ESTEVAO OSCAR MOGNATTO JUNIOR

UM FRAMEWORK PARA CONTROLE DE ADMISSÃO UTILIZANDO O MÉTODO DE DIVERSIDADE DE CAMINHOS EM UM AMBIENTE DIFFSERV

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA

MINAS GERAIS - BRASIL

2009

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
M696f
2009

Mognatto Junior, Estevão Oscar, 1981-
Um framework para controle de admissão utilizando o método de diversidade de caminhos em um ambiente diffserv / Estevão Oscar Mognatto Junior. – Viçosa, MG, 2009.
xiv, 68f.: il. (algumas col.) ; 29cm.

Orientador: Carlos de Castro Goulart.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 65-68.

1. Rede de computadores. 2. NET Framework (Tecnologia de rede de computador). I. Universidade Federal de Viçosa. II.Título.

CDD 22.ed. 004.69

ESTEVÃO OSCAR MOGNATTO JUNIOR

UM FRAMEWORK PARA CONTROLE DE ADMISSÃO UTILIZANDO O MÉTODO DE DIVERSIDADE DE CAMINHOS EM UM AMBIENTE DIFFSERV

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 27 de março de 2009.

Alcione de Paiva Oliveira
(Co-Orientador)

Mauro Nacif Rocha
(Co-Orientador)

Leacir Nogueira Bastos

Luiz Henrique Andrade Correia

Carlos de Castro Goulart
(Orientador)

AGRADECIMENTOS

- Primeiramente a Deus por estar comigo em todos os passos desta jornada.

- Aos meus pais Estêvão e Ilka pelo incentivo, motivação e apoio nos momentos mais difíceis.

- Aos meus irmãos pelo companheirismo e colaboração.

- Ao professor Carlos pela paciência, disponibilidade a acima de tudo pela competência na orientação para o desenvolvimento deste trabalho.

- A todos os professores e funcionários do DPI, especialmente a José Luis Braga e Altino Alves de Souza Filho pela paciência, atenção e ajuda.

- A todos aos meus amigos do mestrado pelos bons momentos de convivência.

- À Universidade Federal de Viçosa, especialmente ao Departamento de Informática, pela oportunidade de realizar este trabalho.

BIOGRAFIA

ESTEVÃO OSCAR MOGNATTO JUNIOR, filho de Estêvão Oscar Mognatto e Ilka Mello e Silva Mognatto, nasceu em 12 de Julho de 1981, em Nova Venécia, Espírito Santo.

Em 1999, concluiu o 2º Grau no Centro Educacional Casa do Estudante, Aracruz, ES. Em 2002, iniciou o curso de Ciência da Computação na Universidade Federal de Viçosa (MG), que foi concluído no ano de 2006. Durante a graduação, teve a oportunidade de desenvolver trabalhos de iniciação científica por dois anos.

Em Maio de 2006, ingressou no Curso de Mestrado do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Viçosa, atuando na linha de pesquisa "Redes de Computadores". Nesse tempo, trabalhou por um ano e meio como Analista de Sistemas em uma empresa da cidade. Em março de 2009, submeteu-se à defesa desta dissertação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AF: Assured Forwarding

BA: Behavior Aggregate

BB: B*andwidth Broker*

CBR: Constant Bit Rate

CBS: Committed Burst Size

CIR: Committed Information Rate

DiffServ: *Differentiated Services*

DSCP: Differentiated Service CodePoint

ECN: Explicit Congestion Notification

EF: Expedited Forwarding

EvalVid: A Framework for Video Transmission and Quality Evaluation

FEC: Forward Error Correction

FTP: File Transfer Protocol

HP: *Hewlett-Packard*

IETF: Internet Engineering Task Force

IntServ: Integrated Services

IP: *Internet Protocol*

JPEG: *Joint Photographic Experts Group*

MDC: Multiple Description Coding

MDCAC: Multiple Description Coding Based Admission Control

MDRR: Modified Deficit Round Robin

MPEG: Motion Picture Experts Group

NS2: Network Simulator 2

PHB: Per-Hop Behavior

QoS: Quality of service

RED: Random Early Detection

RFC: Request for Comments

RSVP: Resource ReSerVation Protocol

RTP: Real-time Transport Protocol

SLA: *Service Level Agreement*

TCP: Transmission Control Protocol

TOS: *Type of Service*

VoD: Video on Demand

VoIP: Voice over IP

RESUMO

MOGNATTO JUNIOR, Estevão Oscar, M.Sc., Universidade Federal de Viçosa, março de 2009. **Um framework para controle de admissão utilizando o método de diversidade de caminhos em um ambiente diffserv.** Orientador: Carlos de Castro Goulart. Co-Orientadores: Alcione de Paiva Oliveira e Mauro Nacif Rocha.

O protocolo IP é baseado em um modelo chamado de melhor esforço, no qual todos os pacotes são tratados de forma igual, sem nenhum tipo de diferenciação ou mecanismo explícito de garantia de entrega. Este modelo foi desenvolvido para lidar com aplicações que não possuíam nenhum requisito de *QoS (Quality of service)* como aplicações de correio eletrônico e transferência de arquivos. Contudo, mais e mais sistemas de comunicação suportam diferentes tipos de transmissões em tempo real, como voz sobre IP (*VoIP)*, TV com interatividade e videoconferência. Obviamente, o modelo de melhor esforço não é adequado para lidar com esses novos tipos de sistemas de comunicação de alta qualidade em tempo real. Logo, novos modelos precisam ser desenvolvidos. Os desafios de transmitir fluxos de dados de alta qualidade sobre redes baseadas no modelo de melhor esforço incluem retardo de pacotes, perda de pacotes, variação do atraso fim-a-fim e tem motivado o desenvolvimento de diferentes soluções com o objetivo de suportar aplicações multimídia de forma eficiente. Neste contexto, o método de diversidade de caminhos (*path diversity*) e *MDC (Multiple Description Coding)* surgiram como técnicas promissoras para a transmissão robusta de tráfego multimídia sobre redes com a possibilidade de perda de pacotes como é o caso da Internet. O método de diversidade de caminhos é uma técnica que explora diferentes caminhos disponíveis entre dois *hosts* específicos na Internet. *MDC* é uma técnica na qual um fluxo de vídeo é dividido em múltiplos subfluxos (chamados de descrições), de tal forma que a qualidade do vídeo recebido aumenta com o número de descrições recebidas. Neste trabalho, foi desenvolvido um conjunto de disciplinas de controle de admissão para uma rede *DiffServ* que leva em consideração a utilização do método de diversidade

de caminhos (um fluxo pode ser admitido utilizando-se mais de um caminho). Nosso principal objetivo foi prover um entendimento dos benefícios de usar as técnicas de *MDC* e de diversidade de caminhos combinadas para se enviar fluxos de vídeo sobre a Internet. Os parâmetros para a admissão de um fluxo são a banda passante necessária, o tempo de início e o tempo de fim do fluxo. Os resultados apresentados neste trabalho mostram que a utilização do método de diversidade de caminhos exibe uma melhora nas características de *QoS* se comparado com o método tradicional, no qual o melhor caminho é sempre utilizado. Os experimentos indicam que a abordagem utilizada foi capaz de utilizar os recursos da rede de uma forma mais eficiente e atingir um melhor nível de requisições aceitas, assim como uma menor taxa de perda de pacotes.

ABSTRACT

MOGNATTO JUNIOR, Estevão Oscar, M.Sc., Universidade Federal de Viçosa, March, 2009. **A framework for admission control based on path diversity in a diffserv environment.** Adviser: Carlos de Castro Goulart. Co-Advisers: Alcione de Paiva Oliveira and Mauro Nacif Rocha.

The IP protocol is based on a model called best effort, where all packets are treated the same way, without any discrimination or explicit delivery guarantees. It was developed to deal with application without *QoS* (Quality of service) requirements such as e-mail and *FTP* (File Transfer Protocol). However, more and more communication systems are supporting different kinds of real-time transmission, such as voice over IP (*VoIP*), interactive TV and videoconferencing. Of course, the best effort model is not suitable to cope with these new high quality real-time communication systems, so a new model needs to be developed. The challenges of transmitting high quality streaming over best-effort networks include delay, packet loss, varying bandwidth and have led to several different approaches to efficiently support multimedia streaming applications. In this context, path diversity and *MDC* (Multiple Description Coding) have emerged as promising techniques for robust transmission of multimedia traffic over lossy packet networks as the Internet. Path diversity is a technique that exploits different paths available between two specific hosts in the Internet. *MDC* is a technique where the video stream is divided into multiple sub-streams (called descriptions), such that the quality of the received video increases with the number of descriptions received. In this paper, a set of admission control disciplines was developed in a *DiffServ* environment which take into consideration path diversity, in which one flow can be accepted using more than one single path. Our main goal is to provide the fundamental understanding of the benefits of using methodologies like *MDC* and path diversity together to deliver continuous media stream over the Internet. The parameters for admission are the bandwidth, the request start and stop time. Our results show that the use of path diversity exhibits better *QoS* characteristics when compared with the traditional

methodology, where the best path is always used. The experiments indicate that the approach used was able to use the network resources efficiently and to achieve a better level of requests accepted as well as decreases the packet loss rate.

Capítulo 1

Introdução

O sucesso do protocolo IP (*Internet Protocol*) [Postel 1981] se deve em grande parte à simplicidade de sua arquitetura, onde a complexidade é concentrada nos pontos finais da rede (*end-hosts*). Tal simplicidade tornou o modelo escalável, o que possibilitou a absorção do crescimento da demanda, que hoje chega à ordem de milhões de computadores conectados pelo globo. A arquitetura da Internet foi desenvolvida baseada no modelo de melhor esforço (*best-effort*) e teve o objetivo de atender os aplicativos típicos de sua época: correio eletrônico e transferência de arquivos.

Neste modelo, todos os pacotes são tratados de maneira igual, independentemente da aplicação à qual pertençam e sem se considerar parâmetros de qualidade de serviço como o retardo fim-a-fim, variação do atraso fim-a-fim, banda passante e taxa de perda de pacotes. Logo, uma aplicação com requisitos de atraso fim-a-fim é tratada da mesma forma que uma aplicação de correio eletrônico, que não possui requisitos de atraso fim-a-fim. Outro problema ocorre quando a rede está congestionada, pois os mecanismos de controle de congestionamento presentes hoje na Internet descartam pacotes de forma indiscriminada.

Contudo, a utilização de aplicações multimídia na Internet vem ganhando uma importância considerável e estes tipos de tráfego são altamente susceptíveis a parâmetros de qualidade de serviço (*QoS – Quality of Service*). A utilização de mecanismos de *QoS* torna-se essencial em aplicações como voz sobre IP (*VoIP*), videoconferência, vídeo sob demanda (*VoD*), educação a distância (*e-learning)*, comércio eletrônico (*e-commerce)*, etc. A Tabela 1.1 mostra os diferentes tipos de aplicações e seus respectivos parâmetros de qualidade de serviço.

Tabela 1.1. Principais aplicações da Internet [Tanenbaum 2003].

