gera tráfego de acordo com a distribuição Pareto *On/Off*. Pacotes são enviados com tamanho e taxa fixos durante os períodos em que a fonte estiver *On* e nos períodos em que a fonte estiver *Off* nenhum pacote é enviado. Este tipo de fonte é usada para gerar tráfego agregado com *Long Range Dependence* (cauda longa). Por essa característica, é usada para simular servidores de FTP e HTTP. A fonte Exponential gera tráfego de acordo com a distribuição Exponential *On/Off*. Pacotes são enviados com tamanho e taxa fixos durante os períodos em que a fonte estiver *On* e nos períodos em que a fonte estiver *Off* nenhum pacote é enviado. Este tipo de fonte é usada para gerar tráfego agregado com *Short Range Dependence*. Devido a essa característica é usada para simular o tráfego VoIP. A distribuição de fontes de tráfego é apresentada abaixo.

- Os *switches* **SwA**, **SwB**, **Sw2**, **Sw3**, **Sw4** e **Rot5** geram fontes CBR e Pareto para simular tráfego FTP, HTTP, *stream* de áudio e vídeo, os quais são usados como tráfego de fundo. Este tráfego é baseado no protocolo UDP e serve para congestionar a rede e avaliar a eficiência do modelo DiffServ em relação ao *Best-Effort*. Para atingir a volume de tráfego medido na rede da empresa *Freedom Network* foram colocadas 2000 fontes CBR para gerar tráfego nos *swicthes* **SwA**, **Sw2**, **Sw3** e **Rot5** e 2800 fontes Pareto para gerar tráfego nos *switches* **SwB** e **Sw4**;
- Os *switches* **Sw2**, **Sw3** e **Sw6** geram fontes Exponential para simular tráfego VoIP. Este tráfego é baseado no protocolo UDP e serve para avaliar os parâmetros de QoS para o tráfego VoIP e a eficiência do modelo DiffServ em relação ao *Best-Effort*.

Como o objetivo principal deste trabalho é usar o modelo DiffServ para mostrar a melhora na qualidade das ligações VoIP na rede da empresa *Freedom Network* usando o simulador NS-2, colocou-se 100 fontes VoIP no *switch* **Sw_Helpdesk** para simular a Central de Atendimento aos Usuários do serviço disponibilizado. Colocou-se também 1000 fontes VoIP em cada um dos *switches* **Sw2** e **Sw3** para simular a conexão com os novos prédios que serão construidos onde serão implantados 2000 novos telefones VoIP. Por fim, colocou-se 2000 fontes VoIP para simular a saída dos telefones VoIP para a rede de telefônia comum (PSTN), simulando a Central Telefônica existente. Além disso para gerar o tráfego VoIP são usadas variáveis aleatórias com distribuição Exponential.

A distribuição de carga gerada por cada nó foi organizada conforme as características de carga observadas na rede corporativa em questão e mostrada abaixo:

- *Switch* **SwA**: Pode iniciar até 2000 fontes CBR para os *switches* **Sw2** e **Sw3**;
- *Switch* **Sw4**: Pode iniciar até 2000 fontes CBR para o *switch* **SwA** e iniciar até 2800 fontes Pareto para o *switch* **SwB**;
- *Switch* **SwB**: Pode iniciar até 2000 fontes Pareto para o *switch* **Sw4**;
- *Switch* **Sw6**: Pode iniciar até 2000 fontes Exponential para os *switches* **Sw2** e **Sw3** e iniciar até 2000 fontes CBR para o *switch* **SwB**;
- *Switch* **Sw2**: Pode iniciar até 1000 fontes Exponential para os *switches* **Sw3** e **Sw6**, iniciar até 2000 fontes CBR para o *switch* **Sw3** e iniciar até 2800 fontes Pareto para o *switch* **Sw4**;
- *Switch* **Sw3**: Pode iniciar até 1000 fontes Exponential para os *switches* **Sw6** e **Sw2**, iniciar até 2000 fontes CBR para o *switch* **Sw2** e iniciar até 2800 fontes Pareto para o *switch* **Sw4**;
- Roteador **Rot5**: Pode iniciar até 2000 fontes CBR para o *switch* **SwA**.

As fontes Exponential usadas para simular o tráfego de voz são encapsuladas em um cabeçalho RTP e transmitidas a uma taxa de 72 kbps. As fontes CBR são transmitidas a uma taxa de 1 Mbps. As fontes Pareto são usadas para simular o tráfego IP e são transmitidas a uma taxa de 400 kbps. O tempo total de cada simulação foi de 1 hora e 30 minutos. Como o intuito desse trabalho é simular um cenário realista, as sessões foram iniciadas em intervalos de tempo diferentes não havendo sincronia entre fontes. Além disso, foi implementado um mecanismo para controlar o número total de fontes ativas de forma a que se possa controlar a carga a qual a rede está submetida.

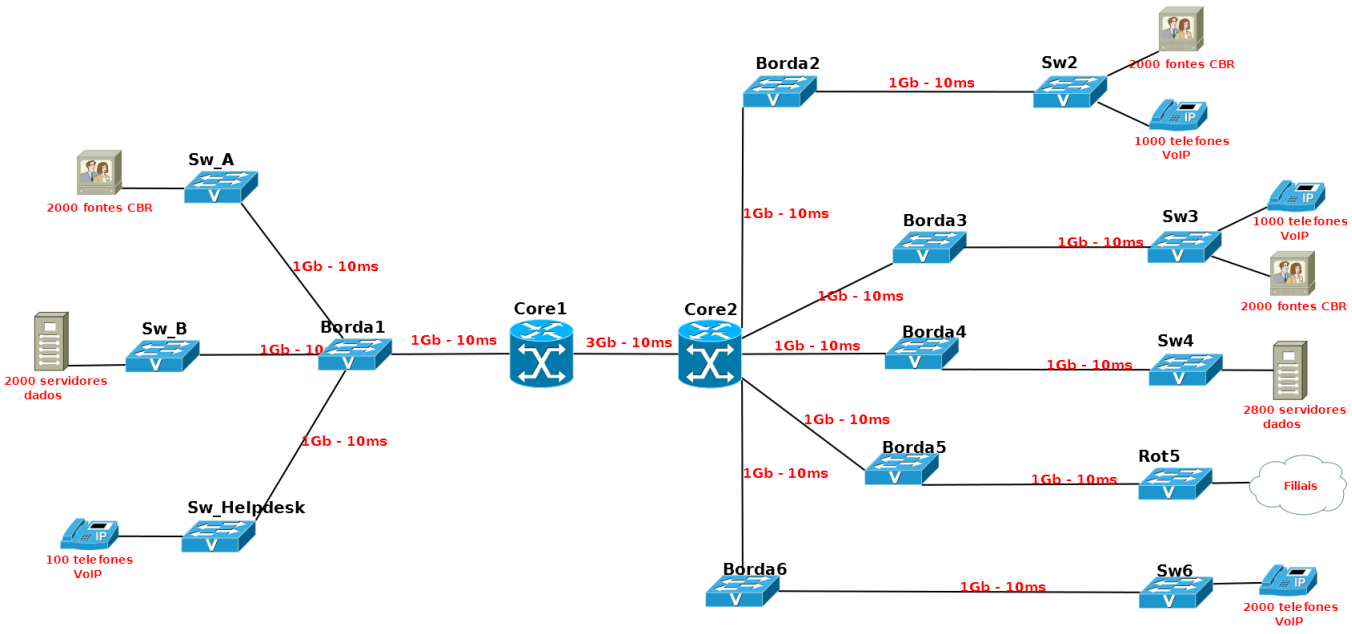

Figura 5.3: Cenário das simulações.

5.5 Resultados Numéricos

Para demonstrar o impacto sobre as fontes VoIP foram feitas análises observando os parâmetros de QoS atraso, perda e *jitter* (Seção 2.2) obtidos pelos scripts apresentados nos Apêndices B e C, os quais geram as métricas: Mínimo, médio, máximo atraso e *jitter*, perda de pacotes. Para gerar os resultados, cinco simulações foram realizadas para cada modelo, utilizando a técnica de replicações independentes. Ao fim de cada simulação o arquivo de traços (**Trace**) é disponibilizado pelo NS-2. O Programa em C que faz a Leitura do Trace (Apêndice A) cria um arquivo para cada fluxo VoIP de interesse (Seção 5.5). Cada um destes arquivos é lido pelo Programa em C para Geração das Métricas (Apêndice B) que calcula: porcentagem de pacotes descartados, mínimo, médio e máximo atraso e *jitter*. Para calcular o desvio padrão do atraso é usado o programa em C para Cálculo do Desvio Padrão (Apêndice C). Como a quantidade total de fontes VoIP do cenário em questão são 2000 fontes, as métricas foram avaliadas com uma variação de carga de 400, 800, 1200, 1600 e 2000 fontes e para cada uma das cargas foi calculada a média do atraso e *jitter*, o desvio padrão do atraso e o descarte de pacotes. Os resultados apresentados apresentam Intervalo de Confiança com confiabilidade de 95% e estão disponíveis nas Figuras 5.4, 5.5, 5.6, 5.7 e 5.8.

Nos dois cenário analisou-se o comportamento do modelo DiffServ e do *Best-Effort* em termos de atraso, perda e *jitter* ocorridos nos enlaces durante todo o tempo da simulação para verificar o ganho de desempenho do modelo DiffServ em relação ao *Best-Effort*.

Comparando-se as métricas obtidas das simulações, evidenciou-se a eficiência do modelo DiffServ em priorizar o tráfego VoIP em relação aos demais tráfegos.

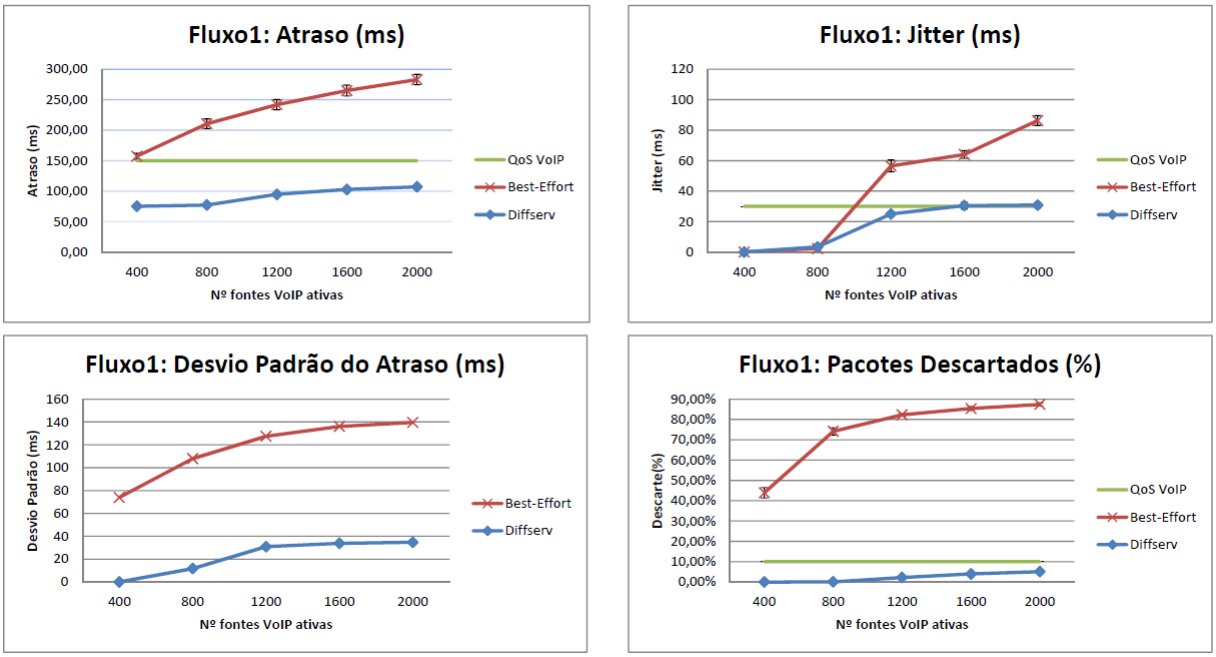

Figura 5.4: Fluxo 1: Atraso, Descarte de pacotes, *Jitter* e Desvio Padrão do Atraso para o *Best-Effort* e o DiffServ.

Na Figura 5.4 são mostrados os parâmetros de QoS atraso, perda e *jitter* para o Fluxo 1 em função do número de fontes VoIP ativas. Logo, como os pacotes VoIP e os pacotes dos outros tráfegos são tratados da mesma maneira no *Best-Effort*, pode-se verificar que o atraso, o *jitter* e a porcentagem de pacotes VoIP descartados para este modelo além de ser bem maior que o aceitável para uma aplicação VoIP conforme mostrado na curva QoS VoIP e discutido na Seção 4.3 também é maior que o atraso, o *jitter* e a porcentagem de pacotes VoIP descartados mostrados pelo modelo DiffServ para este Fluxo. Uma característica deste Fluxo é que ele sofre a menor influência do tráfego VoIP (apenas 100 fontes VoIP ativas) e alta influência do tráfego de fundo, passa pela região mais crítica da rede corporativa onde estão localizados os *switches* Core e passa por uma quantidade maior de enlaces da origem ao destino. Além disso, o atraso de cada enlace tem os maiores valores de atraso do cenário aumentando a quantidade de pacotes enfileirados nas filas dos *switches* e consequentemente aumentando o descarte de pacotes, o atraso e o desvio padrão do atraso o que causa um aumento do *jitter*. Já o modelo DiffServ consegue oferecer uma vazão maior ao tráfego VoIP, embora este fluxo sofra alta influência do tráfego de fundo, devido a uma menor quantidade de pacotes VoIP, mas com alta prioridade no fluxo, fazendo com que os pacotes VoIP sejam transmitidos mais rapidamente que no *Best-Effort* diminuindo a quantidade de pacotes VoIP nos enlaces e consequentemente diminuindo o descarte de pacotes VoIP, o atraso e o desvio padrão do atraso o que causa um menor *jitter*.

Analisando os parâmetros de QoS do Fluxo 1 do modelo DiffServ em relação ao *Best-Effort* verifica-se um ganho de 82% em relação ao descarte de pacotes, 176 ms em relação ao atraso e 55 ms em relação ao *jitter*. Logo, evidencia-se que é impraticável o uso de aplicações VoIP no modelo *Best-Effort* para este Fluxo. Por outro lado, observa-se que o modelo DiffServ consegue manter os parâmetros de QoS em níveis aceitáveis para o tráfego VoIP, evidenciando a eficiência deste modelo sendo o mais indicado como solução para melhorar o tráfego VoIP deste Fluxo.

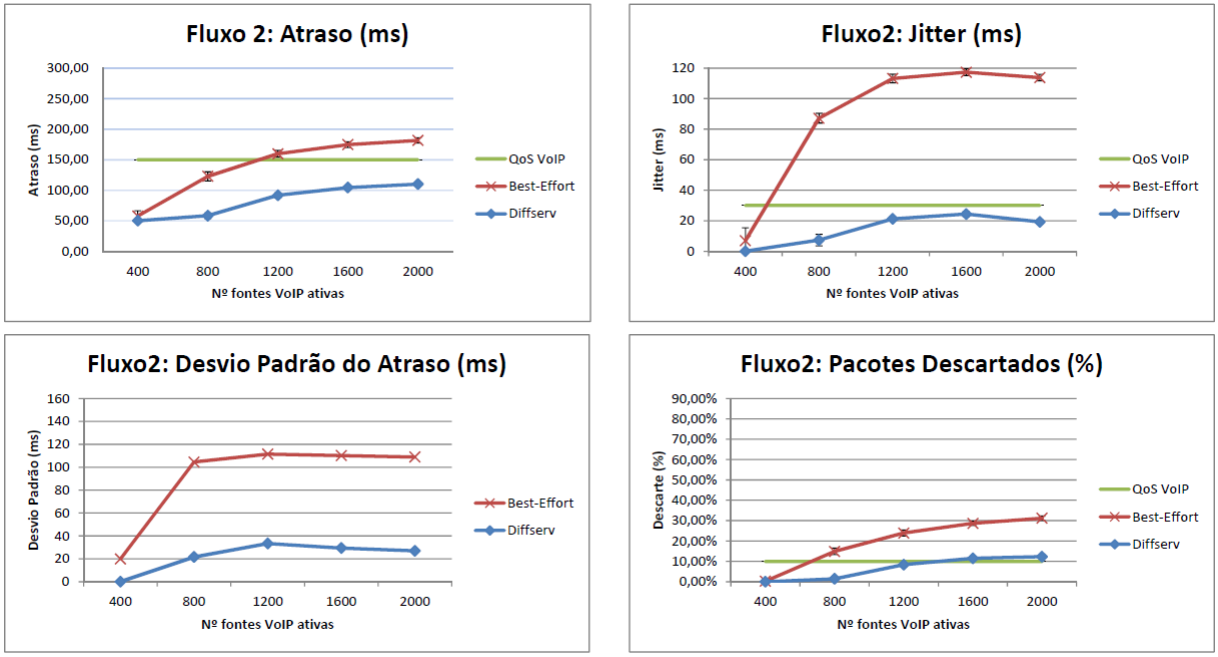

Figura 5.5: Fluxo 2: Atraso, Descarte de pacotes, *Jitter* e Desvio Padrão do Atraso para o *Best-Effort* e o DiffServ.

Na Figura 5.5 são mostrados os parâmetros de QoS atraso, perda e *jitter* para o Fluxo 2 em função do número de fontes VoIP ativas. Logo, como os pacotes VoIP e os pacotes dos outros tráfegos são tratados da mesma maneira no *Best-Effort*, pode-se verificar que o atraso, o *jitter* e a porcentagem de descarte dos pacotes VoIP para este modelo além de ser bem maior que o aceitável para uma aplicação VoIP conforme mostrado na curva QoS VoIP e discutido na Seção 4.3 também é maior que o atraso, o *jitter* e a porcentagem de pacotes VoIP descartados mostrados pelo modelo DiffServ para este Fluxo. Uma característica deste Fluxo é que todos os enlaces possuem o mesmo atraso, mesma largura de banda e sofrem alta influência do tráfego VoIP assim como do tráfego de fundo. Pelo fato do atraso de cada enlace ser igual em todos os enlaces, porém sofrer alta influência do tráfego VoIP e do tráfego de fundo, nota-se que o modelo *Best-Effort*, até 1200 fontes VoIP ativas, consegue manter o atraso em um nível aceitável de QoS, ou seja, abaixo da curva QoS VoIP. Embora o atraso até 1200 fontes VoIP ativas esteja em um nível aceitável, o *jitter* apresenta um valor inaceitável de QoS com pouco mais de 400 fontes VoIP ativas, evidenciando que se torna impraticável o uso de aplicações VoIP no modelo *Best-Effort* para este Fluxo. Este aumento rápido do *jitter* se deve ao fato do aumento rápido do desvio padrão do atraso o qual indica que o atraso está sofrendo altas flutuação consequentemente aumentando o *jitter*. Contudo, observa-se que o atraso e o *jitter* estabilizam somente quando ocorre um aumento da porcentagem de pacotes VoIP descartados diminuindo a quantidade de pacotes VoIP nas filas dos roteadores. Já o modelo DiffServ consegue oferecer uma vazão maior a este Fluxo devido a alta prioridade dos pacotes VoIP, fazendo com que estes pacotes seja transmitido mais rapidamente que no *Best-Effort* diminuindo a quantidade de pacotes VoIP nos enlaces e consequentemente o descarte de pacotes VoIP, o atraso e o desvio padrão do atraso o que também causa um

menor *jitter*. Vale ressaltar que embora o tráfego VoIP tenha alta prioridade, devido a alta influência deste tráfego os valores de atraso, *jitter* e descarte de pacotes VoIP tendem a aumentar podendo até ultrapassar os valores aceitáveis de QoS para VoIP mesmo usando o modelo DiffServ. Este fato é observado neste Fluxo e no Fluxo 3 quando a porcentagem de pacotes VoIP descartados ultrapassa o valor aceitável de QoS acima de 1600 fontes VoIP ativas.

Analisando os parâmetros de QoS do Fluxo 2 do modelo DiffServ em relação ao *Best-Effort* verifica-se um ganho de 17% em relação ao descarte de pacotes, 71,6 ms em relação ao atraso e 93,5 ms em relação ao *jitter*. Logo, conclui-se que o modelo DiffServ consegue manter os parâmetros de QoS em níveis aceitáveis para o tráfego VoIP, evidenciando a eficiência deste modelo para este Fluxo.

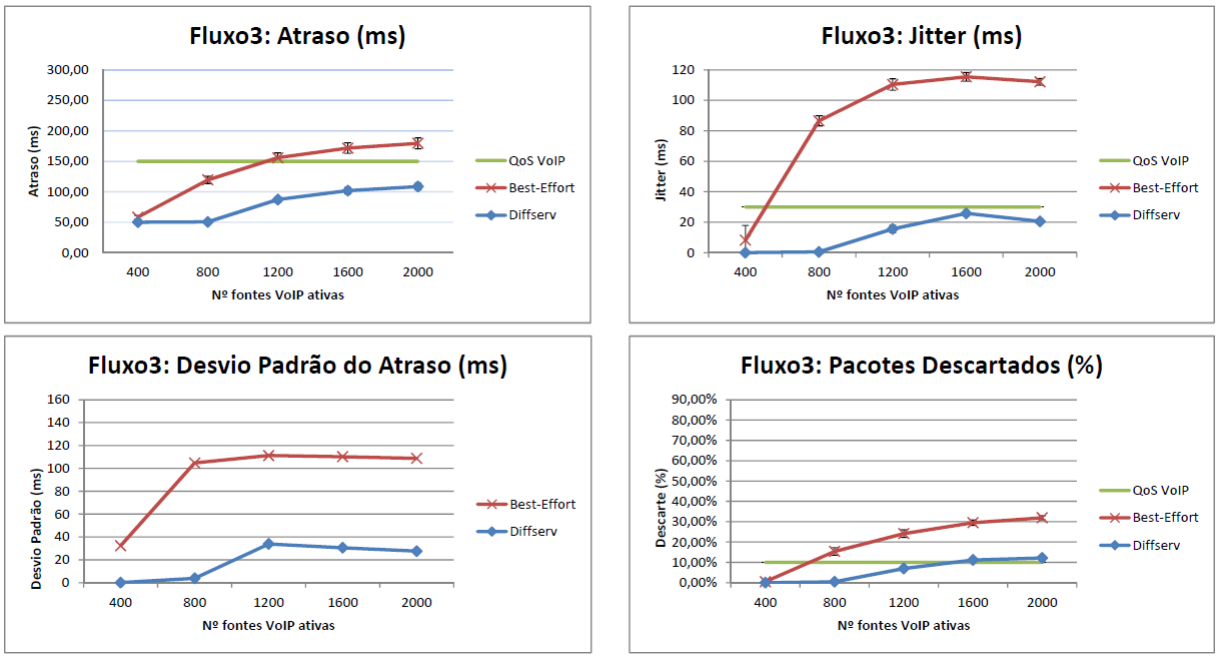

Figura 5.6: Fluxo 3: Atraso, Descarte de pacotes, *Jitter* e Desvio Padrão do Atraso para o *Best-Effort* e o DiffServ.

Na Figura 5.6 são mostrados os parâmetros de QoS atraso, perda e *jitter* para o Fluxo 3 em função do número de fontes VoIP ativas. Logo, como os pacotes VoIP e os pacotes dos outros tráfegos são tratados da mesma maneira no *Best-Effort*, pode-se verificar que o atraso, o *jitter* e a porcentagem de descarte dos pacotes VoIP para este modelo além de ser bem maior que o aceitável para uma aplicação VoIP conforme mostrado na curva QoS VoIP e discutido na Seção 4.3 também é maior que o atraso, o *jitter* e a porcentagem de pacotes VoIP descartados mostrados pelo modelo DiffServ para este Fluxo. Este Fluxo possui as mesmas características do Fluxo 2, ou seja, todos os enlaces possuem o mesmo atraso, mesma largura de banda e sofrem alta influência do tráfego VoIP e do tráfego de fundo. Pelo fato do atraso de cada enlace ser igual em todos os enlaces, porém sofrer alta influência do tráfego VoIP e do tráfego de fundo, nota-se que o modelo *Best-Effort*, até 1200 fontes VoIP ativas, consegue manter o nível de atraso em um nível aceitável de QoS, ou seja, abaixo da curva QoS VoIP. Embora o atraso até 1200 fontes VoIP

ativas esteja em um nível aceitável, o *jitter* apresenta um valor inaceitável de QoS com pouco mais de 400 fontes VoIP ativas, evidenciando que se torna impraticável o uso de aplicações VoIP no *Best-Effort* para este Fluxo. Este aumento rápido do *jitter* se deve ao fato do aumento rápido do desvio padrão do atraso, estabilizando os valores de atraso e *jitter* somente quando ocorre o aumento da porcentagem de pacotes VoIP descartados, conforme observado no Fluxo 2. Já o modelo DiffServ consegue oferecer uma vazão maior ao tráfego VoIP devido a alta prioridade deste, fazendo com que os pacotes VoIP sejam transmitidos mais rapidamente que no *Best-Effort* diminuindo a quantidade de pacotes VoIP nos enlaces e consequentemente o descarte, o atraso e o desvio padrão do atraso o que também causa um menor *jitter*.

Analisando os parâmetros de QoS do Fluxo 3 do modelo DiffServ em relação ao *Best-Effort* verifica-se um ganho de 20% em relação ao descarte de pacotes, 71 ms em relação ao atraso e 91,65 ms em relação ao *jitter* praticamente os mesmos resultados apontados pela análise do Fluxo 2. Estes resultados eram esperados visto que ambos os fluxos possuem as mesmas características. Dessa forma, conclui-se que o modelo DiffServ consegue manter os parâmetros de QoS abaixo dos valores apresentados no *Best-Effort* e em níveis aceitáveis para o tráfego VoIP, evidenciando a eficiência deste modelo para este Fluxo.

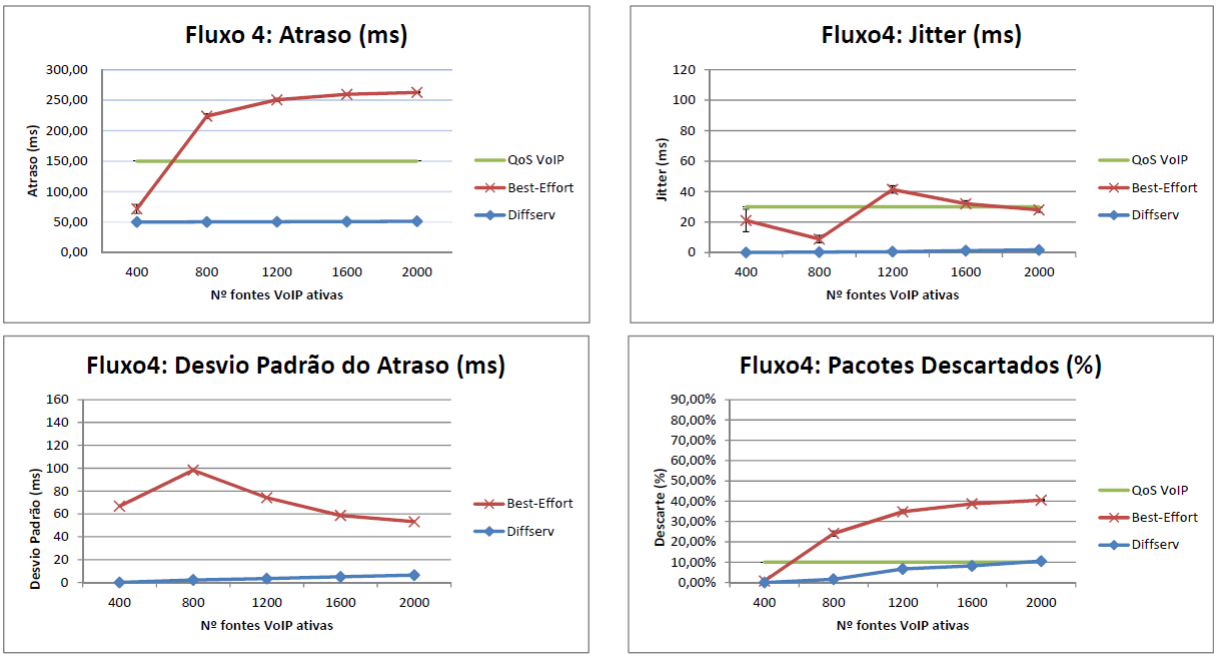

Figura 5.7: Fluxo 4: Atraso, Descarte de pacotes, *Jitter* e Desvio Padrão do Atraso para o *Best-Effort* e o DiffServ.