Aplicação	**Confiabilidade**	**Atraso**	**Variação do Atraso**	**Banda Passante**
Correio Eletrônico	Alto	Baixo	Baixo	Baixo
Transferência de Arquivos	Alto	Baixo	Baixo	Médio
Acesso Web	Alto	Médio	Baixo	Médio
Login Remoto	Alto	Médio	Médio	Baixo
Áudio sob demanda	Baixo	Baixo	Alto	Médio
Vídeo sob demanda	Baixo	Baixo	Alto	Alto
Telefonia	Baixo	Alto	Alto	Baixo
Videoconferência	Baixo	Alto	Alto	Alto

Dentre os tipos de aplicações listadas na Tabela 1.1, o interesse deste trabalho se concentra mais no estudo dos fluxos de mídia contínua, que seriam o caso das quatro últimas aplicações da tabela. Observe que estas aplicações apresentam um requisito baixo para o parâmetro de confiabilidade, significando que a perda ou alteração de um ou alguns bits não influenciarão de forma significativa a qualidade final da mídia recebida. Contudo, tais fluxos apresentam uma grande sensibilidade no parâmetro de *QoS* chamado de rajada de perda de pacotes (vários pacotes seguidos são perdidos), apesar deste parâmetro não aparecer na Tabela 1.1.

O gerenciamento desses parâmetros de *QoS* é uma tarefa muito complexa e envolve conceitos tanto administrativos como técnicos que geralmente demandam tarefas como a distribuição de recursos disponíveis, a configuração de níveis de usuários, a diferenciação das aplicações e a confiabilidade.

Neste contexto, vários modelos surgiram na tentativa de tornar possível a transmissão de tráfego multimídia sobre redes sem garantias de qualidade como é o caso da Internet. Um deles é a utilização do método de diversidade de caminhos (*path diversity*), no qual os pacotes de uma aplicação são enviados para o destino utilizando dois ou mais caminhos distintos [Teixeira 2003]. O protocolo de roteamento padrão usado na Internet procura usar sempre o melhor caminho entre origem e destino para encaminhar todos os pacotes de um mesmo fluxo. A abordagem de diversidade de caminhos apresenta alguns benefícios como melhorar a confiabilidade das transmissões, suavizar o tráfego [Maxemchuk 1975] e reduzir o

tamanho da rajada de perda de pacotes [Liang 2003] [Apostolopoulos 2001]. A redução da rajada de perda é especialmente importante para o caso de transmissões de vídeo e áudio.

Com base na possibilidade da utilização do método de diversidade de caminhos para fluxos multimídia, o objetivo deste trabalho foi investigar várias disciplinas de controle de admissão em um ambiente *DiffServ*, levando-se em consideração a transmissão de um tráfego de dados por mais de um caminho.

1.1. O Problema e sua importância

O modelo de Serviços Diferenciados (*DiffServ*) [Blake 1998] tradicional criado pelo *IETF* (*Internet Engineering Task Force*) dificilmente alcança os requisitos de *QoS* solicitados quando a rede se encontra em uma situação de congestionamento. Isso acontece porque as redes *DiffServ* atuais não conhecem o estado da rede interna, ou seja, em um determinado momento os roteadores de borda não têm informação sobre o que está acontecendo nos roteadores de centro. Logo, mesmo havendo uma situação de congestionamento na rede interna, os roteadores de borda continuarão encaminhando tráfego excedente para dentro da rede. Sendo a arquitetura *DiffServ* uma solução escalável, a sua utilização juntamente com um controle de admissão eficiente seria uma solução alternativa ao modelo *IntServ* [Braden 1994].

Na Internet atual, é cada vez maior o uso de aplicações como vídeo sobre demanda (*VoD*), videoconferência, voz sobre IP (*VoIP*), dentre outros. Tais aplicações são altamente susceptíveis a parâmetros de qualidade de serviço como a variação do atraso fim-a-fim, a variação do retardo fim-a-fim e a perda de pacotes, além de demandarem uma grande utilização de banda passante. Alguns destes fluxos apresentam características específicas, como por exemplo, o padrão *MPEG* (*Motion Picture Experts Group*). Como será visto posteriormente, na transmissão deste tipo de vídeo, alguns pacotes serão específicos e terão uma importância maior do que outros, como pacotes que transportam uma cena base. A perda deste tipo de pacote na rede pode gerar perdas mais significativas de informação do que a perda de um pacote que não transmita cenas base.

Logo, devido ao aumento do uso de aplicações multimídia e a tendência de convergência destes meios de comunicação, mecanismos que previnam o

congestionamento na rede, como disciplinas de controle de admissão, se tornaram essenciais.

1.2. Objetivos

1.2.1. Objetivo geral

Avaliar disciplinas de controle de admissão em um ambiente *DiffServ* que leve em consideração a transmissão de um tráfego de dados multimídia por mais de um caminho. O controle de admissão deverá garantir a manutenção da Qualidade de Serviço previamente acordada para os usuários já admitidos, maximizando ao mesmo tempo, a utilização da rede.

1.2.2. Objetivos específicos

1) Criar mecanismos para que pacotes de alta prioridade que não estejam de acordo com o perfil de tráfego previamente acordados através de um *SLA* (*Service Level Agreement*) sejam aceitos caso haja recursos na rede;

2) Em situações de congestionamento, prover mecanismos para identificação de fluxos que já estejam admitidos, mas que não estejam seguindo o perfil de tráfego acordado.

Os dois objetivos específicos acima visavam maximizar a utilização da rede, pois utilizando o primeiro objetivo permite-se que um fluxo utilize mais banda passante do que este havia solicitado, caso a rede tenha banda passante disponível. A utilização do segundo objetivo específico visa desfazer as ações feitas pelo primeiro objetivo, ou seja, fazer com que os fluxos voltem a utilizar a banda passante que haviam se comprometido, no caso de haver uma situação de congestionamento na rede. Nenhum desses objetivos foram implementados neste trabalho, devido a questões de tempo e escopo.

1.3. Motivação e contribuições

Levando-se em consideração os problemas apresentados na sub-seção 1.1, fica claro que mecanismos adicionais devem ser desenvolvidos para se evitar situações de congestionamento na rede. A própria *RFC (Request For Comments)* que define a arquitetura *DiffServ* [Blake 1998] diz que “mecanismos adicionais de alocação como o negociador de banda passante (*BB – Bandwidth Broker)* ou o protocolo *RSVP*

podem ser usados para alocar recursos dinamicamente para um *BA* (*Behavior Aggregate*)".

Com isso em mente, este trabalho apresenta várias disciplinas de controle de admissão para um rede *DiffServ* que leva em consideração a utilização do método de diversidade de caminhos, ou seja, onde um fluxo possa ser admitido utilizando-se mais de um caminho. Os parâmetros principais para a admissão de um fluxo foram a banda passante necessária, o tempo de início e o tempo de fim do fluxo.

1.4. Proposta do trabalho

Neste trabalho foi desenvolvido um algoritmo de controle de admissão que leva em consideração a transmissão de um tráfego de dados multimídia por mais de um caminho. Este algoritmo foi nomeado *MDCAC (Multiple Description Coding Based Admission Control).*

Para isso, o *framework* para controle de admissão desenvolvido por [Bouras 2007] foi estendido de tal forma a implementar o algoritmo *MDCAC*. Este *framework* é uma implementação de um negociador de banda passante (*Bandwidth Broker - BB)* para uma rede *DiffServ*, onde o ns2 (*Network Simulator*) foi utilizado como ambiente de simulação [McCanne 2008].

O ns2 é um simulador discreto de eventos desenvolvido para a pesquisa de diversos tipos de redes de computadores. É um simulador muito popular no meio acadêmico por sua facilidade de expansão e por utilizar um modelo de código aberto (*open source)* de desenvolvimento, além de possuir uma vasta documentação. O ns2 suporta a simulação de roteamento *multicast* e *unicast* sobre rede cabeadas ou sem-fio, tem suporte a redes *ad-hoc*, dentre outras funcionalidades.

1.5. Estrutura da Dissertação

O restante desta dissertação encontra-se estruturada da seguinte forma: no Capítulo 2 são apresentados alguns aspectos sobre o uso eficiente de redes IP para fluxos de vídeo e é o resultado da pesquisa bibliográfica sobre as áreas relacionadas ao trabalho. As várias versões das disciplinas de controle de admissão propostas são apresentadas no Capítulo 3, juntamente com a implementação do *framework* utilizado. No Capítulo 4 são apresentados os resultados das simulações e no Capítulo 5 as conclusões e alguns possíveis trabalhos futuros.

Capítulo 2

Alguns aspectos sobre o uso eficiente de redes IP para fluxos de vídeo

Neste capítulo serão introduzidos alguns conceitos fundamentais para o entendimento do trabalho desenvolvido. Serão abordados primeiro os aspectos mais gerais, como conceitos de congestionamento e de qualidade de serviços em *redes IP*. Em seguida, serão abordados temas sobre metodologias que se propõe à utilização da rede de uma forma mais eficiente, como o uso do método de diversidade de caminhos junto com a técnica *MDC* (*Multiple Description Coding*). Finalmente, algumas ferramentas que podem ser utilizadas para a implementação das técnicas apresentadas acima serão discutidas.

2.1. Conceitos sobre congestionamentos em redes IP

Um dos principais problemas encontrados em redes IP é o problema do congestionamento. O congestionamento ocorre quando o número de pacotes enviados para parte de uma rede é maior do que os recursos que a rede dispõe para acomodar a carga enviada.

De uma forma ideal, pode-se imaginar que uma rede ou parte de uma rede apresentasse um limite de carga que esta poderia receber, em pacotes durante um determinado tempo T, na qual ao final do intervalo T quase todos os recursos da rede estivessem saturados. Supondo que esse valor fosse X, pode-se pensar que ao receber X acrescido de um valor Y no tempo T, a rede encaminhasse X pacotes e descartasse Y pacotes. Infelizmente não é assim que as coisas acontecem em uma rede IP. O envio de X acrescido de Y levaria a rede a uma situação na qual os seus recursos estariam completamente esgotados, caracterizando uma situação de congestionamento, na qual nenhum ou quase nenhum pacote seria encaminhado. O que aconteceria é que a rede entraria em um colapso, descartando pacotes indiscriminadamente até que a situação se normalizasse. A Figura 2.1 ilustra a situação apresentada.

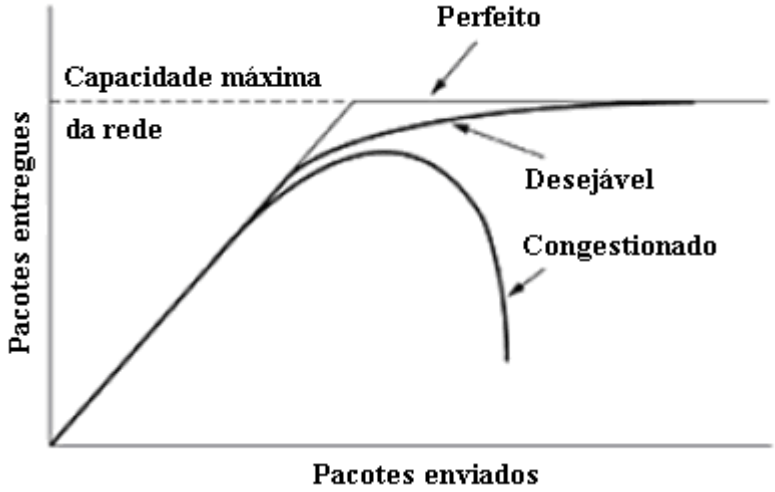

Figura 2.1. Rede IP em uma situação de congestionamento [Tanenbaum 2003].

A linha pontilhada na Figura 2.1 representaria a capacidade máxima da rede. Uma situação perfeita é indicada pela linha superior, na qual a rede aumenta a taxa de entrega de maneira proporcional ao aumento dos pacotes enviados, até o seu limite de X pacotes. Quando o número de pacotes enviados ultrapassa X, a rede continuaria a entregar X pacotes e descartaria o excedente (Y pacotes). Como a situação perfeita é muito difícil de se alcançar, a linha do meio representa o que seria desejável que acontecesse. A linha de baixo representa o que realmente acontece quando a rede está congestionada.

Diversas são as possibilidades que levam uma rede ao congestionamento e existem três fatores fundamentais que devem ser considerados ao se analisar as causas de um congestionamento: (1) o tamanho da fila dos roteadores; (2) o poder de processamento dos roteadores e (3) a banda passante dos *links*.

Obviamente, se vários *links* transportarem pacotes simultaneamente para um roteador com uma fila pequena, esta logo se esgotaria e pacotes começariam a ser descartados. O aumento da fila ajudaria a diminuir o problema até certo ponto, mas segundo Nagle (1987), se usássemos uma fila com tamanho infinito em todos os roteadores do domínio, o problema pioraria em vez de melhorar. A razão é que em uma situação de grande carga, as filas se tornariam muito cheias e os pacotes começariam a sofrem *timeout* o que obrigaria certos protocolos como o *TCP* (*Transmission Control Protocol*) a reenviar pacotes, o que tornaria o problema ainda pior.

Processadores limitados também podem levar a rede a uma situação de congestionamento, pois se o processador não for capaz de processar os pacotes rapidamente, as filas associadas a estes se encherão rapidamente e pacotes serão perdidos. De forma análoga, uma banda passante pequena também pode levar a perda de pacotes, pois se uma linha de saída não for capaz de transportar pacotes a uma taxa satisfatória, a fila associada a esta encherá.

Logo, é possível perceber que existe uma estreita relação entre o tamanho da fila, o poder de processamento e a capacidade do *link*. Deve haver um equilíbrio entre estes recursos para se diminuir as chances de congestionamento, ou seja, não adianta nada aumentar o valor da banda passante do *link* se o poder de processamento associado tiver um valor pequeno, assim como o tamanho da fila deve ser uma função do poder de processamento e a banda passante, de tal forma a se absorver eventuais rajadas de pacotes.

Segundo Tanenbaum (2003) é importante se estabelecer a diferença entre controle de congestionamento e controle de fluxo. Controle de congestionamento é o mecanismo para se evitar que uma situação de congestionamento se inicie e envolve o sistema como um todo. Estão envolvidos neste processo os emissores (*hosts*), as características dos roteadores como tamanho de fila e poder de processamento e todos os outros fatores que venham a influenciar a rede de uma forma global.

Controle de fluxo se refere ao controle de tráfego ponto a ponto, ou seja, o controle do fluxo enviado entre um emissor e um receptor. Tem como objetivo evitar que o emissor envie pacotes a uma taxa maior do que o receptor pode suportar. Geralmente envolvem protocolos que utilizam algum tipo de mecanismo de realimentação do receptor para dizer ao emissor como está a capacidade de recepção.

Existem vários algoritmos e métodos para o controle de congestionamento, tanto no contexto de redes orientadas a conexão quanto para redes não orientadas a conexão, como no caso das redes IP. Yang e Reddy (1995) desenvolveram uma taxonomia para os algoritmos de controle de congestionamento. Primeiro eles os dividiram em duas categorias: os de laço aberto (*open loop*) e os de laço fechado (*close loop*). Logo em seguida dividiram os algoritmos de laço aberto em duas subcategorias: os que agem nas fontes de dados e os que agem nos destinos de

dados. Também dividiram os algoritmos de laço fechado em duas subcategorias: os de realimentação explícita versus os de realimentação implícita.

Os algoritmos de laço aberto tentam resolver o problema do congestionamento através de uma arquitetura bem projetada, ou seja, em teoria devido ao projeto desenvolvido, o problema de congestionamento não ocorreria. Todos os algoritmos desta categoria têm em comum o fato de tomarem decisões sem medirem o estado atual da rede. Tais decisões seriam relacionadas com quando aceitar um novo fluxo, quando fazer o descarte de pacotes, decisões de escalonamento, dentre outros.

Já os algoritmos de laço fechado são baseados em mecanismos com realimentação de informações recebidas da rede e envolvem questões como: onde um congestionamento está ocorrendo, onde as ações devem ser tomadas e como corrigir estes problemas. Nos algoritmos de realimentação explícita, como o próprio nome diz, existe um mecanismo explícito para avisar o emissor que um congestionamento está ocorrendo. É claro que os pacotes enviados explicitamente pelos roteadores podem tornar o problema ainda pior, por isso foram desenvolvidos os algoritmos com realimentação implícita, no qual o emissor deduz que um congestionamento está ocorrendo analisando informações como o tempo de chegada de pacotes de confirmações (*acknowledgements packets*) e a ocorrência de *timeouts* de pacotes.

2.2. Mecanismos de controle de Qualidade de Serviço (*QoS*) em redes IP

QoS pode ser definida como um conjunto de requisitos necessários para que uma determinada funcionalidade seja executada de acordo com os parâmetros acordados previamente através de um *SLA* (*Service Level Agreement*). Devido ao fato do modelo de melhor esforço ter se tornado inadequado para o tráfego de aplicações de mídia contínua (como vídeo e áudio) o *IETF* (*Internet Engineering Task Force*) desenvolveu dois modelos de controle de *QoS* com abordagem distintas: os Serviços Integrados (*IntServ*) [Braden 1994] [Shenker 1997] e os Serviços Diferenciados (*DiffServ*) [Blake 1998].

2.2.1. IntServ

O *IntServ (Integrated Services)* é um mecanismo que emula a existência de circuitos virtuais em uma rede IP, ou seja, é uma tentativa de prover um nível garantido de qualidade de serviço através de reserva de recursos. O principal protocolo definido pelo *IETF* para esta arquitetura é o *RSVP* (*Resource ReSerVation Protocol*) [Braden 1997] e é utilizado para fazer a alocação de recursos na rede.

O *RSVP* é um mecanismo complexo de *QoS* e exerce um rigoroso controle para cada fluxo fim-a-fim que trafega na rede. A reserva é sempre unidirecional, assim para uma aplicação enviar e receber dados são necessárias duas reservas. Cada reserva de recurso é feita através da especificação da descrição do fluxo (*FlowSpec*) [Wroclawski 1997b], que é composta de dois parâmetros:

- o *Rspec* (Resource Specification) – indica a classe de serviço desejada;
- o *Tspec* (Traffic Specification) – indica as características do que será transmitido;

O *IntServ* oferece dois tipos de serviços: o Serviço Garantido (*Guaranteed Service*) [Guerin 1997] e o Serviço de Carga Controlada (*Controlled Load Service)* [Wroclawski 1997a]. O Serviço Garantido oferece um maior nível de garantia no cumprimento de requisitos de *QoS* por utilizar um protocolo de reserva de recursos como o *RSVP,* que por sua vez utiliza os parâmetros *Rspec* e *Tspec* conforme descritos anteriormente. Já o Serviço de Carga Controlada é mais flexível e tem a intenção de oferecer uma rede sem congestionamentos por meio de um controle de carga na rede e para isso só utiliza o parâmetro *Tspec* para a especificação de tráfego.

Para que o *IntServ* seja implementado de forma satisfatória, todos os roteadores do domínio devem suportar o protocolo de reserva de recursos utilizado, que no caso seria o *RSVP*. Além disso, cada roteador deve manter informações do estado de cada fluxo que esteja circulando através da rede. Devido a estes problemas de escalabilidade o *IntServ* se tornou inadequado como arquitetura padrão da Internet. Para contornar os problemas que o *IntServ* apresentava, o *IETF* criou o *DiffServ* no qual os fluxos são tratados como agregados e não individualmente.

2.2.2. DiffServ

A arquitetura *DiffServ* (*Differentiated Services*) busca a simplicidade através da agregação de fluxos que receberão um mesmo nível de qualidade de serviço. Para isso define classes que receberão tratamento diferenciado dentro de um nó *DiffServ*. Este tratamento diferenciado será definido mediante de contratos de nível de serviço (*Service Level Agreements - SLAs*) entre clientes e seus provedores de serviço.

Cada agregado de fluxos deve ser classificado e para isso o modelo *DiffServ* definiu uma reutilização do campo *TOS* (*Type of Service*) de 8 bits do cabeçalho IPv4 e do campo *Traffic Class* do IPv6 (também de 8 bits), que foram renomeados como *byte DS* ou *DSField* [Nichols 1998]. O *byte DS* é dividido em duas partes: os seis primeiros bits são chamados de *DSCP (Differentiated Service CodePoint)* e indicam uma marcação que definirá uma classe de tráfego para um determinado pacote e os dois últimos bits representam o campo *EC*N (*Explicit Congestion Notification*) [Black 2001], conforme é mostrado na Figura 2.2.

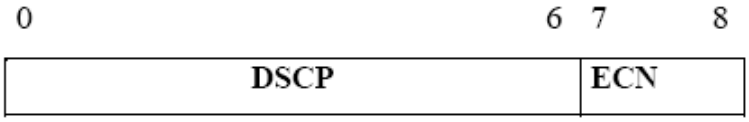

Figura 2.2. *DS Field* para IPv4 e IPv6 (*Traffic Clas*s) [Black 2001].

Um conjunto de nós (roteadores) *DiffServ* caracteriza um domínio *DiffServ*, onde os nós de borda do domínio são chamados de nós de fronteira (*edge)* e os nós que ficam entre os nós de borda são chamados de nós de centro (*core*). A marcação *DSCP* é adicionada a cada pacote pelo *host* de origem ou pelo primeiro nó de borda de um domínio *DiffServ*.

A partir da classificação das marcações *DSCP* dos pacotes, cada nó dentro da rede *DiffServ* encaminha os pacotes de acordo com o mapeamento do campo *DSCP* para um comportamento específico de *QoS* e este mapeamento é chamado de *PHB (Per-Hop Behavior)*. A Figura 2.3 mostra os elementos que compõem um nó *DiffServ*.

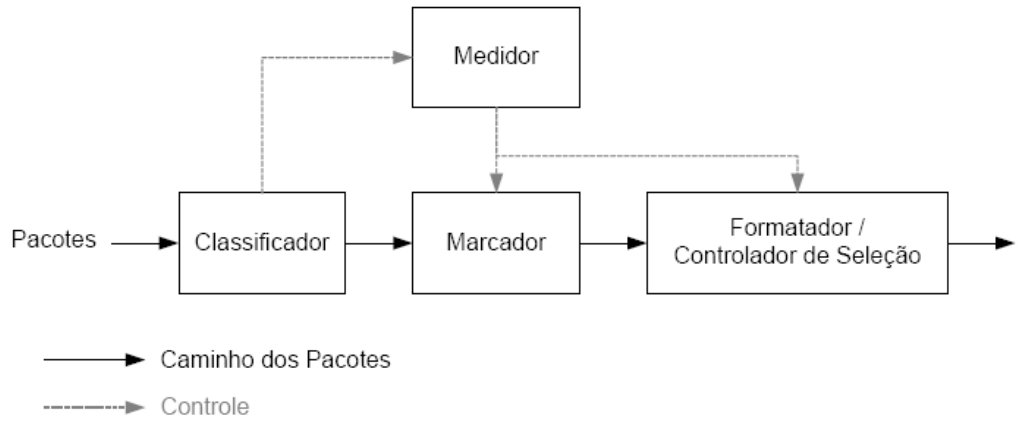

Figura 2.3. Componentes de um nó *DiffServ* [Blake 1998].