Na Figura 5.7 são mostrados os parâmetros de QoS atraso, perda e *jitter* para o Fluxo 4 em função do número de fontes VoIP ativas. Logo, como os pacotes VoIP e os pacotes dos outros tráfegos são tratados da mesma maneira no *Best-Effort*, pode-se verificar que a porcentagem de descarte, o atraso e o *jitter* dos pacotes VoIP para este modelo além de ser bem maior que o aceitável para uma aplicação VoIP conforme mostrado na curva QoS VoIP e discutido na Seção 4.3 também é maior que a porcentagem de pacotes descartados, o atraso e o *jitter* mostrados pelo modelo DiffServ para este Fluxo. Uma característica

deste Fluxo é que todos os enlaces possuem o mesmo atraso e a mesma largura de banda. Além disso, sofre influência dos tráfegos de fundo que passam pelos *switches* **Core2**, **Sw2** e **Sw3** e a maior influência do tráfego VoIP (podendo chegar a 2000 fontes VoIP ativas) da rede corporativa. Pelo fato do atraso de cada enlace ser igual em todos os enlaces e sofrer influência dos tráfegos de fundo dos *switches* **Core2**, **Sw2** e **Sw3** nota-se que o modelo *Best-Effort* consegue manter o atraso em um nível aceitável de QoS, ou seja, abaixo da curva QoS VoIP até pouco mais de 400 fontes VoIP ativas. Embora o *jitter* esteja em um nível inaceitável apenas nas proximidades de 1200 fontes VoIP ativas para o modelo *Best-Effort* devido ao desvio padrão do atraso e o descarte de pacotes VoIP estarem aumentando lentamente, verifica-se que o atraso e o descarte dos pacotes VoIP apresentam valores inaceitáveis de QoS com pouco mais de 400 fontes VoIP ativas, inviabilizando o uso de aplicações VoIP no *Best-Effort* aumentado o tempo de atraso dos pacotes VoIP, o desvio padrão do atraso e o descarte de pacotes assim como o *jitter* para este Fluxo. Já o modelo DiffServ consegue oferecer uma vazão maior ao fluxo VoIP devido devido a alta prioridade deste tráfego em todos os *switches* fazendo com que os pacotes VoIP deste Fluxo sejam transmitidos mais rapidamente que no *Best-Effort* diminuindo a quantidade de pacotes VoIP nos enlaces e consequentemente o descarte, o atraso e o desvio padrão do atraso o que também causa um menor *jitter*.

Analisando os parâmetros de QoS do Fluxo 4 do modelo DiffServ em relação ao *Best-Effort* verifica-se um ganho de 30% em relação ao descarte de pacotes, 211 ms em relação ao atraso e 26,2 ms em relação ao *jitter*. Logo, conclui-se que o modelo DiffServ consegue manter os parâmetros de QoS em níveis aceitáveis, evidenciando a eficiência deste modelo para este Fluxo.

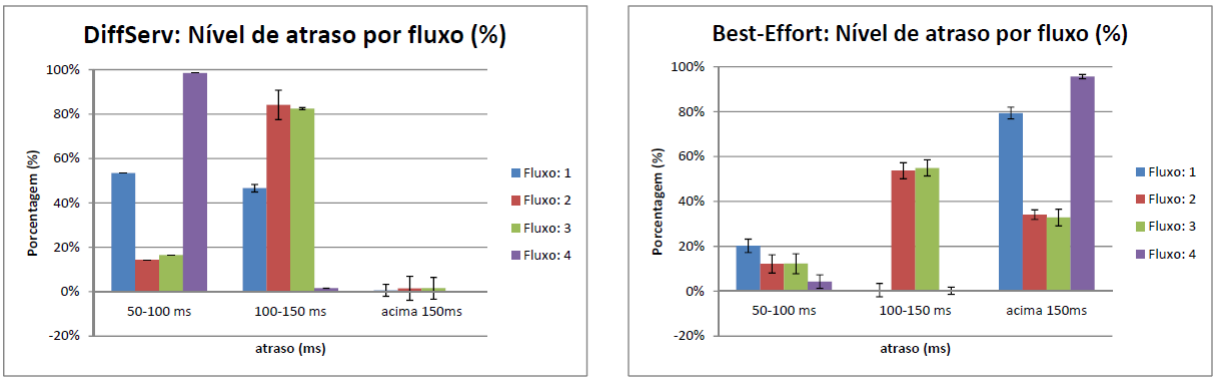

Figura 5.8: Nível de atraso dos pacotes VoIP por Fluxo.

Para melhor visualização do desempenho apresentado pelo modelo DiffServ, na Figura 5.8 é mostrada a porcentagem de pacotes que atendem ou não aos requisitos de atraso para o tráfego VoIP para o modelo DiffServ e para o *Best-Effort*. A porcentagem dos pacotes é dividida em classes de atraso para uma melhor visualização do nível de atraso experimentado por cada fluxo. Verifica-se que mesmo nos nós com a maior quantidade de fontes VoIP, o nível de atraso se mantém em um nível aceitável para o cenário do modelo DiffServ ao contrário do que se observa no desempenho apresentado pelo *Best-Effort* onde a boa parte dos pacotes está em um nível inaceitável evidenciando que este cenário apresenta baixo desempenho para o uso da tecnologia VoIP, enquanto que

o DiffServ apresenta um bom desempenho sendo o mais adequado para todos os fluxos no quesito atraso.

Os gráficos mostrados acima demonstram a eficiência da solução DiffServ em relação ao *Best-Effort* usado na redes corporativas da empresa *Freedom Network*. Nota-se que para todos os fluxos monitorados o modelo DiffServ apresenta os valores de atraso, perda de pacotes e *jitter* dentro dos valores aceitáveis para aplicações VoIP, além de um ganho significativo em relação aos valores apresentados pelo *Best-Effort*. Logo, verifica-se a importância de se ter uma infraestrutura capaz de oferecer suporte QoS às aplicações como VoIP. Assim, pode-se concluir através dos resultados mostrados que a solução DiffServ pode ser aplicada na rede corporativa da empresa *Freedom Network* para oferecer suporte QoS às aplicações VoIP com um excelente ganho de desempenho.

Capítulo 6

Discussão e Conclusão

O modelo *Best-Effort* é um serviço de rede no qual não se provê garantias de entrega para os pacotes dos usuários deste serviço. Assim todos os usuários deste modelo compartilham a largura de banda com todos os fluxos de dados de outros usuários. Para que os pacotes de um fluxo cheguem ao destino, eles utilizam as rotas definidas e a largura de banda que estiver disponível. Quando ocorre congestionamento pacotes são descartados sem distinção.

As características do serviço disponibilizado pelo *Best-Effort* fizeram com que outros modelos de serviço fossem criados, visando melhorar o transporte de pacotes que contém características especiais, como é o caso dos pacotes de aplicações multimídia. Dos modelos propostos, o modelo de Serviços Diferenciados é o mais promissor devido a sua característica de escalabilidade.

Como forma de demonstrar e avaliar o desempenho do modelo DiffServ foram realizadas simulações utilizando o cenário da rede corporativa da empresa *Freedom Network*. Uma simulação foi realizada utilizando o *Best-Effort* e a outra utilizando o modelo DiffServ. São aplicados às simulações o mesmo cenário e a mesma quantidade de fontes de tráfego, assim como os valores de largura de banda e atraso para cada enlace, diferindo no tipo de fila usado: FIFO para o cenário *Best-Effort* e WRR, policiadores *Token Bucket* e mecanismo de Controle de Congestionamento RED para o cenário DiffServ.

Os resultados alcançados demonstram o desempenho superior do modelo DiffServ em todos os parâmatros analisados: atraso, *jitter* e perda de paoctes em relação ao *Best-Effort*, justificando assim o uso do modelo DiffServ como uma solução apropriada de QoS para aplicações VoIP na rede corporativa da empresa *Freedom Network*.

6.1 Desafios da Pesquisa

O método de Serviços Diferenciados é utilizado para conseguir QoS com capacidade para operar sobre grandes volumes de dados. Para tal finalidade, acordos de nível de serviço são necessários para especificar quais classes de tráfego serão servidas, que garantias são necessárias para cada classe e qual o volume de dados para cada classe.

O método Diffserv tem sido o modelo mais usado para implementação de QoS, pois exige menos dos roteadores, provê bons métodos de classificação, policiamento, montagem e remarcação dos pacotes, além da vantagem de que a imposição das políticas é realizada

nas bordas da rede, liberando o núcleo da rede para operar sem a preocupação com a complexidade de contabilização e imposição dos fluxos.

Como principais desafios pode-se citar:

- **Uso do simulador NS-2** - Com o simulador NS-2 pode-se ter a noção do funcionamento da rede estudada fazendo os ajustes necessários incluindo fontes de diversos tipos, filas, políticas de escalonamento, ou seja, uma estrutura com recursos para auxiliar na contrução de cenários de rede futuras ou em fase de mudanças. Dessa maneira, nota-se que a construção de cenários usando o simulador não é algo trivial. É necessário entender o funcionamento da rede física, o funcionamento do simulador e qual a melhor forma de implementar o cenário usando o simulador. Dependendo do tamanho do cenário a ser montado mais hardware e atenção é necessária para que não se tenha nenhum equívoco.
- **Scripts** - Como a saída do simulador NS-2 é um arquivo de log contendo tudo o que ocorreu na simulação, faz-se necessário o uso de scripts a fim de analisar estatisticamente a simulação. Normalmente os scripts são implementados na linguagem Pearl, mas podem ser implementados em qualquer linguagem. Para simulações envolvendo QoS, os scripts devem gerar métricas relacionadas aos parâmetros de QoS: atraso fim-a-fim, *jitter* e perda.
- **Mapeamento da rede** - Para a construção do cenário no simulador, o primeiro passo é entender o funcionamento e características da rede para que se possa fazer um mapeamento eficaz. No mapeamento deverá estar especificado os principais nós da rede, juntamente com os enlaces que os interligam, largura de banda utilizada pelos enlaces, protocolos das camadas de aplicação e transporte usados.
- **Hardware** - O NS-2 é um software capaz de simular grandes redes. Para tanto alguns cuidados devem ser tomados visando diminuir a quantidade de memória usada e de disco, pois quanto maior for o número de nós e fontes utilizadas na simulação maior será o uso do disco para guardar o arquivo de *log* e maior será o uso de memória para comportar o número de fontes ativas em um dado momento.

6.2 Contribuições do Trabalho

Este trabalho faz uso do simulador *Network Simulator 2* e do módulo de Serviços Diferenciados para avaliar o desempenho e o impacto causado em uma rede corporativa usando voz sobre IP com tratamento de QoS. A simulação realizada com o modelo Diffserv permitiu demonstrar suas vantagens em comparação com a simulação que não usava tratamento de QoS para os pacotes. Pode-se notar a redução significativa dos valores de atraso, *jitter* e perda em todos os enlaces, principalmente nos enlaces em que se tem um maior número de fontes de voz. Dessa forma, conclui-se que utilizando o modelo Diffserv na rede corporativa da empresa *Freedom Network* pode-se obter as características necessárias para o funcionamento adequado de aplicações VoIP, sendo assim uma forma de melhorar o tráfego para pacotes de aplicações multimídia em tempo real.

6.3 Trabalhos Futuros

Os resultados deste trabalho podem servir como base para a realização de uma avaliação no ambiente da rede corporativa da empresa *Freedom Network*, permitindo a validação dos mesmos. Acredita-se que os benefícios do uso de aplicações VoIP nessa rede poderá diminuir os custos com ligações telefônicas, com garantias de que o desempenho dessas aplicações em termos de atraso, *jitter* e perda de pacotes esteja dentro dos limites aceitos para conversações VoIP.

Referências

[1] Paulo Henrique Azevêdo Filho. Agregação de pacotes em transmissões voip sobre redes sem fio saturadas. *CEP*, 70910:900, 2010. 26, 28

[2] San Blake, Donald Black, Michael Carlson, Edward Davies, Zhen. Wang, and Wei. Weiss. An architecture for differentiated services. *RFC 2475*, 1998. 18, 20, 21, 22, 23

[3] Richard Braden, Dan Clark, and Sebastian Shenker. Rfc 1633: Integrated services in the internet architecture: an overview. *RFC 1633*, 1994. 5, 14

[4] Lamia Chaari, Mohamed Fourati, Nouri Masmoudi, and Lotfi Kamoun. An adaptive coded modulation with multi-levels qos analysis in multimedia environment. *WSEASTransactions on Communications*, 2009. 4

[5] Yan Chen, Toni Farley, and Nong Ye. Qos requirements of network applications on the internet. *Information, Knowledge, Systems Management*, 4(1):55–76, 2004. 4

[6] H3C Technologies Company. Qos introduction, 2005. Acesso em: 8 abril 2012. `http://www.margalho.pro.br/tutoriais.htm`. vii, 7, 9, 11

[7] H3C Technologies Company. Network simulator-guia básico para iniciantes, 2012. Acesso em: 20 novembro 2013. `http://www.margalho.pro.br/tutoriais.htm`. vii, 40

[8] Bruce Davie, Anna Charny, JCR Bennet, Kent Benson, Jean-Yves Le Boudec, William Courtney, Shahram Davari, Victor Firoiu, and Dimitrios Stiliadis. An expedited forwarding phb (per-hop behavior). *RFC 3246*, 2002. 36

[9] Thays Cristina Costa de Araújo and Márcio Vinıcius de Moura Ribeiro. Avaliação do impacto da implementação de protocolos seguros em um ambiente voip. *CEP*, 70910:900, 2007. 26, 27, 31

[10] Karina Karla C de Oliveira, Djamel Sadok, Judith Kelner, and Obionor Nóbrega. Avaliação de um algoritmo adaptativo de transmissão de voz em redes ip com qos. In *Simpósio Brasileiro de Redes de Computadores*, page 425, 2003. 36

[11] Paul Ferguson and Geoff Huston. *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*. John Wiley & Sons, Inc., 1998. 7, 9

[12] R Frederick, V Jacobson, and Packet Design. Rtp: A transport protocol for real-time applications. *RFC 3550*, 2003. vii, 31, 32, 33, 34

[13] Sérgio Gorender, RJA Macêdo, and WL Pacheco Jr. Controle de admissão para qos em sistemas distribuídos híbridos, tolerantes a falhas. In *Anáis do XI Workshop de Testes e Tolerância a Falhas, WTF2010*, pages 45–58, 2010. 15