A seguir temos uma explicação de cada componente da Figura 2.3:

- Classificador – seleciona os pacotes através dos cabeçalhos e encaminha aqueles que correspondem às regras de classificação para processamento posterior. Existem dois tipos de classificadores, os multicampos, que utilizam outras informações além do *byte DS* e os de comportamento agregado (*BA*), que somente se baseiam no *byte DS*;
- Medidor – verifica se os pacotes classificados e encaminhados estão seguindo o perfil de tráfego anteriormente definidos através de um *SLA* e passa estas informações para outros elementos do nó;
- Marcador – responsável pela marcação ou remarcação de pacotes. A marcação é feita quando o pacote chega sem nenhuma marcação e a remarcação ocorre em duas situações: (1) quando o nó seguinte no encaminhamento possui uma interpretação diferente para o *byte DS* e (2) quando o tráfego não seguir o perfil de tráfego definido;
- Formatador / Controlador de Seleção – é responsável pela adaptação e descarte dos pacotes de acordo com as propriedades estabelecidas pelo *PHB.*

Diferente do que ocorreu no *IntServ,* no qual as especificações dos serviços foram totalmente detalhadas, no *DiffServ* o *IETF* decidiu não detalhar os serviços e definiu apenas duas classes de encaminhamento: o *PHB EF (Expedited Forwarding)* [Jacobson 1999] e o *PHB AF (Assured Forwarding)* [Heinanen 1999].

O *PHB EF* tenta simular um *link* dedicado com banda fixa entre dois *hosts*. A idéia seria de que os pacotes expressos devessem ser capazes de transitar pela rede como se nenhum outro pacote estivesse presente ao custo de se estabelecer rígidos limites de atrasos e praticamente sem perdas de pacotes. O *PHB EF* apresenta o maior nível de priorização de tráfego dentro de um domínio e os fluxos são encaminhados com uma taxa no mínimo igual ao da especificada pelo *SLA*, independentemente de outros fluxos. Para se evitar os possíveis danos que este tipo de política poderia trazer a outros fluxos, limites como taxa máxima de utilização e tamanho de rajada devem ser especificados, sendo os pacotes que excederem estes parâmetros descartados.

O *PHB AF* apresenta um esquema um pouco mais elaborado para gerenciar as classes de serviços, no qual são definidas quatro classes de prioridade de encaminhamento, cada uma com seus próprios recursos (espaço de buffer e largura de banda). Além disso, cada classe possui três níveis de probabilidade de descarte de pacotes no caso de um congestionamento: baixo, médio e alto, totalizando 12 classes de serviços.

Em caso de congestionamento, os nós *DiffServ* tentam evitar a perda de pacotes com menor nível de preferência de descarte. A Tabela 2.1 mostra as classes definidas pelo *PHB AF*.

Tabela 2.1. Valores *DSCP* para classes *AF* [Heinanen 1999].

Nível de Descarte	Prioridades de encaminhamento (Classe 1 tem a maior prioridade e a Classe 4 a menor)			
	Classe 1	Classe 2	Classe 3	Classe 4
Baixo	001010 (AF11)	010010 (AF21)	011010 (AF31)	100010 (AF41)
Médio	001100 (AF12)	010100 (AF22)	011100 (AF32)	100100 (AF42)
Alto	001110 (AF13)	010110 (AF23)	011110 (AF33)	100110 (AF43)

O tratamento dos fluxos como agregados (*BAs*) torna o modelo *DiffServ* escalável, mas em sua arquitetura tradicional, este modelo dificilmente alcança os requisitos de *QoS* solicitados quando a rede se encontra em uma situação de

congestionamento. Isso acontece porque as redes *DiffServ* atuais não conhecem o estado de rede interna, ou seja, em um determinado momento os roteadores de borda não têm informação sobre o que está acontecendo nos roteadores de centro. Logo, mesmo havendo uma situação de congestionamento na rede interna, os roteadores de borda continuarão encaminhando tráfego excedente para dentro da rede.

2.2.3. Mecanismos de Controle de admissão

Em redes de baixa velocidade, a utilização de algoritmos de controle de congestionamento reativo funciona de maneira satisfatória, pois as baixas taxas de transmissão das fontes permitem que as sinalizações dos algoritmos atuem após a detecção do congestionamento sem grandes prejuízos. Mas isso não funciona de forma satisfatória em redes de alta velocidade, porque o tempo de resposta entre a detecção e reação ao congestionamento é suficientemente grande para que haja grandes perdas de pacotes.

Logo, mecanismos de controle de congestionamento preventivos são essenciais para este tipo de rede. Os algoritmos de controle de admissão propõem uma solução para este problema, pois objetivam regular a entrada de fluxos em um domínio em questão, fazendo assim o controle preventivo de congestionamentos. Estes algoritmos têm o papel de controlar e fazer um ajuste entre o número de requisições que serão aceitas e o nível de utilização dos recursos da rede. Este será dito eficiente se conseguir manter o nível de qualidade de serviço para os usuários já admitidos, maximizando ao mesmo tempo, a utilização da rede.

Existem diversos algoritmos que se propõem a fazer o controle de admissão de fluxos na rede e estes podem ser divididos em dois grupos: os algoritmos de alocação não-estatística e os algoritmos de alocação estatística.

Os algoritmos de alocação não-estatística fazem a alocação de um fluxo pela sua taxa de pico e são adequados para fluxos com taxas constantes, chamados de fontes *CBR* (*Constant Bit Rate*), como é o caso de áudio e vídeos não comprimidos. Este tipo de alocação causará um enorme desperdício de banda passante quando utilizados para fluxos que apresentam grandes variações na sua taxa de transmissão (tráfego em rajadas), no qual grandes períodos de silêncio são alternados com períodos de grandes transmissões de dados. Geralmente são algoritmos simples que

exigem pouco poder de processamento por parte da unidade de controle de admissão.

Como apontado anteriormente, a alocação pela taxa de pico pode representar um grande desperdício de recursos ao se utilizar fontes com taxas variáveis de transmissão. Para solucionar este problema, os algoritmos de alocação estatística buscam utilizar os recursos da rede de forma mais eficiente, pois em aplicações como áudio e vídeo comprimidos como o *MPEG*, a taxa média de transmissão costuma ser muito menor do que a taxa de pico. Uma opção para este problema seria alocar o recurso considerando-se uma taxa maior do que a taxa média, mas menor do que a taxa de pico (esta técnica é chamada de multiplexação estatística). Como as taxas de pico dos diversos fluxos são diferentes e podem acontecer em momentos diferentes, os algoritmos podem realizar esta operação amparados por garantias estatísticas.

A complexidade dos algoritmos de alocação estatística é maior em relação aos algoritmos de alocação não-estatística, e sofrem com dois problemas principais: (1) como determinar a banda passante requerida por uma nova conexão; e (2) as decisões devem ser tomas em tempo real, o mais rápido possível, e para isso o processamento na unidade de admissão deve ser simplificado ao máximo. Segundo Rahin (1998) um algoritmo de controle de admissão deve ter as seguintes características:

- Simplicidade – os algoritmos devem ser simples para evitar um uso excessivo de processamento;
- Flexibilidade – a arquitetura deve possibilitar a adição de novos serviços que não haviam sido previstos;
- Rapidez – o algoritmo deve ser rápido o bastante para processar as requisições de admissão em tempo real;
- Eficiência – o algoritmo deve utilizar mecanismos como a multiplexação estatística para otimizar a utilização da rede;
- Efetividade – deve ser capaz de garantir os requisitos de *QoS* admitidos;

- Controle – deve haver o controle sobre o tráfego que entra no domínio sem a degradação dos fluxos já admitidos.

Vários algoritmos abordam esses problemas e podem ser divididos de forma geral em duas categorias: os baseados em modelos e os baseados em medidas. Os algoritmos baseados em modelos aceitam novos fluxos baseados em modelos de tráfegos previamente definidos. Obviamente, a eficiência desses algoritmos estará relacionada diretamente à precisão dos modelos de tráfego utilizados, mas devido à dificuldade de se obter o tráfego característico de certas aplicações, o desempenho de tais algoritmos pode ser bastante degradado.

Em oposição, temos os algoritmos baseados em medidas, que utilizam medidas feitas na rede para a tomada de decisão e são mais simples do que os algoritmos baseados em modelos. O método de medição é bastante importante para este tipo de algoritmo e existem várias técnicas para se realizar esta tarefa. A seguir são apresentados três das principias técnicas de medição: a Técnica de Janelas de Tempo, Amostra de Pontos e Média Móvel Exponencial com Peso [Guan 2001].

- Técnica de Janelas de Tempo – nesta técnica uma amostra é medida na rede a cada intervalo S. A carga da rede é definida como o maior valor das amostras em um intervalo T. Quando uma nova admissão é feita, a janela é reiniciada e o valor da carga da rede é atualizado caso o valor medido naquele momento seja maior do que o valor corrente. A Figura 2.4 mostra graficamente este processo.

- Amostra de Pontos – este método simplesmente toma amostras a cada intervalo S e torna esta medida a carga média da rede.

- Média Móvel Exponencial com Peso – também toma amostras a cada intervalo S, mas a carga média *v'* é atualizada como uma função da amostra instantânea *vi* e da carga média anterior *v*, da seguinte forma: $v' = (1-w)*v + (w)*vi$, onde *w* é uma variável que define quão rápido a média irá se adaptar as novas medidas e este varia de zero a um, ou seja, $0 < w < 1$.

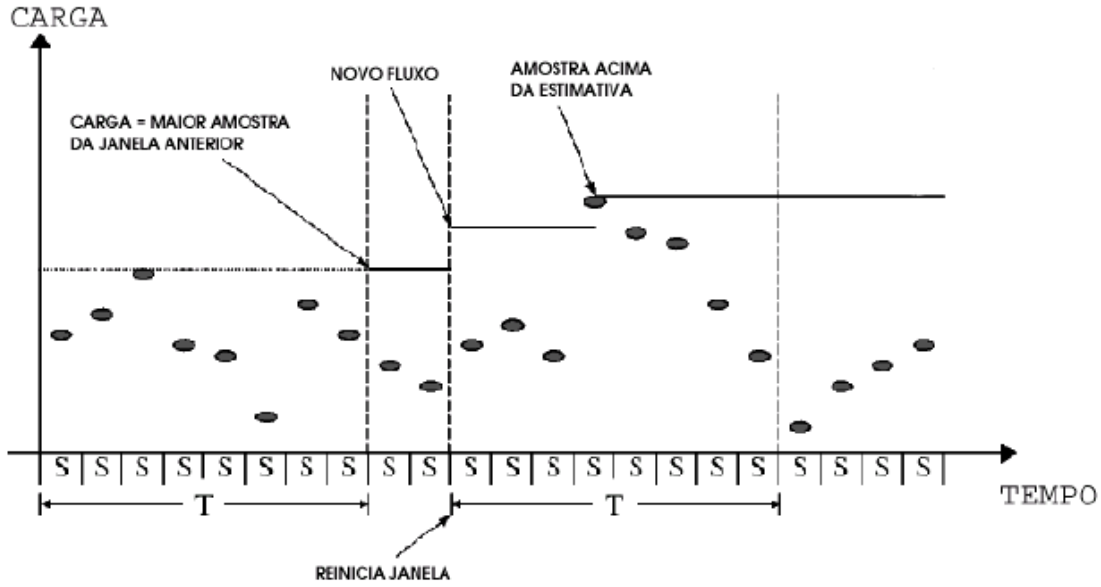

Figura 2.4. Medição da carga da rede baseada em Janelas de Tempo [Guan 2001].

Para aceitar um novo fluxo, a unidade de admissão deve negociar vários parâmetros de *QoS*. Para isso, as *RFCs* 2210 e 2211 [Wroclawski 1997a] [Wroclawski 1997b] definem um conjunto de parâmetros chamados de especificação do fluxo (*flow specification*), que são baseados no modelo balde de fichas (*token bucket*).

De forma bem resumida, este algoritmo possui um "balde de fichas" que é enchido a uma taxa definida. Em sua forma mais simples, um pacote que chegue para ser transmitido só será enviado caso haja uma ficha no balde. Serão enviados tantos pacotes quanto forem a quantidade de fichas armazenadas no balde. Se o balde for enchido a uma taxa de X fichas por segundo e existir um fluxo com uma taxa de X pacotes por segundo, este será transmitido sem problemas. Caso não haja pacotes a serem transmitidos, fichas irão ser armazenadas no balde até atingir a sua capacidade máxima. Estas fichas serão utilizadas para se permitir a suavização das rajadas de pacotes na rede.

Para se entender melhor como este mecanismo funciona, considere o exemplo mostrado por Tanenbaum (2003), que se encontra ilustrado na Figura 2.5. Na Figura 2.5 existem três fichas no balde e cinco pacotes chegam para serem transmitidos (conforme ilustrado na parte "a" da Figura 2.5). Para um pacote ser transmitido, este deve capturar e destruir uma ficha. Na parte "b" da Figura 2.5,

observa-se que três dos cinco pacotes foram transmitidos, mas os outros dois continuarão esperando mais fichas serem geradas.

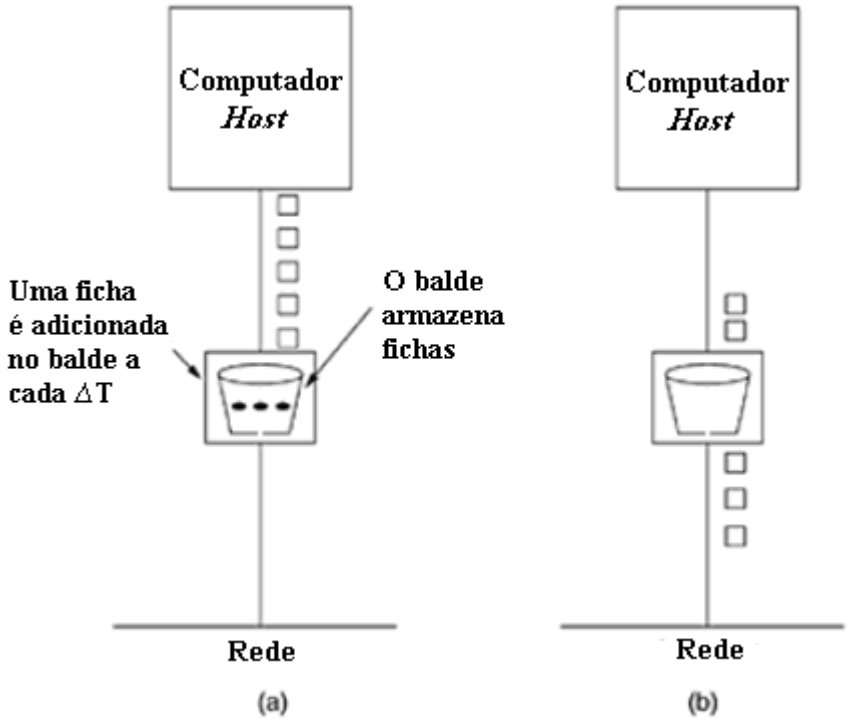

Figura 2.5. Algoritmo balde de fichas [Tanenbaum 2003].

Logo, uma vez definido em que consiste o modelo balde de fichas, podemos utilizá-lo para a definição dos parâmetros de uma especificação de fluxo (*flow specification*). Formalmente, uma especificação de fluxo consiste em cinco parâmetros, exibidos a seguir e ilustrado na Tabela 2.2:

Taxa de enchimento do balde *(Token bucket rate)* – é o número de bytes por segundo no qual o balde de fichas é enchido;

Tamanho do balde de fichas *(Token bucket size)* – é o tamanho do balde, em bytes;

Taxa de pico *(Peak data rate)* – é a taxa de transmissão máxima tolerada, em bytes por segundo;

Tamanho mínimo do pacote *(Minimum packet size)* – menor pacote tolerado, em bytes;

Tamanho máximo do pacote *(Maximum packet size)* – maior pacote tolerado, em bytes.

Tabela 2.2. Um exemplo de especificação de fluxo [Tanenbaum 2003].

Parâmetro	**Unidade**
Taxa de enchimento do balde	Bytes/s
Tamanho do balde de fichas	Bytes
Taxa de pico	Bytes/s
Tamanho mínimo do pacote	Bytes
Tamanho máximo do pacote	Bytes

2.3. *Multiple Description Coding* (*MDC*)

Multiple Description Coding é uma técnica na qual um fluxo (*stream)* de vídeo multimídia é quebrado em dois ou mais subfluxos independentes que são chamados de descrições (*descriptions)* [Venkata 2002] [Pereira 2002]. Qualquer subconjunto de descrições pode ser usado para remontar o vídeo original, onde a qualidade do vídeo final será proporcional a quantidade de descrições recebidas.

MDC é um dos métodos de codificação de vídeo que pode amenizar o problema de perda de pacotes na rede, pois diferentes descrições podem ser enviadas através de diferentes caminhos na rede, que podem possuir diferentes características em termos de *QoS*. Segundo Somasundaram (2004), o envio paralelo de descrições sobre diferentes caminhos deveria garantir uma melhor qualidade na transmissão.

Conforme discutido por Fitzek (2004), a quebra de um fluxo de vídeo em múltiplos subfluxos vem com o preço de se aumentar a quantidade de banda passante necessária para a transmissão do vídeo e este aumento se deve ao processo de codificação e as características do vídeo propriamente dito. Este *overhead* aumenta com o número de descrições utilizadas, ou seja, quanto mais subfluxos forem utilizados, maior será o *overhead* em termos de banda passante. Uma discussão detalhada pode ser encontrada em [Fitzek 2004] e foge do escopo deste trabalho.

2.3.1. Processo de codificação *MDC*

Em sua forma mais simples, o processo de codificação *MDC* consiste em quebrar um fluxo de vídeo sem compressão em dois subfluxos (em quadros ímpares e quadros pares), que seriam então codificados separadamente e independentemente, sendo que cada subfluxo teria seu próprio estado de codificação.

Como cada subfluxo possui seu próprio estado, estes deveriam ser enviados por caminhos que tivessem características diferentes em termos de perda de pacotes, retardo e variação do atraso fim-a-fim, minimizando-se as chances de se perder um dos subfluxos. O processo de codificação poderia consistir em dois codificadores separados ou em um esquema de codificação mais elaborado e este ditaria qual seria o formato da codificação, onde poderia ser utilizado o *MPEG-4*, *H.263* ou outro formato qualquer.

Savage e Collins (1999) desenvolveram um estudo baseado em medições no qual compararam o desempenho do caminho padrão de roteamento com caminhos alternativos entre dois *hosts*. Eles descobriram que em cerca de 30 a 80% dos casos, há um caminho com uma qualidade significativamente melhor do que a rota padrão. Em suas medidas, os autores utilizaram métricas como o tempo de ida e volta de um pacote, a taxa de perda de pacotes e a banda passante. Estes resultados demonstram que a utilização de *MDC* pode ser uma alternativa promissora para uma rede sem garantias de entrega de pacotes como a Internet.

2.3.2. Processo de decodificação *MDC*

De forma similar ao processo de codificação, o processo de decodificação pode ser feito utilizando-se dois decodificadores independentes ou um decodificador só que se alterne entre os quadros ímpares e pares recebidos. Se não houver erro na recepção dos quadros ímpares e pares, estes serão decodificados de forma independente e depois serão intercalados para formar o fluxo de vídeo original.

Se um dos subfluxos apresentar um erro na recepção, o outro pode ser utilizado para remontar o vídeo enviado com a metade da taxa de quadros por segundo do vídeo original. Este erro produziria uma redução temporária na taxa de quadros por segundo, contudo não haveria distorções, o que seria preferível ao caso convencional onde o decodificador seria obrigado a congelar o vídeo ou utilizar algum método de interpolação dos quadros recebidos para estimar os quadros perdidos, o que poderia levar a uma significante distorção no vídeo recebido.

Apostolopoulos (2001) propõe um esquema de decodificação que se aproveita do fato dos subfluxos possuírem estados de codificações independentes e utiliza este recurso para a reconstrução dos quadros que porventura cheguem corrompidos. Para isso o esquema proposto provê acesso aos quadros recebidos

anteriormente e aos quadros futuros para corrigir um quadro que chegue corrompido, como é mostrado na Figura 2.6.

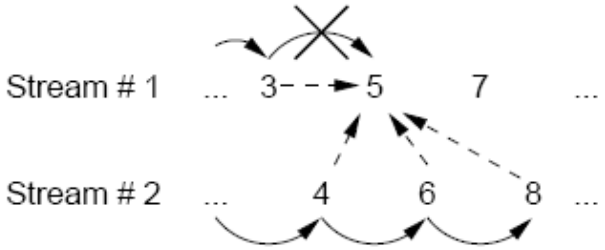

Figura 2.6. Reconstrução de um quadro perdido [Apostolopoulos 2001].

Como mostra a Figura 2.6, o quadro número 5 chegou corrompido ao se receber o subfluxo ímpar, mas o subfluxo par foi recebido com sucesso. Neste caso os quadros número 3, 4, 6 e 8 podem ser utilizados para se realizar a reconstrução (através de algum método de interpolação) do quadro número 5.

2.3.3. Outras técnicas de transmissão de mídias contínuas

Existem outras técnicas que se propõe a transmissão de mídias contínuas sobre rede com perda de pacotes. Uma delas é chamada de codificação em camadas (*layered coding*) e consiste na transmissão de fluxo de vídeo através de camadas, sendo a primeira camada, denominada de base, e as camadas subseqüentes aumentam a qualidade do vídeo à medida que são recebidas. Contudo, o vídeo pode ser completamente perdido se houver um erro no envio da camada base.

Um método comum é adicionar redundância ao fluxo multimídia transmitido. Esta técnica é chamada *FEC* (*Forward Error Correction*) e consiste na transmissão de informações redundantes em cada pacote enviado em uma seqüência de pacotes [Hardman 1995] [Bolot 1999] [Sanneck 2000]. Um pacote perdido pode ser recuperado através de cópias que estão em pacotes subseqüentes, que neste caso devem ser recebidos com sucesso. Neste esquema, a recuperação de pacotes perdidos é realizada ao custo de se aumentar a latência. Como a perda de um pacote pode vir acompanhada de uma rajada de perda de pacotes, a eficiência deste método pode diminuir de forma significativa [Bolot 1993] [Bolot 1999]. Para resolver este problema, as informações de redundância podem ser adicionadas a pacotes distantes no tempo, mas esta solução aumenta ainda mais o retardo que este método apresenta.