[14] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. Springer, 2011. 38

[15] Yunseok Jang, Jin Wook Chung, Ji Han Seo, et al. Design and implementation of sip ua for a manufacturing network. *The International Journal of Advanced Manufacturing Technology*, 28(7-8):822–826, 2006. viii, 30

[16] Carlos Alberto Kamienski and Djamel Sadok. Qualidade de serviço na internet. *minicurso, 18º SBRC, Belo Horizonte/MG*, 2000. 1, 4, 6, 7

[17] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Pearson, São Paulo, trad. 5 ed. edition, 2010. vii, 1, 4, 5, 6, 7, 9, 10, 11, 28, 31, 32, 33, 34, 35

[18] Francois Le Faucheur, L Wu, B Davie, S Davari, P Vaananen, R Krishnan, P Cheval, and J Heinanen. Multi-protocol label switching (mpls) support of differentiated services. *RFC 3270*, 2002. 20, 23

[19] José A.S. Monteiro, Leobino Sampaio, and Mércia Figueredo. Qualidade de serviço: Diagnóstico e alternativas. *RNP*, 2002. 5

[20] Kathleen Nichols, David L Black, Steven Blake, and Fred Baker. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. *RFC 2474*, 1998. 20

[21] Kun I Park. *QoS in packet networks*. Springer, 2005. vii, 8, 15, 17, 19, 21, 22, 23

[22] Lydia Parziale. *TCP/IP Tutorial and Technical Overview*. IBM Redbooks, 2006. 14, 15, 16, 17, 18, 20, 22, 26, 27, 31, 32, 33

[23] Wei Ren. Qos-aware and compromise-resilient key management scheme for heterogeneous wireless internet of things. *International Journal of Network Management*, 21(4):284–299, 2011. 5

[24] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, Eve Schooler, et al. Sip: Session initiation protocol. *RFC 3261*, 2002. viii, 27, 28, 29, 30

[25] Henning Schulzrinne, Anup Rao, and Robert Lanphier. Real time streaming protocol (rtsp). *RFC 2326*, 1998. 34

[26] Scott Shenker. Specification of guaranteed quality of service. *RFC 2212*, 1997. 16

[27] Scott Shenker and John Wroclawski. General characterization parameters for integrated service network elements. *RFC 2215*, 1997. 15

[28] Marcelo Maia Sobral. Redes multimídia, 2012. Acesso em: 28 fevereiro 2013. `http://wiki.sj.ifsc.edu.br/`. vii, 22

[29] Cisco System. Cisco ios quality of service solutions configuration guide, 1999. Acesso em: 10 maio 2012. http://www.cisco.com. 1

[30] Tim Szigeti and Christina Hattingh. Quality of service design overview. *Cisco, San Jose, CA*, 2004. 36

[31] Andrew Tanenbaum. *Redes de Computadores, 4a edição*. Pearson, São Paulo, 2003. 10, 11, 28

[32] Mário Antonio Meireles Teixeira. *Suporte a Serviços Diferenciados em Servidores Web: Modelos e Algoritmos*. PhD thesis, PhD thesis, ICMC-USP, São Carlos-SP, 2004. 4, 12, 16

[33] Zheng Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann, 2001. 17, 23

[34] Stephanie Wood and Samir Chatterjee. Network quality of service for the enterprise: A broad overview. *Information Systems Frontiers*, 4(1):63–84, 2002. 4, 12

[35] John Wroclawski. Specification of the controlled-load network element service. *RFC 2211*, 1997. 16

[36] XiPeng Xiao, Thomas Telkamp, Victoria Fineberg, Cheng Chen, and Lionel M Ni. A practical approach for providing qos in the internet backbone. *Communications Magazine, IEEE*, 40(12):56–62, 2002. 12

[37] Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource reservation protocol (rsvp)-version 1 functional specification. *RFC 2205*, 1997. vii, 17

Apêndice A

Programa em C para Leitura do *Trace*

Este programa recebe como entrada o arquivo de traços (*Trace*) que é disponibilizado pelo NS-2. É responsável por separar o *Trace* por fluxos VoIP criando 1 arquivo para cada fluxo. Esses arquivos são usados pelo Programa em C para Geração das Métricas (Apêndice B).

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <signal.h>
#include <sys/types.h>
#include <unistd.h>

void EscreveSaida(FILE *arqInput){
 char event[1],pkgType[2],flags[6];
 int src,dst,pkgSize,numSeq,idFluxo,auxPkgId;
 double time;
 float endOrg,endDest;
 long int pkgId,linhasEntrada=0,linhasSaida=0;

 FILE *arqSaida,*arqSaida1,*arqSaida2,*arqSaida3,*arqSaida4;
 arqSaida=fopen("./resultados/output5.tr","w");
 arqSaida1=fopen("./resultados/output1.tr","w");
 arqSaida2=fopen("./resultados/output2.tr","w");
 arqSaida3=fopen("./resultados/output3.tr","w");

 while (!feof(arqInput)){
  fscanf(arqInput,"%s %lf %d %d %s %d %s %d %f %f %d %d\n",\
  event,&time,&src,&dst,pkgType,&pkgSize,flags,\
  &idFluxo,&endOrg,&endDest,&numSeq,&auxPkgId);

  if(auxPkgId<0) pkgId=(-1)*auxPkgId;
  else pkgId=auxPkgId;
```

```
  linhasEntrada++;
  if((strcmp(event,"+")==0 || strcmp(event,"r")==0 || strcmp(event,"d")==0)){
   linhasSaida++;
   if(idFluxo==4)fprintf(arqSaida,"%s %lf %d %ld\n",event,time,idFluxo,pkgId);
   if(idFluxo==1)fprintf(arqSaida1,"%s %lf %d %ld\n",event,time,idFluxo,pkgId);
   if(idFluxo==2)fprintf(arqSaida2,"%s %lf %d %ld\n",event,time,idFluxo,pkgId);
   if(idFluxo==3)fprintf(arqSaida3,"%s %lf %d %ld\n",event,time,idFluxo,pkgId);
  }
 }
 fclose(arqSaida);
 fclose(arqSaida1);
 fclose(arqSaida2);
 fclose(arqSaida3);

 printf("\t\tLinhas\n Entrada: %ld --> Saida: %ld\n\n",linhasEntrada,linhasSaida);
}

int main(int argc, char *argv[]){

 FILE *arq;
 arq=fopen(argv[1],"r");
 EscreveSaida(arq);
 fclose(arq);
 return 0;
}
```

Apêndice B

Programa em C para Geração de Métricas

Este programa é responsável por calcular a porcentagem de pacotes descartados, mínimo, médio e máximo atraso e *jitter* para cada fluxo VoIP de interesse. Além disso, cria um arquivo contendo os tempos de chegada dos pacotes ao destino e os atrasos relacionados ao fluxo que é usado pelo Programa em C que calcula o Desvio Padrão do Atraso (Apêndice C). Para isso, recebe como entrada cada um dos arquivos criados pelo Programa em C para Leitura do *Trace* (Apêndice A) contendo o evento ocorrido, tempo em que ocorreu o evento, identificador do fluxo e o identificador de pacote dentro do fluxo.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <unistd.h>

typedef struct nodo{
 double atrasoFluxo;
 double tempoChegada;
 double time;
 long int pkgId;
 int qtdePlus;
 short int fatbal;
 struct nodo *esq, *dir;
}arvoreAVL;

typedef struct statistics{
 double minimo;
 double medio;
 double maximo;
 double qtde;
}pkgStats;

typedef struct SJitter{
```

```
 double atrasoFluxo;
 struct SJitter *prox;
}SJitter;

char event[1];
int linhas=0;
int idFluxo;
long int pkgId;
double recebidos=0.0,dropados=0.0;
double time;
SJitter *epinicio=NULL;
SJitter *listJitter=NULL;
pkgStats atraso;

FILE *arqAtraso1,*arqAtraso2,*arqAtraso3,*arqAtraso4,*arqMedias;

void InicializaAVL(arvoreAVL **eainicio){
 *eainicio = malloc(sizeof (arvoreAVL));
 (*eainicio)->qtdePlus=-1;
 (*eainicio)->esq = NULL;
 (*eainicio)->dir = NULL;

 atraso.medio=0.0;
 atraso.minimo=5000.0;
 atraso.maximo=0.0;
 atraso.qtde=0.0;

 arqAtraso1=fopen("./resultados/atraso/atraso1.txt","w");
 arqAtraso2=fopen("./resultados/atraso/atraso2.txt","w");
 arqAtraso3=fopen("./resultados/atraso/atraso3.txt","w");
 arqAtraso4=fopen("./resultados/atraso/atraso4.txt","w");

 arqMedias=fopen("./resultados/medias/medias.txt","a");
}

arvoreAVL destroi(arvoreAVL **a){
if (a!=NULL){
 destroi((a)->esq);
 destroi((a)->dir);
 free(*a);
}
*a=NULL;
}

void InsereAVL(arvoreAVL *adesc,FILE *arq, double tempo){
arvoreAVL *paux, *pant, *pP, *pQ, *pantP, *pnovo;
```

```
int poschave;
int achou;

 void RotacaoSimples(){
  if (pP->fatbal == 1){
   pP->dir = pQ->esq;
   pQ->esq = pP;
  }
  else{
   pP->esq = pQ->dir;
   pQ->dir = pP;
  }
  paux = pQ;
  pP->fatbal = 0;
  pQ->fatbal = 0;
 }

 void RotacaoDupla (){
  if (pP->fatbal == 1){
   paux = pQ->esq;
   pQ->esq = paux->dir;
   paux->dir = pQ;
   pP->dir = paux->esq;
   paux->esq = pP;
  }
  else {
   paux = pQ->dir;
   pQ->dir = paux->esq;
   paux->esq = pQ;
   pP->esq = paux->dir;
   paux->dir = pP;
  }
  if (paux->fatbal == -poschave){
   pP->fatbal = 0;
   pQ->fatbal = poschave;
}
  else
 if (paux->fatbal == 0){
    pP->fatbal = 0;
    pQ->fatbal = 0;
 }
   else {
    pP->fatbal = -poschave;
    pQ->fatbal = 0;
 }
   paux->fatbal = 0;
```

```
}

void AjustaFatoresAVL(){
 if (pkgId < pP->pkgId){
  pQ = pP->esq;
  paux = pP->esq;
 }
 else{
  pQ = pP->dir;
  paux = pP->dir;
 }
 while (paux->pkgId != pkgId)
  if (pkgId < paux->pkgId){
   paux->fatbal = paux->fatbal - 1;
   paux = paux->esq;
  }
  else{
   paux->fatbal = paux->fatbal + 1;
   paux = paux->dir;
 }
}

void BalanceiaAVL(){
 if (pkgId < pP->pkgId) poschave = -1;
 else poschave = 1;
  if (pP->fatbal == 0) pP->fatbal = poschave;
  else
   if (pP->fatbal == -poschave) pP->fatbal = 0;
   else {
    if (pQ->fatbal * poschave > 0) RotacaoSimples();
    else RotacaoDupla();
     if (pantP->dir == pP) pantP->dir = paux;
      else pantP->esq = paux;
   }
}

while (!feof(arq)){
 fscanf(arq,"%s %lf %d %ld\n",event,&time,&idFluxo,&pkgId);
 linhas++;
 if(time<tempo){
  paux = adesc->dir;
  pP = paux;
  pant = adesc;
  pantP = adesc;
  achou = 0;
  while ((!achou) && (paux != NULL)){
```

```
    if (paux->fatbal != 0){
     pP = paux;
     pantP = pant;
    }
    pant = paux;
    if (pkgId == paux->pkgId) achou = 1;
else
     if (pkgId < paux->pkgId) paux = paux->esq;
 else paux = paux->dir;
   }
   if(achou){
    if(strcmp(event,"+")==0){
     paux->time=(-1)*time;
     paux->qtdePlus++;
    }
    else{
     if(strcmp(event,"r")==0){
      paux->atrasoFluxo+=((paux->time)+time);
      paux->qtdePlus--;
      paux->tempoChegada=time;
     }
     else{
      if(strcmp(event,"d")==0){
       dropados++;
       paux->qtdePlus=-3;
      }
     }
    }
   }
 else{
    if (strcmp(event,"+")==0){
     recebidos++;
     pnovo = malloc (sizeof (arvoreAVL));
     pnovo->pkgId = pkgId;
     pnovo->atrasoFluxo=0;
     pnovo->time=(-1)*time;
     pnovo->tempoChegada=0;
     pnovo->qtdePlus=1;
     pnovo->esq = NULL;
     pnovo->dir = NULL;
     pnovo->fatbal = 0;
    }
    if (adesc->dir == NULL) adesc->dir = pnovo;
    else{
     if (pkgId < pant->pkgId) pant->esq = pnovo;
     else pant->dir = pnovo;
```

```
     AjustaFatoresAVL();
     BalanceiaAVL();
    }
   }
  }
 }
}

SJitter **imprimi(arvoreAVL **eainicio){
if((*eainicio)!=NULL){
 imprimi(&(*eainicio)->esq);
 if((*eainicio)->qtdePlus==0){
  listJitter=malloc(sizeof(SJitter));
  listJitter->atrasoFluxo=(*eainicio)->atrasoFluxo;
  listJitter->prox=epinicio;
  epinicio=listJitter;

  (atraso.qtde)++;
  atraso.medio+=(*eainicio)->atrasoFluxo;
  if(atraso.minimo>(*eainicio)->atrasoFluxo)\
  atraso.minimo=(*eainicio)->atrasoFluxo;
  if(atraso.maximo < (*eainicio)->atrasoFluxo)\
  atraso.maximo=(*eainicio)->atrasoFluxo;

  if (idFluxo==1) fprintf(arqAtraso1,"%lf %lf\n",\
  (*eainicio)->tempoChegada,(*eainicio)->atrasoFluxo);
  else{
   if (idFluxo==2) fprintf(arqAtraso2,"%lf %lf\n",\
   (*eainicio)->tempoChegada,(*eainicio)->atrasoFluxo);
   else{
    if (idFluxo==3) fprintf(arqAtraso3,"%lf %lf\n",\
    (*eainicio)->tempoChegada,(*eainicio)->atrasoFluxo);
    else{
     if (idFluxo==4) fprintf(arqAtraso4,"%lf %lf\n",\
     (*eainicio)->tempoChegada,(*eainicio)->atrasoFluxo);
     else{
      if (idFluxo==5) fprintf(arqAtraso5,"%lf %lf\n",\
      (*eainicio)->tempoChegada,(*eainicio)->atrasoFluxo);
   }
  }
 }
}
}
imprimi(&(*eainicio)->dir);
}
return &epinicio;
```

```
}

void imprimiJitter(SJitter **a){
float jitterFluxos=0;
SJitter *aux;
pkgStats jitter;
jitter.minimo=1.0,jitter.medio=0.0,jitter.maximo=0.0,jitter.qtde=0.0;
while(*a!=NULL){
 if ((*a)->prox!=NULL){
  (jitter.qtde)++;
  jitterFluxos=(*a)->prox->atrasoFluxo-(*a)->atrasoFluxo;
  if(jitterFluxos<0)jitterFluxos*=(-1);
  jitter.medio+=jitterFluxos;
  if(jitter.minimo>jitterFluxos) jitter.minimo=jitterFluxos;
  if(jitter.maximo<jitterFluxos) jitter.maximo=jitterFluxos;
 }
 aux = (*a);
 (*a)=(*a)->prox;
 free(aux);
}
printf("\t\t\nfluxo: %d\n\nminimoAtraso: %lfms\tminimoJitter: %lfms\n\
medioAtraso: %lfms\tmedioJitter: %lfms\nmaximoAtraso: %lfms\t\
maximoJitter: %lfms\n\nRecebidos: %.0lf\t Dropados: %.0lf\t\
Perda: %3.2lf%\n\n",idFluxo,atraso.minimo*1000,jitter.minimo*1000,
(atraso.medio/atraso.qtde)*1000,(jitter.medio/jitter.qtde)*1000,\
atraso.maximo*1000,(jitter.maximo)*1000,\
recebidos,dropados,(dropados/recebidos)*100);

fprintf(arqMedias,"\t\t\nfluxo: %d\n\nminimoAtraso: %lfms\t\
minimoJitter: %lfms\n\
medioAtraso: %lfms\tmedioJitter: %lfms\nmaximoAtraso: %lfms\t\
maximoJitter: %lfms\n\nRecebidos: %.0lf\t Dropados: %.0lf\t\
Perda: %3.2lf%\n\n",idFluxo,atraso.minimo*1000,jitter.minimo*1000,
(atraso.medio/atraso.qtde)*1000,(jitter.medio/jitter.qtde)*1000,\
atraso.maximo*1000,(jitter.maximo)*1000,\
recebidos,dropados,(dropados/recebidos)*100);
}

//-----------------------------------------
int main (int argc, char *argv[]){
arvoreAVL *aini;
SJitter *a;

FILE *arq;
arq=fopen(argv[1],"r");
InicializaAVL(&aini);
```

```
InsereAVL(aini,arq,atof(argv[2]));
fclose(arq);
a=*(imprimi(&aini));
destroi(aini);
imprimiJitter(&a);

return 0;
}
```