2.4. Diversidade de Caminhos

Segundo Teixeira (2003), diversidade de caminhos descreve o número de caminhos distintos entre dois *hosts* e suas características como latência, banda passante e tamanho (em *hops*). Conforme discutido em [Apostolopoulos 2001], existem duas formas de se implementar diversidade de caminhos na atual infra-estrutura da Internet: (1) através do uso de roteamento de origem (*source routing)* e (2) utilizando uma estrutura de *relays.*

No caso do uso de roteamento de origem (*source routing*), bastaria se definir uma seqüência de nós (na verdade uma seqüência de endereços IP) que cada pacote deveria atravessar. Poderiam ser fornecidos a seqüência inteira de endereços (*loose source routing*) ou um subconjunto de endereços IP (*strict source routing*). A diferença é que no primeiro caso os pacotes deveriam passar apenas pelos roteadores definidos pela lista de endereços, enquanto que no segundo caso a lista de endereços conteria apenas alguns dos roteadores que devem encaminhar os pacotes. Contudo, o uso do roteamento de origem apresenta diversos problemas que são apresentados a seguir:

1. A definição de um caminho envolve a definição de uma seqüência de endereços IP e esta informação por si só revela detalhes internos de uma rede que podem levar a problemas de segurança. Por esse motivo, a maioria dos roteadores da Internet vem com esta funcionalidade desligada por motivos de segurança.
2. Mesmo que todos os roteadores da Internet habilitassem a funcionalidade de roteamento de origem, o problema agora seria como escolher um caminho na Internet, pois boa parte da Internet é desconhecida. Além disso, uma vez escolhida uma rota, esta poderia mudar dinamicamente o que levaria à necessidade de uma redefinição do caminho.

Apesar desses problemas, nada impediria o uso do roteamento de origem dentro de uma intranet, pois o domínio seria menor e se teria total acesso as informações necessárias.

Outra alternativa para se implementar o método de diversidade de caminhos (*path diversity)* é utilizar uma estrutura de *relays* [Apostolopoulos 2001]. Neste

método, o pacote IP a ser enviado é encapsulado dentro de outro pacote IP com o destino apontando para o *relay* a ser utilizado, que por sua vez tem o papel de obter o pacote original e enviar novamente para a rede (para o destino final ou um novo *relay*), como é mostrado na Figura 2.7. Esta técnica é chamada por alguns autores de tunelamento [Tanenbaum 2003].

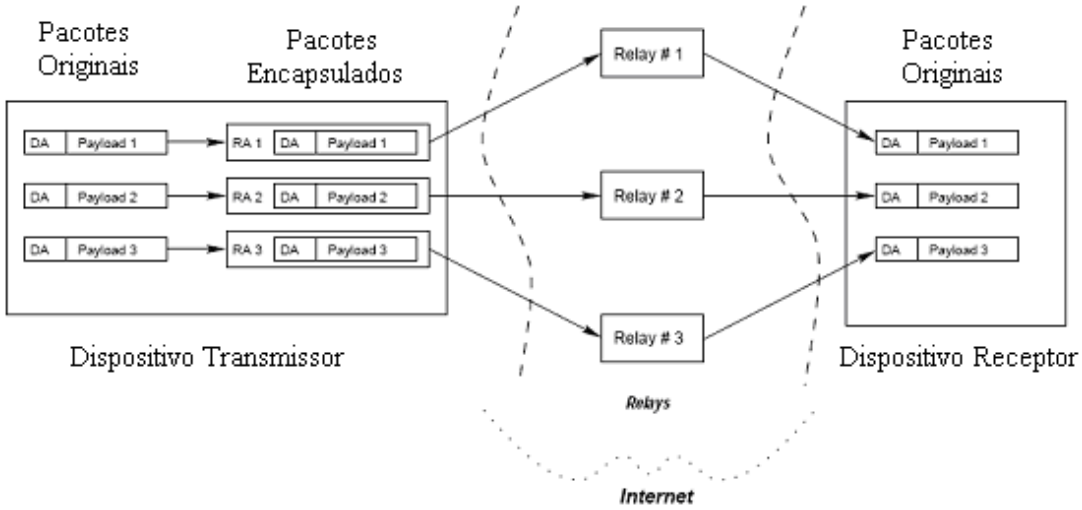

Figura 2.7. Uma estrutura de *relays* na Internet [Apostolopoulos 2001].

Para exemplificar como isso funcionaria, imagine a utilização de *MDC* para a quebra de um fluxo de vídeo em dois subfluxos que seriam enviados de A para B, sendo que existe um *relay* C disponível que define uma rota alternativa diferente da rota padrão, ou seja, o *relay* C não faria parte do melhor caminho entre A e B. Então A poderia enviar o primeiro subfluxo da forma usual, simplesmente enviando os pacotes para a rede. Já o segundo subfluxo poderia ser enviado através de uma rota alternativa, bastando para isso que A encapsule os pacotes em outro pacote IP com o destino apontando para C. O trabalho do *relay* C seria o de obter o pacote original e enviá-lo para B. O uso de *relays* vem se tornando uma alternativa promissora, pois não exige mudança na infra-estrutura da Internet atual.

Apostolopoulos (2001) propõe um sistema para comunicação de vídeo sobre uma rede com a possibilidade de perda de pacotes como é o caso da Internet. O sistema mostrado na Figura 2.8 é composto de duas partes: (1) um mecanismo de codificação e de decodificação chamado *multiple state video encoding and decoding* e (2) um sistema de transmissão de vídeo usando diversidade de caminhos, no qual o vídeo é quebrado em múltiplos subfluxos utilizando-se *MDC* juntamente com uma infra-estrutura de *relays*. A inovação do trabalho de Apostolopoulos (2001) se

encontra no sistema de decodificação feito no receptor, conforme já explicado anteriormente, que utiliza informações dos múltiplos fluxos de vídeo para a recuperação de quadros (*frames*) que porventura cheguem danificados.

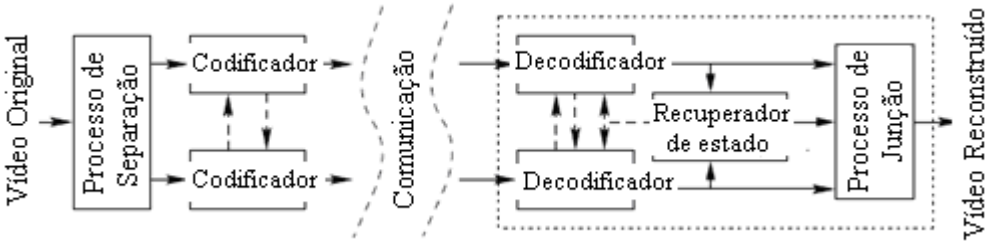

Figura 2.8. Sistema para comunicação de vídeo sobre uma rede IP [Apostolopoulos 2001].

Ainda segundo Apostolopoulos (2001) o método de diversidade de caminhos apresenta as seguintes vantagens:

- aplicações fim-a-fim vêem um "caminho virtual" que possui menor variabilidade em termos de parâmetros de qualidade de serviço se comparado com qualquer outro caminho individual;
- reduz o tamanho da rajada de perda de pacotes (onde vários pacotes seguidos são perdidos);
- a probabilidade de haver uma falha (onde quase nenhum ou nenhum pacote é entregue) diminui gradativamente, pois isso ocorrerá somente se todos os caminhos falharem simultaneamente;

Encontra-se em [Ribeiro 2004] uma análise sobre a eficiência do uso do método de diversidade de caminhos em função do número de caminhos utilizados. Uma das conclusões do trabalho foi a de que a resposta depende da sensibilidade da aplicação a pequenas e longas rajadas de perda. O aumento do número de caminhos não é garantia de que o método melhore. Segundo o autor, ao se utilizar caminhos independentes e identicamente distribuídos, quando aumentamos o número de caminhos a probabilidade de rajadas grandes diminui, no entanto, a probabilidade de rajadas menores aumenta. Como a probabilidade de rajadas grandes em geral é pequena, pode ser que não seja eficiente a utilização de muitos caminhos.

2.5. O *framework EvalVid*

O *framework EvalVid (A Framework for Video Transmission and Quality Evaluation)* [Klaue 2003] foi utilizado neste trabalho para a implementação da técnica de *MDC*. O *EvalVid* apresenta uma estrutura completa para quebra, codificação, transmissão, recepção, decodificação e avaliação da qualidade dos vídeos recebidos, como é mostrado na Figura 2.9. Cada componente é explicado nos itens a seguir:

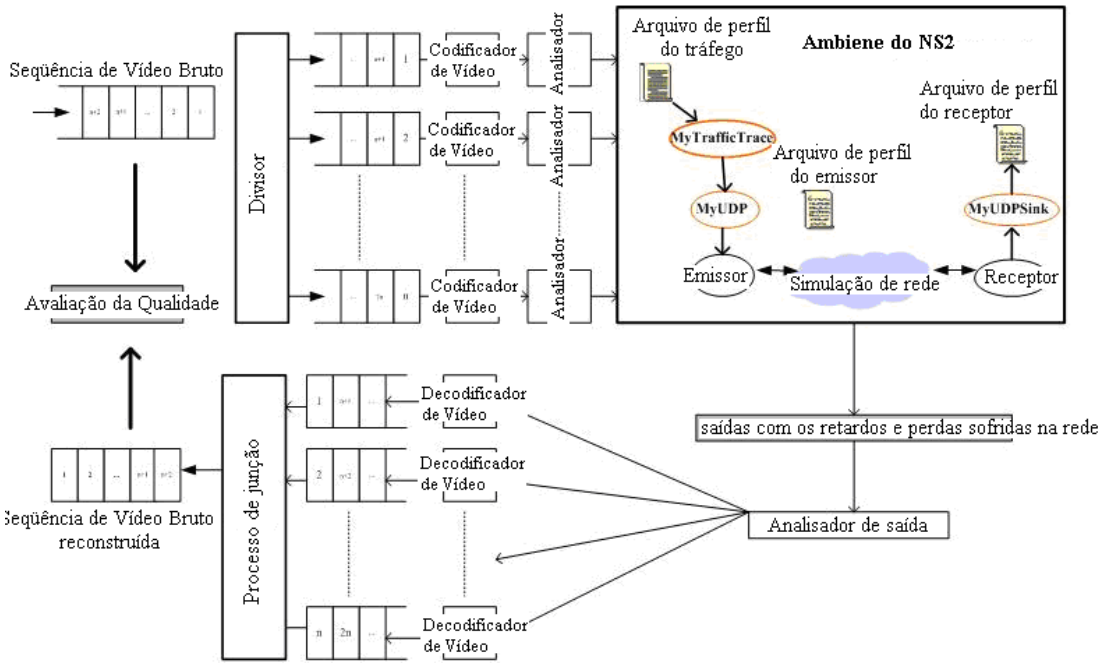

Figura 2.9. Esquema de funcionamento do framework *EvalVid* [Chih-Heng 2006].

- Seqüência de Vídeo Bruto *(Raw Video Sequence)* – representa o vídeo sem nenhum esquema de compressão. Este framework aceita vídeos no formato *YUV QCIF (Y* é um componente que define o brilho e UV são componentes que definem as cores, *QCIF- Quarter Common Intermediate Format*) ou *YUV CIF* (*CIF- Common Intermediate Format*).

- Divisor *(Spliter)* – componente responsável por quebrar o vídeo sem compressão e criar *i* (dois ou mais) subfluxos de vídeo independentes, de tal forma que os subfluxos contenham os quadros n, 2n, 3n e assim por diante (ou seja, sempre alternando entre quadros pares e ímpares).

- Codificador de Vídeo *(Video Encoder)* – faz a compressão do vídeo em um formato específico (MPEG4, H.263, H.264, etc.). Neste trabalho foi utilizado o formato MPEG4.

- Analisador *(Parser)* – é um programa que lê cada subfluxo de vídeo codificado e gera um arquivo de perfil de tráfego que contém o *id* (número de identificação) do quadro a ser transmitido, o tipo do quadro (I, B, P, etc.), o tamanho do quadro em bytes e o instante em que deve ser transmitido.