Apêndice C

Programa em C para Cálculo do Desvio Padrão do Atraso

Este programa é responsável por calcular a porcentagem de pacotes dentro das classes de atraso: 0-50 ms, 50-100 ms, 100-150 ms e acima de 150 ms. É responsável também pelo cálculo do desvio padrão do atraso. Para tal finalidade recebe como entrada um arquivo que é criado pelo Programa em C para Geração das Métricas (Apêndice B) para cada fluxo VoIP de interesse e que contém os tempos de chegada dos pacotes ao destino e os atrasos relacionados ao fluxo. Como Saída obtém-se a quantidade de linhas lidas do arquivo, a variância, média e o desvio padrão do atraso e a porcentagem de pacotes dentro das classes de atraso: 0-50ms, 50-100ms, 100-150ms e acima de 150ms. O desvio padrão do atraso é calculado usando as fórmulas:

$$media = \frac{\sum atraso}{(linhas - 1)}$$

$$variancia = \sum \frac{(atraso - media)^2}{(linhas - 1)}$$

$$DesvioPadrao = \sqrt{variancia}$$

.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

void EscreveSaida(FILE *arqInput){
double time, atraso;

double linhas=0, soma=0, media=0, var=0, desviop=0;

double aceitavel50=0,aceitavel100=0,aceitavel150=0, naoaceitavel=0;

while (!feof(arqInput)){
 fscanf(arqInput,"%lf %lf\n",&time,&atraso);
 linhas++;
 media+=atraso;
```

```
 if(atraso >= 0.050){
  if(atraso>=0.1){
 if(atraso <= 0.15) aceitavel150++;
}
 else aceitavel100++;
 }
  else if(atraso<0.050) aceitavel50++;
  if(atraso>0.15) naoaceitavel++;
}

media/=(linhas-1);

printf("linhas: %.0lf\t aceitavel 50: %3.2lf%\t aceitavel 100: %3.3lf%\t \
aceitavel 150: %3.3lf%\t aceitavel: %3.3lf%\t NaoAceitavel: %3.3lf%\n",
linhas,(aceitavel50/linhas)*100,(aceitavel100/linhas)*100,(aceitavel150/linhas)*100,\
((aceitavel50+aceitavel100+aceitavel150)/linhas)*100,(naoaceitavel/linhas)*100);

rewind(arqInput);

while (!feof(arqInput)){
 fscanf(arqInput,"%lf %lf\n",&time,&atraso);
 var = (var + (pow(atraso - media,2))/(linhas-1));
 desviop = sqrt(var);
}
printf("Desvio padrao: %lf\t Variancia: %lf\t media: %lf\n",desviop*1000,var*1000,media*1000);
fclose(arqInput);
}

int main(int argc, char *argv[]){

FILE *arq;
arq=fopen(argv[1],"r");
EscreveSaida(arq);
return 0;
}
```

Apêndice D

Script TCL para o NS-2: Módulo de Configuração

```
##################
### CONSTANTES ###
##################
set max_voip_helpdesk 100
set max_voip_sw2 1000
set max_voip_sw3 1000
set max_voip_sw6 2000
set max_dados 2000
set max_dados2 2800
set max_video 2000
set max_video2 2000
set totalTime 4800
set setupWindow 5
set total 0
set qlim 30000
set qlim_core 35000
set cbs0 10000; #-> voz
set cbs1 1000; #-> dados
set cbs2 1000; #-> video
```

Apêndice E

Script TCL para o NS-2: Módulo de Criação de Fontes

```
########################
# Variaveis aleatorias #
########################

#Seeds
for {set i 1} {$i <= 6} {incr i} {
 set rng($i) [new RNG]
 $rng($i) seed 0
}

# VOICE
set distChegadaVoice [new RandomVariable/Exponential]
$distChegadaVoice use-rng $rng(1)
$distChegadaVoice set avg_ 400ms
$distChegadaVoice set avg_ 150s

set distDurVoice [new RandomVariable/Exponential]
$distDurVoice use-rng $rng(2)
$distDurVoice set avg_ 300s
$distDurVoice set avg_ 1000s

# DATA
set distChegadaData [new RandomVariable/Exponential]
$distChegadaData use-rng $rng(3)
$distChegadaData set avg_ 400ms

set distDurData [new RandomVariable/Exponential]
$distDurData use-rng $rng(4)
$distDurData set avg_ 300s
```

```
# VIDEO
set distChegadaVideo [new RandomVariable/Exponential]
$distChegadaVideo use-rng $rng(5)
$distChegadaVideo set avg_ 1s

set distDurVideo [new RandomVariable/Exponential]
$distDurVideo use-rng $rng(6)
$distDurVideo set avg_ 1000s

###############################
# Contadores de Fontes ativas #
###############################

set fontesVozAtivas($sw_helpdesk) 0
set fontesVozAtivas($sw2) 0
set fontesVozAtivas($sw3) 0
set fontesVozAtivas($sw6) 0
set fontesVideoAtivas($swA) 0
set fontesVideoAtivas($sw3) 0
set fontesVideoAtivas($sw2) 0
set fontesVideoAtivas($swB) 0
set fontesDadosAtivas($sw4) 0
set fontesDadosAtivas($swB) 0
set fontesVozAtivas($total) 0
set fontesVozAtivas($sw2,$sw_helpdesk) 0
set fontesVozAtivas($sw2,$sw3) 0
set fontesVozAtivas($sw2,$sw6) 0
set fontesVozAtivas($sw3,$sw2) 0
set fontesVozAtivas($sw3,$sw6) 0
set fontesVozAtivas($sw3,$sw_helpdesk) 0
set fontesVozAtivas($sw6,$sw_helpdesk) 0
set fontesVozAtivas($sw6,$sw2) 0
set fontesVozAtivas($sw6,$sw3) 0
set fontesVideoAtivas($rot5,$swA) 0
set fontesVideoAtivas($sw4,$swA) 0
set fontesVideoAtivas($sw2,$sw3) 0
set fontesVideoAtivas($sw3,$sw2) 0
set fontesVideoAtivas($swA,$sw3) 0
set fontesVideoAtivas($swA,$sw2) 0
set fontesVideoAtivas($sw6,$swB) 0
set fontesDadosAtivas($swB,$sw4) 0
set fontesDadosAtivas($sw4,$swB) 0
set fontesDadosAtivas($sw3,$sw4) 0
set fontesDadosAtivas($sw2,$sw4) 0
```

```
############################
##### CRIA FONTES VOIP #####
############################

cria_fontes_voip $max_voip_helpdesk $sw6 $sw_helpdesk 2
cria_fontes_voip $max_voip_sw2 $sw6 $sw2 5
cria_fontes_voip $max_voip_sw3 $sw6 $sw3 5
cria_fontes_voip $max_voip_helpdesk $sw2 $sw_helpdesk 2
cria_fontes_voip $max_voip_sw2 $sw2 $sw6 4
cria_fontes_voip $max_voip_sw2 $sw2 $sw3 4
cria_fontes_voip $max_voip_sw3 $sw3 $sw6 3
cria_fontes_voip $max_voip_sw3 $sw3 $sw2 3
cria_fontes_voip $max_voip_helpdesk $sw3 $sw_helpdesk 2

############################
##### CRIA FONTES VIDEO ###
############################

cria_fontes_video $max_video $rot5 $swA
cria_fontes_video $max_video $sw4 $swA
cria_fontes_video $max_video2 $sw2 $sw3
cria_fontes_video $max_video2 $sw3 $sw2
cria_fontes_video $max_video $swA $sw2
cria_fontes_video $max_video $swA $sw3
cria_fontes_video $max_video $sw6 $swB

############################
##### CRIA FONTES DADOS ####
############################

cria_fontes_data $max_dados $swB $sw4
cria_fontes_data $max_dados $sw4 $swB
cria_fontes_data $max_dados2 $sw3 $sw4
cria_fontes_data $max_dados2 $sw2 $sw4
```

Apêndice F

Script TCL para o NS-2: Cenário *Best-Effort*

```
######################
### CRIACAO DE NOS ###
######################

set swB [$ns node]
set sw_helpdesk [$ns node]
set swA [$ns node]
set borda1 [$ns node]
set core1 [$ns node]
set core2 [$ns node]
set borda6 [$ns node]
set sw6 [$ns node]
set rot5 [$ns node]
set borda5 [$ns node]
set sw3 [$ns node]
set borda3 [$ns node]
set sw4 [$ns node]
set borda4 [$ns node]
set sw2 [$ns node]
set borda2 [$ns node]

########################
### CRIACAO DE LINKS ###
########################

$ns duplex-link $core1 $core2 3Gb 25ms DropTail
$ns duplex-link $borda2 $sw2 1Gb 10ms DropTail
$ns duplex-link $borda3 $sw3 1Gb 10ms DropTail
$ns duplex-link $borda4 $sw4 1Gb 10ms DropTail
$ns duplex-link $borda6 $sw6 1Gb 10ms DropTail
$ns duplex-link $borda5 $rot5 1Gb 15ms DropTail
```

```
$ns duplex-link $core1 $borda1 1Gb 15ms DropTail
$ns duplex-link $core2 $borda2 1Gb 15ms DropTail
$ns duplex-link $core2 $borda3 1Gb 15ms DropTail
$ns duplex-link $core2 $borda6 1Gb 15ms DropTail
$ns duplex-link $core2 $borda4 1Gb 15ms DropTail
$ns duplex-link $core2 $borda5 1Gb 15ms DropTail
$ns duplex-link $borda1 $swB 1Gb 10ms DropTail
$ns duplex-link $borda1 $swA 1Gb 10ms DropTail
$ns duplex-link $borda1 $sw_helpdesk 1Gb 10ms DropTail

#TAMANHO DAS FILAS
$ns queue-limit $core1 $core2 $qlim
$ns queue-limit $core2 $core1 $qlim
$ns queue-limit $borda2 $sw2 $qlim
$ns queue-limit $borda3 $sw3 $qlim
$ns queue-limit $borda4 $sw4 $qlim
$ns queue-limit $borda6 $sw6 $qlim
$ns queue-limit $borda5 $rot5 $qlim
$ns queue-limit $core1 $borda1 $qlim
$ns queue-limit $borda1 $core1 $qlim
$ns queue-limit $core2 $borda2 $qlim
$ns queue-limit $borda2 $core2 $qlim
$ns queue-limit $core2 $borda3 $qlim
$ns queue-limit $borda3 $core2 $qlim
$ns queue-limit $core2 $borda6 $qlim
$ns queue-limit $borda6 $core2 $qlim
$ns queue-limit $core2 $borda4 $qlim
$ns queue-limit $borda4 $core2 $qlim
$ns queue-limit $core2 $borda5 $qlim
$ns queue-limit $borda5 $core2 $qlim
$ns queue-limit $borda1 $swB $qlim
$ns queue-limit $swB $borda1 $qlim
$ns queue-limit $borda1 $swA $qlim
$ns queue-limit $swA $borda1 $qlim
$ns queue-limit $borda1 $sw_helpdesk $qlim
$ns queue-limit $sw_helpdesk $borda1 $qlim
```

Apêndice G

Script TCL para o NS-2: Gerador de Tráfego

```
############## TODOS OS PROCEDIMENTOS PARA CRIAR TRAFEGO ############################

proc cria_fontes_voip {qtde orig dest fid} {

global ns i j src_voip_ida src_voip_volta sink_dst_voip_ida \
fonte_voice taxa_voip

###########TRANSPORTE##############
 for {set i 1} {$i <= $qtde} {incr i} {
 #TRAFEGO VOIP
  set src_voip_ida($orig,$dest,$i) [new Agent/RTP]
  $ns attach-agent $orig $src_voip_ida($orig,$dest,$i)
  $src_voip_ida($orig,$dest,$i) set fid_ $fid
  $src_voip_ida($orig,$dest,$i) set class_ $fid
  set sink_dst_voip_ida($orig,$dest,$i) [new Agent/Null]
  $ns attach-agent $dest $sink_dst_voip_ida($orig,$dest,$i)
  $ns connect $src_voip_ida($orig,$dest,$i) $sink_dst_voip_ida($orig,$dest,$i)

#########APLICACAO##########

  #TRAFEGO VOIP
  set fonte_voice($orig,$dest,$i) [new Application/Traffic/Exponential]
  $fonte_voice($orig,$dest,$i) attach-agent $src_voip_ida($orig,$dest,$i)
  $fonte_voice($orig,$dest,$i) set packetSize_ 53B
  $fonte_voice($orig,$dest,$i) set burst_time_ 800ms
  $fonte_voice($orig,$dest,$i) set idle_time_ 1200ms
  $fonte_voice($orig,$dest,$i) set rate_ 72000
  $fonte_voice($orig,$dest,$i) set codePt_ 46
 }
}

########### VIDEO ########################################

proc cria_fontes_video {max_video orig dest} {

global ns i src_video_ida sink_dst_video_ida src_video_volta \
```

```
dst_video_volta fonte_video size_video taxa_video

 for {set i 1} {$i <= $max_video} {incr i} {
  #TRAFEGO Video
  set src_video_ida($orig,$dest,$i) [new Agent/UDP]
  $ns attach-agent $orig $src_video_ida($orig,$dest,$i)
  $src_video_ida($orig,$dest,$i) set class_ 6
  set sink_dst_video_ida($orig,$dest,$i) [new Agent/Null]
  $ns attach-agent $dest $sink_dst_video_ida($orig,$dest,$i)
  $ns connect $src_video_ida($orig,$dest,$i) $sink_dst_video_ida($orig,$dest,$i)
  #### APLICACAO
set fonte_video($orig,$dest,$i) [new Application/Traffic/CBR]
  $fonte_video($orig,$dest,$i) attach-agent $src_video_ida($orig,$dest,$i)
  $fonte_video($orig,$dest,$i) set packetSize_ 1000
  $fonte_video($orig,$dest,$i) set rate_ 1000000
  $fonte_video($orig,$dest,$i) set random_ false
  $fonte_video($orig,$dest,$i) set codePt_ 0
 }
}
############### Fontes de Dados

proc cria_fontes_data {max_fontes orig dest} {

 global ns i src_dados_ida sink_dst_dados_ida fonte_data taxa_dados

 for {set i 1} {$i <= $max_fontes} {incr i} {
 #### TRAFEGO DADOS
  set src_dados_ida($orig,$dest,$i) [new Agent/UDP]
  $ns attach-agent $orig $src_dados_ida($orig,$dest,$i)
  $src_dados_ida($orig,$dest,$i) set class_ 6
  set sink_dst_dados_ida($orig,$dest,$i) [new Agent/Null]
  $ns attach-agent $dest $sink_dst_dados_ida($orig,$dest,$i)
  $ns connect $src_dados_ida($orig,$dest,$i) $sink_dst_dados_ida($orig,$dest,$i)
 #### APLICACAO
set fonte_data($orig,$dest,$i) [new Application/Traffic/Pareto]
$fonte_data($orig,$dest,$i) attach-agent $src_dados_ida($orig,$dest,$i)
$fonte_data($orig,$dest,$i) set packetSize_ 53B
$fonte_data($orig,$dest,$i) set burst_time_ 250ms
$fonte_data($orig,$dest,$i) set idle_time_ 250ms
$fonte_data($orig,$dest,$i) set rate_ 400000
  $fonte_data($orig,$dest,$i) set shape_ 1.5
$fonte_data($orig,$dest,$i) set codePt_ 0
 }
}
```

Apêndice H

Script TCL para o NS-2: Módulo de Inicialização de Fontes

```
###################################
##### PROCEDIMENTO FONTE VOIP #####
###################################

proc FonteVoz {time orig dest num max_fontes} {
 global ns distChegadaVoice distDurVoice fonte_voice fontesVozAtivas total

 if {$fontesVozAtivas($dest) < $max_fontes} {
  if {! [$fonte_voice($orig,$dest,$num) set running_]} {
   set final [expr $time + [$distDurVoice value]]
   #START
   $fonte_voice($orig,$dest,$num) start
 #$fonte_voice($dest,$orig,$num) start
 incr fontesVozAtivas($orig,$dest)
 incr fontesVozAtivas($dest)
 incr fontesVozAtivas($total)
 #STOP
 #$ns at $final "$fonte_voice($orig,$dest,$num) stop; incr fontesVozAtivas($orig,$dest) -1;\
 incr fontesVozAtivas($dest) -1; incr fontesVozAtivas($total) -1"
 #$fonte_voice($dest,$orig,$num) stop;
  }
 }
  if {$num >= $max_fontes} {set num 0}
  set prox_inicio [expr $time + [$distChegadaVoice value]/10]
  $ns at $prox_inicio "FonteVoz $prox_inicio $orig $dest [expr $num + 1] $max_fontes"
}

###################################
##### PROCEDIMENTO VIDEO CONF #####
###################################

#args: time origem destino cont
proc FonteVideo {time orig dest num max_video} {
 global ns distChegadaVideo distDurVideo fonte_video fontesVideoAtivas
 if {$fontesVideoAtivas($dest) < $max_video } {
  if {! [$fonte_video($orig,$dest,$num) set running_]} {
   set final [expr $time + [$distDurVideo value]]
 #START
```

```
 $fonte_video($orig,$dest,$num) start
 incr fontesVideoAtivas($orig,$dest)
 incr fontesVideoAtivas($dest)
 #STOP
 $ns at $final "$fonte_video($orig,$dest,$num) stop;\
 incr fontesVideoAtivas($orig,$dest) -1; incr fontesVideoAtivas($dest) -1"
}
 }
 if {$num >= $max_video} {set num 0}
 set prox_inicio [expr $time + [$distChegadaVideo value]]
 $ns at $prox_inicio "FonteVideo $prox_inicio $orig $dest [expr $num + 1] $max_video"
}

##################################
#### PROCEDIMENTO FONTES DADOS ###
##################################

proc FonteData {time orig dest num max_dados} {
 global ns distChegadaData distDurData fonte_data fontesDadosAtivas

 if {$fontesDadosAtivas($dest) < $max_dados } {
  if {! [$fonte_data($orig,$dest,$num) set running_]} {
   set final [expr $time + [$distDurData value]]
   #START
   $fonte_data($orig,$dest,$num) start
   incr fontesDadosAtivas($orig,$dest)
   incr fontesDadosAtivas($dest)
   #STOP
   $ns at $final "$fonte_data($orig,$dest,$num) stop;\
   incr fontesDadosAtivas($orig,$dest) -1;  incr fontesDadosAtivas($dest) -1"
  }
 }
 if {$num >= $max_dados} {set num 0}
 set prox_inicio [expr $time + [$distChegadaData value]/1000]
 $ns at $prox_inicio "FonteData $prox_inicio $orig $dest [expr $num + 1] $max_dados"
}
```

Apêndice I

Script TCL para o NS-2: Módulo de Eventos

```
############################
##### INICIALIZA FONTES #####
############################

$ns at 0 "FonteVoz 0 $sw6 $sw_helpdesk 1 $max_voip_helpdesk"
$ns at 0 "FonteVoz 0 $sw6 $sw2 1 $max_voip_sw2"
$ns at 0 "FonteVoz 0 $sw6 $sw3 1 $max_voip_sw3"

$ns at 0 "FonteVoz 0 $sw2 $sw_helpdesk 1 $max_voip_helpdesk"
$ns at 0 "FonteVoz 0 $sw2 $sw6 1 $max_voip_sw2"
$ns at 0 "FonteVoz 0 $sw2 $sw3 1 $max_voip_sw2"

$ns at 0 "FonteVoz 0 $sw3 $sw6 1 $max_voip_sw3"
$ns at 0 "FonteVoz 0 $sw3 $sw2 1 $max_voip_sw3"
$ns at 0 "FonteVoz 0 $sw3 $sw_helpdesk 1 $max_voip_helpdesk"

$ns at 0 "FonteVideo 0 $rot5 $swA 1 $max_video"
$ns at 0 "FonteVideo 0 $sw4 $swA 1 $max_video"
$ns at 0 "FonteVideo 0 $swA $sw2 1 $max_video"
$ns at 0 "FonteVideo 0 $swA $sw3 1 $max_video"
$ns at 0 "FonteVideo 0 $sw2 $sw3 1 $max_video2"
$ns at 0 "FonteVideo 0 $sw3 $sw2 1 $max_video2"
$ns at 0 "FonteVideo 0 $sw6 $swB 1 $max_video"

$ns at 0 "FonteData 0 $swB $sw4 1 $max_dados"
$ns at 0 "FonteData 0 $sw4 $swB 1 $max_dados"
$ns at 0 "FonteData 0 $sw3 $sw4 1 $max_dados2"
$ns at 0 "FonteData 0 $sw2 $sw4 1 $max_dados2"
```

Apêndice J

Script TCL para o NS-2: Módulo de Simulação *Best-Effort*

```
######################
### INITIAL SETUP ###
######################

#Create a simulator object
set ns [new Simulator]

####################
### CONFIGURACAO ###
####################

source ./codigo/config.tcl

######################
### TOPOLOGY SETUP ###
######################

source ./codigo/ns_topologia.tcl

######################
### TRAFEGO CITEX ###
######################

source ./codigo/ns_trafego.tcl

##################
### CRIA FONTES ###
##################

source ./codigo/ns_cria_fontes.tcl

####################
#### PROC EVENTOS ###
####################

source ./codigo/ns_procedimentos_fontes.tcl
```

```
################
#### EVENTOS ###
################

source ./codigo/ns_events.tcl

##############
#### TRACE ###
##############

proc trace {} {
 global ns tr core1 core2 borda1 sw_helpdesk borda3 borda2\
 borda6 sw2 sw3 sw6

 set tr [open "| grep exp > ./resultados/normalTrace.tr" w]

 $ns trace-queue $core1 $core2 $tr
 $ns trace-queue $core2 $core1 $tr
 $ns trace-queue $core1 $borda1 $tr
 $ns trace-queue $borda1 $core1 $tr
 $ns trace-queue $borda1 $sw_helpdesk $tr
 $ns trace-queue $core2 $borda2 $tr
 $ns trace-queue $borda2 $core2 $tr
 $ns trace-queue $core2 $borda3 $tr
 $ns trace-queue $borda3 $core2 $tr
 $ns trace-queue $core2 $borda6 $tr
 $ns trace-queue $borda6 $core2 $tr
 $ns trace-queue $borda2 $sw2 $tr
 $ns trace-queue $borda3 $sw3 $tr
 $ns trace-queue $borda6 $sw6 $tr
 $ns trace-queue $sw2 $borda2 $tr
 $ns trace-queue $sw3 $borda3 $tr
 $ns trace-queue $sw6 $borda6 $tr
}

###################################
#### IMPRESSAO DE FONTES ATIVAS ###
###################################

proc ImprimeFontesAtivas {} {

 global ns max_voip_helpdesk max_eventos sw2 sw6 max_voip_sw2 max_voip_sw3\
 sw3 sw_helpdesk fontesDadosAtivas total sw4 swB fontesVozAtivas\
 fontesVideoAtivas rot5 max_video swA  max_dados max_video2 max_dados2

 puts "Fontes ativas: Origem - Destino:
 Voz: totalFontesAtivas($fontesVozAtivas($total))
 Voz: sw6-sw_helpdesk   Max($max_voip_helpdesk) Atual($fontesVozAtivas($sw6,$sw_helpdesk))
 Voz: sw2-sw_helpdesk    Max($max_voip_helpdesk) Atual($fontesVozAtivas($sw2,$sw_helpdesk))
 Voz: sw3-sw_helpdesk    Max($max_voip_helpdesk) Atual($fontesVozAtivas($sw3,$sw_helpdesk))
 Voz: sw6-sw2        Max($max_voip_sw2) Atual($fontesVozAtivas($sw6,$sw2))
 Voz: sw3-sw2         Max($max_voip_sw2) Atual($fontesVozAtivas($sw3,$sw2))
 Voz: sw6-sw3        Max($max_voip_sw3) Atual($fontesVozAtivas($sw6,$sw3))
 Voz: sw2-sw3         Max($max_voip_sw3) Atual($fontesVozAtivas($sw2,$sw3))
 Voz: sw2-sw6        Max($max_voip_sw2) Atual($fontesVozAtivas($sw2,$sw6))
 Voz: sw3-sw6        Max($max_voip_sw3) Atual($fontesVozAtivas($sw3,$sw6))
```

```
 Video: rot5-swA Max($max_video) Atual($fontesVideoAtivas($rot5,$swA))
 Video: sw4-swA     Max($max_video) Atual($fontesVideoAtivas($sw4,$swA))
 Video: swA-sw2 Max($max_video) Atual($fontesVideoAtivas($swA,$sw2))
 Video: swA-sw3     Max($max_video) Atual($fontesVideoAtivas($swA,$sw3))
 Video: sw2-sw3 Max($max_video2) Atual($fontesVideoAtivas($sw2,$sw3))
 Video: sw3-sw2     Max($max_video2) Atual($fontesVideoAtivas($sw3,$sw2))
 Dados: swB-sw4       Max($max_dados) Atual($fontesDadosAtivas($swB,$sw4))
 Dados: sw4-sst       Max($max_dados) Atual($fontesDadosAtivas($sw4,$swB))
 Dados: sw3-sw4       Max($max_dados2) Atual($fontesDadosAtivas($sw3,$sw4))
 Dados: sw2-sw4       Max($max_dados2) Atual($fontesDadosAtivas($sw2,$sw4)) \n\n"
}

for {set i 0} {$i <= 4800} {set i [expr $i + $setupWindow]} {
 $ns at $i "puts \"Lista de fontes ativas no momento $i segundos)\""
 $ns at $i "ImprimeFontesAtivas"
}

##############
### FINAL ###
##############

proc finish {} {
 global ns tr
 $ns flush-trace
 close $tr
 exit 0
}

$ns at 960 "trace"

#Call the finish procedure
$ns at 4800 "finish"

#Run the simulation
$ns run
```

Apêndice K

Script TCL para o NS-2: Cenário DiffServ