- *MyTrafficTrace* – é uma aplicação desenvolvida no ns2 (*Network Simulator 2*) que lê o arquivo de perfil de tráfego (criado pelo Analisador) e gera os pacotes correspondentes a serem transmitidos. Depois de gerados, estes pacotes são enviados para a camada de transporte no instante indicado pelo arquivo de perfil de tráfego.

- *MyUDP* – consiste em um agente *UDP* (na verdade um agente *RTP - Real-time Transport Protocol*). Basicamente recebe os pacotes enviados pela aplicação *MyTrafficTrace* e transmite para a rede. Este agente também gera um arquivo texto chamado de *sd*, no qual registra o tempo de envio, o *id* e o tamanho de cada pacote transmitido.

- MyUDPSink – consiste em outro agente que recebe os pacotes enviados pelo agente *MyUDP*. Este agente também gera um arquivo texto chamado *rd*, no qual registra o tempo de chegada, o *id* e o tamanho dos pacotes recebidos.

- Analisador de saída *(Evaluate Trace)* – após a simulação realizada no ambiente do ns2, lê as entradas do arquivo *sd* (gerado pelo emissor) e do arquivo *rd* (gerado pelo receptor) e calcula os pacotes perdidos e os retardos sofridos por cada pacote. Com estas informações e com os subfluxos dos vídeos originais transmitidos, cria os arquivos de vídeo que seriam encontrados do lado do receptor, com todas as possíveis distorções provenientes das perdas e retardos dos pacotes.

- Decodificador de Vídeo *(Video Decoder)* – faz a descompressão dos subfluxos de vídeos.

- Processo de junção *(Merger)* – após o processo de decodificação realizado no passo anterior, faz a junção dos subfluxos de vídeos que o receptor recebeu com as possíveis distorções, para montar o vídeo completo com todos os quadros. Como o número total de quadros no vídeo completo recebido tem que ser igual ao número de quadros do vídeo original, caso algum quadro dos subfluxos recebidos tenha sido perdido, o processo de junção copia o último quadro recebido com sucesso até que um quadro correto seja achado.
- Avaliação da Qualidade – Através desse módulo é possível fazer uma análise da qualidade comparando-se o vídeo original e o vídeo recebido.

O componente de avaliação da qualidade envolve o julgamento visual dos vídeos recebidos, que consiste em uma avaliação subjetiva, pois diferentes usuários poderiam ter opiniões diferentes sobre o requisito de qualidade dos vídeos. Como este tipo de avaliação está fora do escopo deste trabalho, este componente não foi utilizado.

Capítulo 3

Disciplinas de controle de admissão

Disciplinas de controle de admissão são mecanismos de controle preventivo de congestionamento e objetivam estabelecer como será a entrada de novos fluxos de dados na rede. Logo, uma disciplina de controle de admissão limita o volume de tráfego que transita na rede, mantendo os requisitos de qualidade de serviço previamente acordados.

Este capítulo tem como objetivo a definição de disciplinas de controle de admissão que funcionarão em conjunto com um negociador de banda passante (*BB – Bandwidth Broker*) para limitar a entrada de fluxos multimídia na rede. Serão estabelecidos também critérios de avaliação para estas disciplinas propostas.

3.1. Funcionamento do MDCAC

O *MDCAC* utiliza o *MDC (Multiple Description Coding)* para a divisão do fluxo multimídia em diversos subfluxos. Com o uso do *MDC* [Venkata 2002], torna-se possível a quebra do vídeo original em diversos subfluxos de vídeo independentes, de tal forma que mesmo que partes de um subfluxo de vídeo sejam perdidas, os outros subfluxos ainda possam ser utilizados para a reconstrução do vídeo original.

O mecanismo de controle de admissão utiliza um *BB* (*Bandwidth Broker*) centralizado [Nichols 1999] conforme mostrado na topologia da Figura 3.1. A admissão de um fluxo é feita da seguinte forma:

1) toda vez que o nó A precisar enviar um fluxo de dados para o nó B, o mesmo enviará uma solicitação de admissão para o *BB* indicando a quantidade de banda passante requerida, o tempo de início da transmissão e tempo de fim da transmissão;
2) o *BB* analisa a solicitação utilizando uma das variações de funcionamento que serão definidas e discutidas na seção 3.3;
3) o fluxo poderá ser aceito ou rejeitado. O *BB* mantém estatísticas que servirão de base para a análise dos algoritmos a serem avaliados.

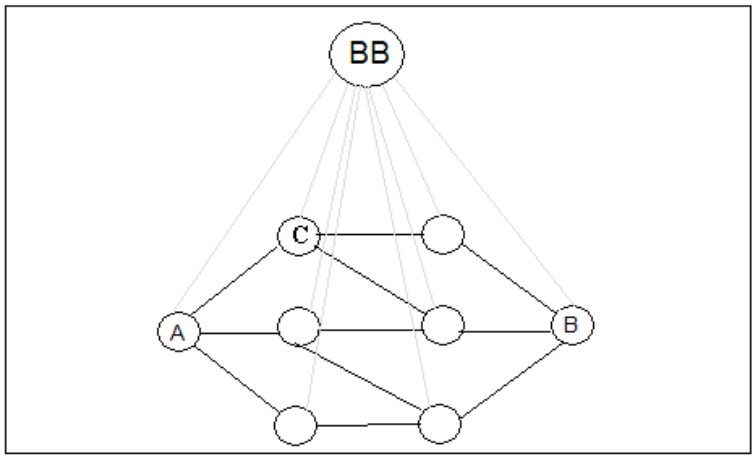

Figura 3.1. Topologia controlada por um *Bandwidth Broker*.

3.2. Método de avaliação

O objetivo deste trabalho foi investigar o *MDCAC* nos seguintes quesitos:

1. número de requisições admitidas;
2. eficiência na utilização da rede (definido como *profit*);
3. taxa de perda de pacotes.

Com base nestes quesitos, o interesse foi investigar se a abordagem de admitir um fluxo utilizando mais de um caminho tem alguma vantagem em relação à utilização de apenas um caminho. Também houve um interesse de se saber qual é o ganho em relação a uma abordagem que utiliza apenas o caminho definido pelo algoritmo de roteamento. O *profit* se refere à utilização da rede e seu valor será calculado multiplicando-se os valores das durações das requisições aceitas pela banda passante requerida por cada uma.

3.3. Variações de funcionamento

O *MDCAC* faz o controle de admissão utilizando uma das variações de funcionamento conforme mostrado nos itens a seguir. A notação utilizada é exibida na Tabela 3.1.

Tabela 3.1. Notação utilizada nos algoritmos de controle de admissão.

Notação	Significado
R(*Bw*, *S1*, *S2*)	Requisição de admissão de fluxo multimídia com banda passante *Bw* com tempo de início *S1* e tempo de término *S2*
Bi	Banda passante disponível no caminho *i*

O *BB* mantém informações da banda passante disponível em cada *link* da topologia controlada. Para cada *link*, o *BB* mantém uma estrutura de dados que armazena a banda passante disponível no *link* em questão para cada instante de tempo da simulação, de uma forma discreta. Por exemplo, suponha um tempo de simulação de 10 segundos e que a banda passante controlada entre A e C da Figura 3.1 seja de 2 Mbps. Para este cenário, o *BB* manterá uma estrutura de dados com 10 entradas, uma para cada segundo da simulação, com um valor inicial de 2 Mbps. Caso receba uma requisição R(1, 4, 7) de A para C, irá verificar se existe banda passante disponível no *link*, checando as entradas de 4 a 7 da estrutura de dados. Para isso verificará se o valores das entradas de 4 a 7 são maiores do que 1 Mbps. Se isso se verificar, o *BB* irá aceitar a requisição e irá alterar a banda passante disponível entre A e C alterando-se os valores das entradas de 4 a 7 com os valores antigos subtraídos do valor alocado, ou seja, os novos valores seriam de 1 Mbps.

A seguir são apresentadas as variações de funcionamento do *MDCAC*. Logo, dado uma requisição R(*Bw, S1, S2*) de um nó A qualquer a um nó B qualquer na rede, cada variação funcionará conforme definido:

3.3.1. Variação número 0

```
Seja i o caminho fornecido pelo algoritmo de roteamento entre A e B.
Se Bi >= Bw no intervalo entre S1 e S2:
        Aceite R passando por esse único caminho i;
Senão:
        Rejeite R;
```

Figura 3.2. Variação número 0 do *MDCAC*.

A variação número 0 (V0) não quebra o fluxo multimídia e só utiliza a rota fornecida pelo algoritmo de roteamento. Trata-se do algoritmo padrão de roteamento encontrado na Internet e servirá de base para a comparação com as outras variações de funcionamento do *MDCAC*. Conforme explicado anteriormente, para cada requisição R(*Bw, S1, S2*), o *BB* irá verificar na sua estrutura de dados interna se existem valores de banda passante maiores do que *Bw* para o intervalo discreto de *S1* a *S2* para os *links* entre A e B. Caso exista, R será aceito e a estrutura de dados interna será atualizada.

3.3.2. Variação número 1

Seja *i* o caminho com maior banda passante de A para B.

Se *Bi* >= *Bw* no intervalo entre *S1* e *S2* então:

    Aceite R passando por esse único caminho *i*;

Senão:

    Rejeite R;

Figura 3.3. Variação número 1 do *MDCAC*.

A variação número 1 (V1) também não quebra o fluxo multimídia e pode utilizar qualquer caminho (inclusive a rota padrão) entre A e B para avaliar R. Também será utilizada de base para a comparação com as outras variações de funcionamento do *MDCAC*. Esta variação procura identificar qual seria o aumento na utilização dos recursos da rede e no número de requisições aceitas caso o melhor caminho entre A e B fosse determinado dinamicamente tendo como métrica a banda passante disponível no caminho.

3.3.3. Variação número 2

Quebre o fluxo da requisição R em dois fluxos independentes de mesma banda passante *Bw /2*;

Se houver dois ou mais caminhos quaisquer com banda passante *Bw/2* disponível entre A e B, selecione dois caminhos *i, j* que disponibilizarem maior banda passante e então:

    Aceite R e envie cada fluxo independente pelos dois caminhos *i, j* selecionados;

Senão:

    Rejeite R;

Figura 3.4. Variação número 2 do *MDCAC*.

A variação número 2 (V2) sempre quebra um fluxo em duas partes iguais e procura por duas rotas diferentes para admitir estes dois subfluxos. Esta variação procura identificar se existe alguma vantagem na utilização dos recursos da rede e no número de requisições aceitas ao se utilizar sempre os dois melhores caminhos disponíveis entre A e B, tendo como métrica a banda passante disponível nos caminhos.

3.3.4. Variação número 3

Seja *i* o caminho com maior banda passante de A para B.

Se Bi >= Bw no intervalo entre *S1* e *S2* então:

    Aceite R passando só por esse único caminho *i*;

Senão:

    Se houver dois ou mais caminhos quaisquer com banda passante *Bw/2* disponível entre A e B, selecione dois caminhos *i, j* que disponibilizarem maior banda passante e então:

        Quebre o fluxo em dois fluxos independentes com banda passante *Bw/2* e envie pelos dois caminhos *i, j* selecionados;

    Senão:

        Rejeite R;

Figura 3.5. Variação número 3 do *MDCAC*.