```
######################
### CRIACAO DE NOS ###
######################

set swB [$ns node]
set sw_helpdesk [$ns node]
set swA [$ns node]
set borda1 [$ns node]
set core1 [$ns node]
set core2 [$ns node]
set borda6 [$ns node]
set sw6 [$ns node]
set rot5 [$ns node]
set borda5 [$ns node]
set sw3 [$ns node]
set borda3 [$ns node]
set sw4 [$ns node]
set borda4 [$ns node]
set sw2 [$ns node]
set borda2 [$ns node]

########################
### CRIACAO DE LINKS ###
########################

$ns duplex-link $core1 $core2 2Gb 25ms DropTail
$ns duplex-link $borda2 $sw2 1Gb 10ms DropTail
$ns duplex-link $borda3 $sw3 1Gb 10ms DropTail
$ns duplex-link $borda4 $sw4 1Gb 10ms DropTail
$ns duplex-link $borda6 $sw6 1Gb 10ms DropTail
```

```
$ns duplex-link $borda5 $rot5 1Gb 10ms DropTail
$ns simplex-link $core1 $borda1 1Gb 15ms dsRED/core
$ns simplex-link $borda1 $core1 1Gb 15ms dsRED/edge
$ns simplex-link $core2 $borda2 1Gb 15ms dsRED/core
$ns simplex-link $borda2 $core2 1Gb 15ms dsRED/edge
$ns simplex-link $core2 $borda3 1Gb 15ms dsRED/core
$ns simplex-link $borda3 $core2 1Gb 15ms dsRED/edge
$ns simplex-link $core2 $borda6 1Gb 15ms dsRED/core
$ns simplex-link $borda6 $core2 1Gb 15ms dsRED/edge
$ns simplex-link $core2 $borda4 1Gb 15ms dsRED/core
$ns simplex-link $borda4 $core2 1Gb 15ms dsRED/edge
$ns simplex-link $core2 $borda5 1Gb 15ms dsRED/core
$ns simplex-link $borda5 $core2 1Gb 15ms dsRED/edge
$ns duplex-link $borda1 $swB 1Gb 10ms DropTail
$ns duplex-link $borda1 $swA 1Gb 10ms DropTail
$ns duplex-link $borda1 $sw_helpdesk 1Gb 10ms DropTail

#########################
### CRIACAO TOPOLOGIA ###
#########################

set qcore1_borda1 [[$ns link $core1 $borda1] queue]
set qborda1_core1 [[$ns link $borda1 $core1] queue]
set qcore2_borda2 [[$ns link $core2 $borda2] queue]
set qborda2_core2 [[$ns link $borda2 $core2] queue]
set qcore2_borda3 [[$ns link $core2 $borda3] queue]
set qborda3_core2 [[$ns link $borda3 $core2] queue]
set qcore2_rot5 [[$ns link $core2 $borda5] queue]
set qrot5_core2 [[$ns link $borda5 $core2] queue]
set qcore2_borda6 [[$ns link $core2 $borda6] queue]
set qborda6_core2 [[$ns link $borda6 $core2] queue]
set qcore2_borda4 [[$ns link $core2 $borda4] queue]
set qborda4_core2 [[$ns link $borda4 $core2] queue]

#TAMANHO DAS FILAS
$ns queue-limit $core1 $core2 $qlim
$ns queue-limit $core2 $core1 $qlim
$ns queue-limit $borda2 $sw2 $qlim
$ns queue-limit $borda3 $sw3 $qlim
$ns queue-limit $borda4 $sw4 $qlim
$ns queue-limit $borda6 $sw6 $qlim
$ns queue-limit $borda5 $rot5 $qlim
$ns queue-limit $core1 $borda1 $qlim
$ns queue-limit $borda1 $core1 $qlim
$ns queue-limit $core2 $borda2 $qlim
$ns queue-limit $borda2 $core2 $qlim
```

```
$ns queue-limit $core2 $borda3 $qlim
$ns queue-limit $borda3 $core2 $qlim
$ns queue-limit $core2 $borda6 $qlim
$ns queue-limit $borda6 $core2 $qlim
$ns queue-limit $core2 $borda4 $qlim
$ns queue-limit $borda4 $core2 $qlim
$ns queue-limit $core2 $borda5 $qlim
$ns queue-limit $borda5 $core2 $qlim
$ns queue-limit $borda1 $swB $qlim
$ns queue-limit $swB $borda1 $qlim
$ns queue-limit $borda1 $swA $qlim
$ns queue-limit $swA $borda1 $qlim
$ns queue-limit $borda1 $sw_helpdesk $qlim
$ns queue-limit $sw_helpdesk $borda1 $qlim
```

Apêndice L

Script TCL para o NS-2: Módulo de Políticas

```
#VOZ
$qborda1_core1 meanPktSize 210
$qborda1_core1 set numQueues_ 2
$qborda1_core1 setNumPrec 1
$qborda1_core1 addPolicyEntry [$sw_helpdesk id] [$sw3 id] TokenBucket 46 72000 $cbs0
$qborda1_core1 addPolicyEntry [$sw_helpdesk id] [$sw2 id] TokenBucket 46 72000 $cbs0
$qborda1_core1 addPolicyEntry [$sw_helpdesk id] [$sw6 id] TokenBucket 46 72000 $cbs0
$qborda1_core1 addPolicerEntry TokenBucket 46 46
$qborda1_core1 addPolicerEntry TokenBucket 0 0
$qborda1_core1 addPHBEntry 46 0 0
$qborda1_core1 addPHBEntry 0 1 0
$qborda1_core1 configQ 0 0 5000 20000 0.02
$qborda1_core1 configQ 1 0 2500 10000 0.10

$qborda2_core2 meanPktSize 210
$qborda2_core2 set numQueues_ 2
$qborda2_core2 setNumPrec 1
$qborda2_core2 addPolicyEntry [$sw2 id] [$sw_helpdesk id] TokenBucket 46 72000 $cbs0
$qborda2_core2 addPolicyEntry [$sw2 id] [$sw3 id] TokenBucket 46 72000 $cbs0
$qborda2_core2 addPolicyEntry [$sw2 id] [$sw6 id] TokenBucket 46 72000 $cbs0
$qborda2_core2 addPolicerEntry TokenBucket 46 46
$qborda2_core2 addPolicerEntry TokenBucket 0 0
$qborda2_core2 addPHBEntry 46 0 0
$qborda2_core2 addPHBEntry 0 1 0
$qborda2_core2 configQ 0 0 5000 20000 0.02
$qborda2_core2 configQ 1 0 2500 10000 0.10

$qborda3_core2 meanPktSize 210
$qborda3_core2 set numQueues_ 2
$qborda3_core2 setNumPrec 1
```

```
$qborda3_core2 addPolicyEntry [$sw3 id] [$sw_helpdesk id] TokenBucket 46 72000 $cbs0
$qborda3_core2 addPolicyEntry [$sw3 id] [$sw2 id] TokenBucket 46 72000 $cbs0
$qborda3_core2 addPolicyEntry [$sw3 id] [$sw6 id] TokenBucket 46 72000 $cbs0
$qborda3_core2 addPolicerEntry TokenBucket 46 46
$qborda3_core2 addPolicerEntry TokenBucket 0 0
$qborda3_core2 addPHBEntry 46 0 0
$qborda3_core2 addPHBEntry 0 1 0
$qborda3_core2 configQ 0 0 5000 20000 0.02
$qborda3_core2 configQ 1 0 2500 10000 0.10

$qborda6_core2 meanPktSize 210
$qborda6_core2 set numQueues_ 2
$qborda6_core2 setNumPrec 1
$qborda6_core2 addPolicyEntry [$sw6 id] [$sw_helpdesk id] TokenBucket 46 72000 $cbs0
$qborda6_core2 addPolicyEntry [$sw6 id] [$sw2 id] TokenBucket 46 72000 $cbs0
$qborda6_core2 addPolicyEntry [$sw6 id] [$sw3 id] TokenBucket 46 72000 $cbs0
$qborda6_core2 addPolicerEntry TokenBucket 46 46
$qborda6_core2 addPolicerEntry TokenBucket 0 0
$qborda6_core2 addPHBEntry 46 0 0
$qborda6_core2 addPHBEntry 0 1 0
$qborda6_core2 configQ 0 0 5000 20000 0.02
$qborda6_core2 configQ 1 0 2500 10000 0.10

#VIDEO
$qborda1_core1 meanPktSize 1000
$qborda1_core1 set numQueues_ 2
$qborda1_core1 setNumPrec 1
$qborda1_core1 addPolicyEntry [$swA id] [$sw3 id] TokenBucket 0 1000000 $cbs2
$qborda1_core1 addPolicyEntry [$swA id] [$sw2 id] TokenBucket 0 1000000 $cbs2
$qborda1_core1 addPolicyEntry [$swA id] [$sw4 id] TokenBucket 0 1000000 $cbs2
$qborda1_core1 addPolicyEntry [$swB id] [$sw6 id] TokenBucket 0 1000000 $cbs2
$qborda1_core1 addPolicyEntry [$swA id] [$rot5 id] TokenBucket 0 1000000 $cbs2
$qborda1_core1 addPolicerEntry TokenBucket 46 46
$qborda1_core1 addPolicerEntry TokenBucket 0 0
$qborda1_core1 addPHBEntry 46 0 0
$qborda1_core1 addPHBEntry 0 1 0
$qborda1_core1 configQ 0 0 5000 20000 0.02
$qborda1_core1 configQ 1 0 2500 10000 0.10

$qborda2_core2 meanPktSize 1000
$qborda2_core2 set numQueues_ 2
$qborda2_core2 setNumPrec 1
$qborda2_core2 addPolicyEntry [$sw2 id] [$swA id] TokenBucket 0 1000000 $cbs2
$qborda2_core2 addPolicyEntry [$sw2 id] [$sw3 id] TokenBucket 0 1000000 $cbs2
$qborda2_core2 addPolicerEntry TokenBucket 46 46
$qborda2_core2 addPolicerEntry TokenBucket 0 0
```

```
$qborda2_core2 addPHBEntry 46 0 0
$qborda2_core2 addPHBEntry 0 1 0
$qborda2_core2 configQ 0 0 5000 20000 0.02
$qborda2_core2 configQ 1 0 2500 10000 0.10

$qborda3_core2 meanPktSize 1000
$qborda3_core2 set numQueues_ 2
$qborda3_core2 setNumPrec 1
$qborda3_core2 addPolicyEntry [$sw3 id] [$swA id] TokenBucket 0 1000000 $cbs2
$qborda3_core2 addPolicyEntry [$sw3 id] [$sw2 id] TokenBucket 0 1000000 $cbs2
$qborda3_core2 addPolicerEntry TokenBucket 46 46
$qborda3_core2 addPolicerEntry TokenBucket 0 0
$qborda3_core2 addPHBEntry 46 0 0
$qborda3_core2 addPHBEntry 0 1 0
$qborda3_core2 configQ 0 0 5000 20000 0.02
$qborda3_core2 configQ 1 0 2500 10000 0.10

$qborda5_core2 meanPktSize 1000
$qborda5_core2 set numQueues_ 2
$qborda5_core2 setNumPrec 1
$qborda5_core2 addPolicyEntry [$rot5 id] [$swA id] TokenBucket 0 1000000 $cbs2
$qborda5_core2 addPolicerEntry TokenBucket 46 46
$qborda5_core2 addPolicerEntry TokenBucket 0 0
$qborda5_core2 addPHBEntry 46 0 0
$qborda5_core2 addPHBEntry 0 1 0
$qborda5_core2 configQ 0 0 5000 20000 0.02
$qborda5_core2 configQ 1 0 2500 10000 0.10

$qborda4_core2 meanPktSize 1000
$qborda4_core2 set numQueues_ 2
$qborda4_core2 setNumPrec 1
$qborda4_core2 addPolicyEntry [$sw4 id] [$swA id] TokenBucket 0 1000000 $cbs2
$qborda4_core2 addPolicerEntry TokenBucket 46 46
$qborda4_core2 addPolicerEntry TokenBucket 0 0
$qborda4_core2 addPHBEntry 46 0 0
$qborda4_core2 addPHBEntry 0 1 0
$qborda4_core2 configQ 0 0 5000 20000 0.02
$qborda4_core2 configQ 1 0 2500 10000 0.10

$qborda6_core2 meanPktSize 1000
$qborda6_core2 set numQueues_ 2
$qborda6_core2 setNumPrec 1
$qborda6_core2 addPolicyEntry [$sw6 id] [$swB id] TokenBucket 0 1000000 $cbs2
$qborda6_core2 addPolicerEntry TokenBucket 46 46
$qborda6_core2 addPolicerEntry TokenBucket 0 0
$qborda6_core2 addPHBEntry 46 0 0
```

```
$qborda6_core2 addPHBEntry 0 1 0
$qborda6_core2 configQ 0 0 5000 20000 0.02
$qborda6_core2 configQ 1 0 2500 10000 0.10

#DADOS
$qborda1_core1 meanPktSize 1000
$qborda1_core1 set numQueues_ 2
$qborda1_core1 setNumPrec 1
$qborda1_core1 addPolicyEntry [$swB id] [$sw4 id] TokenBucket 0 400000 $cbs1
$qborda1_core1 addPolicerEntry TokenBucket 46 46
$qborda1_core1 addPolicerEntry TokenBucket 0 0
$qborda1_core1 addPHBEntry 46 0 0
$qborda1_core1 addPHBEntry 0 1 0
$qborda1_core1 configQ 0 0 5000 20000 0.02
$qborda1_core1 configQ 1 0 2500 10000 0.10

$qborda4_core2 meanPktSize 53
$qborda4_core2 set numQueues_ 2
$qborda4_core2 setNumPrec 1
$qborda4_core2 addPolicyEntry [$sw4 id] [$swB id] TokenBucket 0 400000 $cbs1
$qborda4_core2 addPolicyEntry [$sw4 id] [$sw3 id] TokenBucket 0 400000 $cbs1
$qborda4_core2 addPolicyEntry [$sw4 id] [$sw2 id] TokenBucket 0 400000 $cbs1
$qborda4_core2 addPolicerEntry TokenBucket 46 46
$qborda4_core2 addPolicerEntry TokenBucket 0 0
$qborda4_core2 addPHBEntry 46 0 0
$qborda4_core2 addPHBEntry 0 1 0
$qborda4_core2 configQ 0 0 5000 20000 0.02
$qborda4_core2 configQ 1 0 2500 10000 0.10

$qborda2_core2 meanPktSize 53
$qborda2_core2 set numQueues_ 2
$qborda2_core2 setNumPrec 1
$qborda2_core2 addPolicyEntry [$sw2 id] [$sw4 id] TokenBucket 0 400000 $cbs1
$qborda2_core2 addPolicerEntry TokenBucket 46 46
$qborda2_core2 addPolicerEntry TokenBucket 0 0
$qborda2_core2 addPHBEntry 46 0 0
$qborda2_core2 addPHBEntry 0 1 0
$qborda2_core2 configQ 0 0 5000 20000 0.02
$qborda2_core2 configQ 1 0 2500 10000 0.10

$qborda3_core2 meanPktSize 1000
$qborda3_core2 set numQueues_ 2
$qborda3_core2 setNumPrec 1
$qborda3_core2 addPolicyEntry [$sw3 id] [$sw4 id] TokenBucket 0 400000 $cbs1
$qborda3_core2 addPolicerEntry TokenBucket 46 46
$qborda3_core2 addPolicerEntry TokenBucket 0 0
```

```
$qborda3_core2 addPHBEntry 46 0 0
$qborda3_core2 addPHBEntry 0 1 0
$qborda3_core2 configQ 0 0 5000 20000 0.02
$qborda3_core2 configQ 1 0 2500 10000 0.10

$qcore2_borda4 setSchedularMode WRR
$qcore2_borda4 addQueueWeights 0 8
$qcore2_borda4 addQueueWeights 1 2
$qcore2_borda4 meanPktSize 53
$qcore2_borda4 set numQueues_ 2
$qcore2_borda4 setNumPrec 1
$qcore2_borda4 addPHBEntry 46 0 0
$qcore2_borda4 addPHBEntry 0 1 0
$qcore2_borda4 configQ 0 0 5000 20000 0.02
$qcore2_borda4 configQ 1 0 2500 10000 0.10

$qcore2_borda4 setSchedularMode WRR
$qcore2_borda4 addQueueWeights 0 8
$qcore2_borda4 addQueueWeights 1 2
$qcore2_borda4 meanPktSize 1000
$qcore2_borda4 set numQueues_ 2
$qcore2_borda4 setNumPrec 1
$qcore2_borda4 addPHBEntry 46 0 0
$qcore2_borda4 addPHBEntry 0 1 0
$qcore2_borda4 configQ 0 0 5000 20000 0.02
$qcore2_borda4 configQ 1 0 2500 10000 0.10

$qcore2_borda5 setSchedularMode WRR
$qcore2_borda5 addQueueWeights 0 8
$qcore2_borda5 addQueueWeights 1 2
$qcore2_borda5 meanPktSize 1000
$qcore2_borda5 set numQueues_ 2
$qcore2_borda5 setNumPrec 1
$qcore2_borda5 addPHBEntry 46 0 0
$qcore2_borda5 addPHBEntry 0 1 0
$qcore2_borda5 configQ 0 0 5000 20000 0.02
$qcore2_borda5 configQ 1 0 2500 10000 0.10

$qcore2_borda2 setSchedularMode WRR
$qcore2_borda2 addQueueWeights 0 8
$qcore2_borda2 addQueueWeights 1 2
$qcore2_borda2 meanPktSize 210
$qcore2_borda2 set numQueues_ 2
$qcore2_borda2 setNumPrec 1
$qcore2_borda2 addPHBEntry 46 0 0
$qcore2_borda2 addPHBEntry 0 1 0
```

```
$qcore2_borda2 configQ 0 0 5000 20000 0.02
$qcore2_borda2 configQ 1 0 2500 10000 0.10

$qcore2_borda2 setSchedularMode WRR
$qcore2_borda2 addQueueWeights 0 8
$qcore2_borda2 addQueueWeights 1 2
$qcore2_borda2 meanPktSize 1000
$qcore2_borda2 set numQueues_ 2
$qcore2_borda2 setNumPrec 1
$qcore2_borda2 addPHBEntry 46 0 0
$qcore2_borda2 addPHBEntry 0 1 0
$qcore2_borda2 configQ 0 0 5000 20000 0.02
$qcore2_borda2 configQ 1 0 2500 10000 0.10

$qcore2_borda2 setSchedularMode WRR
$qcore2_borda2 addQueueWeights 0 8
$qcore2_borda2 addQueueWeights 1 2
$qcore2_borda2 meanPktSize 53
$qcore2_borda2 set numQueues_ 2
$qcore2_borda2 setNumPrec 1
$qcore2_borda2 addPHBEntry 46 0 0
$qcore2_borda2 addPHBEntry 0 1 0
$qcore2_borda2 configQ 0 0 5000 20000 0.02
$qcore2_borda2 configQ 1 0 2500 10000 0.10

$qcore2_borda3 setSchedularMode WRR
$qcore2_borda3 addQueueWeights 0 8
$qcore2_borda3 addQueueWeights 1 2
$qcore2_borda3 meanPktSize 210
$qcore2_borda3 set numQueues_ 2
$qcore2_borda3 setNumPrec 1
$qcore2_borda3 addPHBEntry 46 0 0
$qcore2_borda3 addPHBEntry 0 1 0
$qcore2_borda3 configQ 0 0 5000 20000 0.02
$qcore2_borda3 configQ 1 0 2500 10000 0.10

$qcore2_borda3 setSchedularMode WRR
$qcore2_borda3 addQueueWeights 0 8
$qcore2_borda3 addQueueWeights 1 2
$qcore2_borda3 meanPktSize 1000
$qcore2_borda3 set numQueues_ 2
$qcore2_borda3 setNumPrec 1
$qcore2_borda3 addPHBEntry 46 0 0
$qcore2_borda3 addPHBEntry 0 1 0
$qcore2_borda3 configQ 0 0 5000 20000 0.02
$qcore2_borda3 configQ 1 0 2500 10000 0.10
```

```
$qcore2_borda3 setSchedularMode WRR
$qcore2_borda3 addQueueWeights 0 8
$qcore2_borda3 addQueueWeights 1 2
$qcore2_borda3 meanPktSize 53
$qcore2_borda3 set numQueues_ 2
$qcore2_borda3 setNumPrec 1
$qcore2_borda3 addPHBEntry 46 0 0
$qcore2_borda3 addPHBEntry 0 1 0
$qcore2_borda3 configQ 0 0 5000 20000 0.02
$qcore2_borda3 configQ 1 0 2500 10000 0.10

$qcore2_borda6 setSchedularMode WRR
$qcore2_borda6 addQueueWeights 0 8
$qcore2_borda6 addQueueWeights 1 2
$qcore2_borda6 meanPktSize 210
$qcore2_borda6 set numQueues_ 2
$qcore2_borda6 setNumPrec 1
$qcore2_borda6 addPHBEntry 46 0 0
$qcore2_borda6 addPHBEntry 0 1 0
$qcore2_borda6 configQ 0 0 5000 20000 0.02
$qcore2_borda6 configQ 1 0 2500 10000 0.10

$qcore2_borda6 setSchedularMode WRR
$qcore2_borda6 addQueueWeights 0 8
$qcore2_borda6 addQueueWeights 1 2
$qcore2_borda6 meanPktSize 1000
$qcore2_borda6 set numQueues_ 2
$qcore2_borda6 setNumPrec 1
$qcore2_borda6 addPHBEntry 46 0 0
$qcore2_borda6 addPHBEntry 0 1 0
$qcore2_borda6 configQ 0 0 5000 20000 0.02
$qcore2_borda6 configQ 1 0 2500 10000 0.10

$qcore1_borda1 setSchedularMode WRR
$qcore1_borda1 addQueueWeights 0 8
$qcore1_borda1 addQueueWeights 1 2
$qcore1_borda1 meanPktSize 210
$qcore1_borda1 set numQueues_ 2
$qcore1_borda1 setNumPrec 1
$qcore1_borda1 addPHBEntry 46 0 0
$qcore1_borda1 addPHBEntry 0 1 0
$qcore1_borda1 configQ 0 0 5000 20000 0.02
$qcore1_borda1 configQ 1 0 2500 10000 0.10

$qcore1_borda1 setSchedularMode WRR
```

```
$qcore1_borda1 addQueueWeights 0 8
$qcore1_borda1 addQueueWeights 1 2
$qcore1_borda1 meanPktSize 1000
$qcore1_borda1 set numQueues_ 2
$qcore1_borda1 setNumPrec 1
$qcore1_borda1 addPHBEntry 46 0 0
$qcore1_borda1 addPHBEntry 0 1 0
$qcore1_borda1 configQ 0 0 5000 20000 0.02
$qcore1_borda1 configQ 1 0 2500 10000 0.10

$qcore1_borda1 setSchedularMode WRR
$qcore1_borda1 addQueueWeights 0 8
$qcore1_borda1 addQueueWeights 1 2
$qcore1_borda1 meanPktSize 53
$qcore1_borda1 set numQueues_ 2
$qcore1_borda1 setNumPrec 1
$qcore1_borda1 addPHBEntry 46 0 0
$qcore1_borda1 addPHBEntry 0 1 0
$qcore1_borda1 configQ 0 0 5000 20000 0.02
$qcore1_borda1 configQ 1 0 2500 10000 0.10
```

Apêndice M

Script TCL para o NS-2: Módulo da simulação DiffServ

```
######################
### INITIAL SETUP ###
######################

#Create a simulator object
set ns [new Simulator]

####################
### CONFIGURACAO ###
####################

source ./codigo/config.tcl

######################
### TOPOLOGY SETUP ###
######################

source ./codigo/ns_topologia.tcl

######################
### TRAFEGO CITEX ###
######################

source ./codigo/ns_trafego.tcl

##################
### CRIA FONTES ###
##################

source ./codigo/ns_cria_fontes.tcl

####################
### POLICY E PHB ###
####################

source ./codigo/policy.tcl

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

```
#### PROC EVENTOS ###
#####################

source ./codigo/ns_procedimentos_fontes.tcl

#################
#### EVENTOS ###
#################

source ./codigo/ns_events.tcl

###############
#### TRACE ###
###############

proc trace {} {
 global ns tr core1 core2 borda1 sw_helpdesk borda3 borda2\
 borda6 sw2 sw3 sw6

 set tr [open "| grep exp > ./resultados/DiffServTrace.tr" w]

 $ns trace-queue $core1 $core2 $tr
 $ns trace-queue $core2 $core1 $tr
 $ns trace-queue $core1 $borda1 $tr
 $ns trace-queue $borda1 $core1 $tr
 $ns trace-queue $borda1 $sw_helpdesk $tr
 $ns trace-queue $core2 $borda2 $tr
 $ns trace-queue $borda2 $core2 $tr
 $ns trace-queue $core2 $borda3 $tr
 $ns trace-queue $borda3 $core2 $tr
 $ns trace-queue $core2 $borda6 $tr
 $ns trace-queue $borda6 $core2 $tr
 $ns trace-queue $borda2 $sw2 $tr
 $ns trace-queue $borda3 $sw3 $tr
 $ns trace-queue $borda6 $sw6 $tr
 $ns trace-queue $sw2 $borda2 $tr
 $ns trace-queue $sw3 $borda3 $tr
 $ns trace-queue $sw6 $borda6 $tr
}

###################################
#### IMPRESSAO DE FONTES ATIVAS ###
###################################

proc ImprimeFontesAtivas {} {

 global ns max_voip_helpdesk max_eventos sw2 sw6 max_voip_sw2 max_voip_sw3\
 sw3 sw_helpdesk fontesDadosAtivas total sw4 swB fontesVozAtivas\
 fontesVideoAtivas rot5 max_video swA  max_dados max_video2 max_dados2

 puts "Fontes ativas: Origem - Destino:
 Voz: totalFontesAtivas($fontesVozAtivas($total))
 Voz: sw6-sw_helpdesk   Max($max_voip_helpdesk) Atual($fontesVozAtivas($sw6,$sw_helpdesk))
 Voz: sw2-sw_helpdesk    Max($max_voip_helpdesk) Atual($fontesVozAtivas($sw2,$sw_helpdesk))
 Voz: sw3-sw_helpdesk    Max($max_voip_helpdesk) Atual($fontesVozAtivas($sw3,$sw_helpdesk))
 Voz: sw6-sw2        Max($max_voip_sw2) Atual($fontesVozAtivas($sw6,$sw2))
```

```
 Voz: sw3-sw2         Max($max_voip_sw2) Atual($fontesVozAtivas($sw3,$sw2))
 Voz: sw6-sw3        Max($max_voip_sw3) Atual($fontesVozAtivas($sw6,$sw3))
 Voz: sw2-sw3         Max($max_voip_sw3) Atual($fontesVozAtivas($sw2,$sw3))
 Voz: sw2-sw6        Max($max_voip_sw2) Atual($fontesVozAtivas($sw2,$sw6))
 Voz: sw3-sw6        Max($max_voip_sw3) Atual($fontesVozAtivas($sw3,$sw6))
 Video: rot5-swA Max($max_video) Atual($fontesVideoAtivas($rot5,$swA))
 Video: sw4-swA     Max($max_video) Atual($fontesVideoAtivas($sw4,$swA))
 Video: swA-sw2 Max($max_video) Atual($fontesVideoAtivas($swA,$sw2))
 Video: swA-sw3     Max($max_video) Atual($fontesVideoAtivas($swA,$sw3))
 Video: sw2-sw3 Max($max_video2) Atual($fontesVideoAtivas($sw2,$sw3))
 Video: sw3-sw2     Max($max_video2) Atual($fontesVideoAtivas($sw3,$sw2))
 Dados: swB-sw4       Max($max_dados) Atual($fontesDadosAtivas($swB,$sw4))
 Dados: sw4-sst       Max($max_dados) Atual($fontesDadosAtivas($sw4,$swB))
 Dados: sw3-sw4       Max($max_dados2) Atual($fontesDadosAtivas($sw3,$sw4))
 Dados: sw2-sw4       Max($max_dados2) Atual($fontesDadosAtivas($sw2,$sw4)) \n\n"
}

for {set i 0} {$i <= 4800} {set i [expr $i + $setupWindow]} {
 $ns at $i "puts \"Lista de fontes ativas no momento $i segundos)\""
 $ns at $i "ImprimeFontesAtivas"
}

##############
### FINAL ###
##############

proc finish {} {
 global ns tr
 $ns flush-trace
 close $tr
 exit 0
}

$ns at 960 "trace"

#Call the finish procedure
$ns at 4800 "finish"

#Run the simulation
$ns run
```