Já a variação número 3 (V3) procura saturar os recursos da rede tentando primeiro a transmissão do fluxo utilizando apenas um caminho alternativo entre A e B, que no caso seria sempre o melhor caminho em termos de banda passante disponível. Caso não seja possível, este quebra o fluxo em dois subfluxos e tenta a transmissão utilizando os dois melhores caminhos disponíveis entre A e B, tendo como métrica a banda passante disponível nos caminhos.

3.3.5. Variação número 3A

Seja *i* o caminho com maior banda passante de A para B.

Se *Bi* >= *Bw* no intervalo entre *S1* e *S2* então:

Com uma probabilidade p faça:

Aceite R passando por esse único caminho *i*;

E com uma probabilidade q = 1 – p faça:

Se houver dois ou mais caminhos quaisquer com banda passante *Bw/2* disponível entre A e B, selecione dois caminhos *i, j* que disponibilizarem maior banda passante e então:

Aceite R e envie cada fluxo independente pelos dois caminhos *i, j* selecionados;

Senão:

Rejeite R;

Senão:

Se houver dois ou mais caminhos quaisquer com banda passante *Bw/2* disponível entre A e B, selecione dois caminhos *i, j* que disponibilizarem maior banda passante e então:

Aceite R e envie cada fluxo independente pelos dois caminhos *i, j* selecionados;

Senão:

Rejeite R;

Figura 3.6. Variação número 3A do *MDCAC*.

A variação número 3A (V3A) usa uma estratégia probabilística para aceitar os fluxos de vídeo. Se no início, não achar um caminho com banda passante suficiente para a transmissão do fluxo do vídeo através de apenas um caminho, este quebra o vídeo e tenta o envio através de dois caminhos, caso contrário, se no início for possível a transmissão através de apenas um caminho, este o faz com uma probabilidade p e com uma probabilidade de (1 - p) quebra o vídeo e tenta o envio através de dois caminhos. Também utiliza como métrica a banda passante disponível nos caminhos.

3.3.6. Variação número 4

Quebre o fluxo da requisição R em dois fluxos independentes de mesma banda passante *Bw /2*;

Defina um nível de aceitação NA = 1,2 * (*Bw /2);*

Se houver dois ou mais caminhos quaisquer com banda passante com valores maiores ou iguais a NA entre A e B, selecione dois caminhos *i, j* que disponibilizarem maior banda passante e então:

Aceite R e envie cada fluxo independente pelos dois caminhos *i, j* selecionados;

Se não

Rejeite R;

Figura 3.7. Variação número 4 do *MDCAC*.

Como se observa na Figura 3.7, a variação número 4 (V4) só aceita uma nova requisição se houver dois caminhos que disponibilizem um valor de banda passante maior ou igual ao valor requerido acrescido de 20%, mas a alocação do recurso é feita pelo valor nominal da requisição. Por exemplo, a variação quatro (V4) só aceitaria uma solicitação de requisição de 526 Kbps se fosse possível encontrar dois caminhos com banda passante de pelo menos 315,6 Kbps disponível, mas uma vez aceita, a requisição seria alocada usando o valor nominal de 263 Kbps. O nível de aceitação (NA) foi definido através de simulações preliminares, na qual testou-se valores entre 10 e 30 %, obtendo-se um melhor resultado utilizando-se um valor de 20%.

Todas as variações do MDCAC foram implementadas no ambiente do ns2 e comparadas seguindo os critérios adotados na seção 3.2. O próximo capítulo mostra com detalhes as simulações feitas e os resultados obtidos para cada variação do MDCAC.

Capítulo 4

Simulações e resultados

Este capítulo define como as simulações foram realizadas e exibe detalhes de todos os *frameworks* que serviram de suporte para a realização das simulações. Dentre os *frameworks* de suporte utilizados, destacam-se o processo de geração da carga de requisições e o processo de codificação dos vídeos. Também é apresentada a topologia utilizada, assim como a análise dos resultados obtidos.

4.1. Metodologia da Simulação

O ambiente definido para as simulações foi o ns2. Para a análise das variações de funcionamento do *MDCAC,* foram definidas duas simulações diferentes e as chamaremos de simulação 1 e simulação 2. A primeira simulação teve como objetivo a análise do *profit* e do número de requisições aceitas. Por se tratar de uma simulação simples em termos de processamento, envolveu um grande número de requisições e um tempo de simulação longo. Esta primeira simulação só envolvia a análise das informações geradas pelo *BB*.

Já a segunda simulação teve como objetivo a análise da perda de pacotes na rede. Por se tratar de uma simulação mais complexa em termos de processamento, envolveu um pequeno número de requisições em um curto intervalo de simulação. Esta simulação foi mais complexa porque envolveu a transmissão de fluxos de vídeo no ambiente de simulação e a análise de perda de pacotes em cada nó da rede. À medida que os resultados forem sendo apresentados, mais detalhes destas duas simulações serão descritos.

4.2. Gerador de carga de requisições

Uma vez definidas as variações de funcionamento, o próximo passo foi definir uma carga de requisições que simulasse de forma mais real possível um servidor de mídia verdadeiro. Para isso foi utilizado o *framework* definido por [Tang 2007], que consiste em um gerador de carga de requisições (*workload generation*) baseado em modelos estatísticos. Esta ferramenta foi desenvolvida baseada na carga de requisições geradas por dois servidores de mídia da *HP (Hewlett-Packard),* um que servia uma rede corporativa e outro que servia uma rede acadêmica.

Trata-se de um gerador sintético de carga de trabalho, no qual é possível definir vários parâmetros que irão influenciar diretamente o perfil da carga de requisições geradas. Dentre estes parâmetros, os principais são:

- duração – representa a duração dos arquivos armazenados no servidor de mídia;
- taxa de transmissão – representa a taxa média de transmissão de um arquivo;
- popularidade de acesso – define a quantidade de acessos a um arquivo qualquer dentro de um intervalo de tempo;
- prefixo – representa o tempo decorrido entre o início e término da execução do arquivo, pois muitas vezes o usuário não vê o vídeo completo;
- taxa de introdução de arquivos – representa com que taxa novos conteúdos são introduzidos no servidor de mídia, pois é esperado que vídeos antigos saiam de cena e que novos vídeos sejam adicionados;
- tempo de vida – define como a popularidade de um arquivo muda durante um intervalo de tempo determinado, pois é esperado que um arquivo recentemente adicionado tenha uma popularidade maior no início e que esta vá diminuindo com o passar do tempo;
- padrão de acesso diário – define como o número de acessos a um arquivo varia durante um dia (ou período de tempo especificado).

Todos estes parâmetros são especificados através de um arquivo de configuração que serve de entrada para o gerador de carga de requisições. A saída do gerador de carga é um arquivo texto com várias informações, dentre as quais o tempo de início das requisições, o prefixo e a banda passante serviram de entrada para o *MDCAC*. A Tabela 4.1 mostra um exemplo de uma saída gerada pela ferramenta.

Tabela 4.1. Formato do arquivo de saída do gerador de carga de requisições.

Tempo de Início (s)	*Id* do fluxo	Prefixo (s)	Tamanho (s)	Banda Passante (Kbps)
60000	1	26	91	274.0
60001	8	33	173	372.0
60003	4	19	277	526.0

4.3. Topologia utilizada

A topologia utilizada nas simulações é mostrada na Figura 4.1 e consiste de um domínio *DiffServ* tendo-se os nós A e B como nós de fronteira (*edge*) e os nós C, D, E, R1 e R2 como nós de centro (*core*). Dois fatores foram determinantes para a escolha de apenas dois caminhos alternativos entre A e B:

- conforme já discutido na seção 2.4, o aumento do número de caminhos não é uma garantia de que o método de diversidade de caminhos melhore, pois quando se aumenta o número de caminhos alternativos, a probabilidade de rajadas grandes diminui, no entanto, a probabilidade de rajadas menores aumenta [Ribeiro 2004];
- segundo Apostolopoulos (2001) a eficiência de compressão de um fluxo de vídeo diminui conforme o número de descrições aumenta, ou seja, quanto mais subfluxos forem utilizados menor será a eficiência de compressão de cada subfluxo.

Com base nos dois argumentos acima, se optou pela utilização de dois caminhos alternativos pela simplicidade de implementação, atingindo-se, ao mesmo tempo, os objetivos de se diminuir a probabilidade de rajadas de perdas de pacotes e de se manter um nível de eficiência de compressão adequado para os subfluxos. Outros autores também optaram pela utilização de dois caminhos alternativos como Apostolopoulos (2001) e Golubchik (2002).

Para os fluxos multimídia utilizou-se uma classe do tipo *Expedited Forwarding* (encaminhamento expresso), pois a intenção era dar uma grande prioridade para este tipo de fluxo em relação a outros tipos de fluxos. Caso um fluxo

exceda a banda passante média definida, nenhuma remarcação foi definida, uma vez que se queria avaliar os algoritmos de admissão e não a influência de políticas *DiffServ*. A configuração da topologia mostrada na Figura 4.1 se justifica porque o ns2 utilizará o caminho ACB como rota definida pelo algoritmo de roteamento, por ser a mais curta em número de *hops* (esta rota será denominada rota padrão). Outras rotas alternativas serão definidas utilizando-se os *relays* R1 e R2, conforme mostra a Figura 4.1.

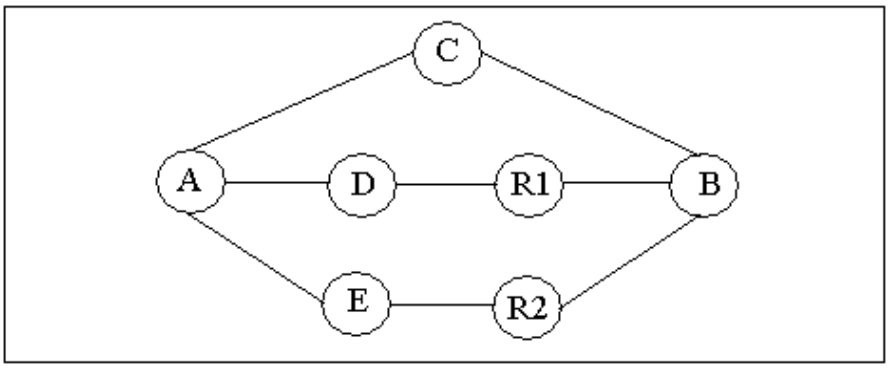

Figura 4.1. Topologia utilizada para as simulações.

Estes *relays* são agentes desenvolvidos para o ns2 e atuam na camada de rede formando uma estrutura de *relays*. Para se definir uma rota alternativa, o nó emissor A deve encapsular o pacote IP original dentro de outro pacote IP com o destino apontando para o *relay* a ser utilizado, que por sua vez tem o papel de obter o pacote original e enviar novamente para a rede. Como o pacote original tem como destino o nó B, este será roteado para B.

Para a simulação de uma rede *DiffServ* foi utilizada a implementação desenvolvida pela *Nortel Networks* [Pieda 2000]. Utilizando este *framework* é possível definir filas físicas, filas virtuais, escalonadores de filas, políticas de policiamento e políticas de remarcações de pacotes em cada nó *DiffServ*.

Neste trabalho, foram definidas duas filas físicas contendo duas filas virtuais em cada nó, totalizando quatro filas. A intenção desta decisão foi utilizar a fila física número 1 para fluxos multimídia e a fila física número 2 para outros tipos de fluxos. Contudo, por motivos de simplicidade, só foram utilizados fluxos multimídia neste trabalho, o que deixou a fila física número 2 sem uso. Foram definidas quatro marcações que são utilizadas pelo classificador do nó *DiffServ* para selecionar qual fila o pacote irá ser enviado. O esquema montado para as filas é mostrado na Figura 4.2.

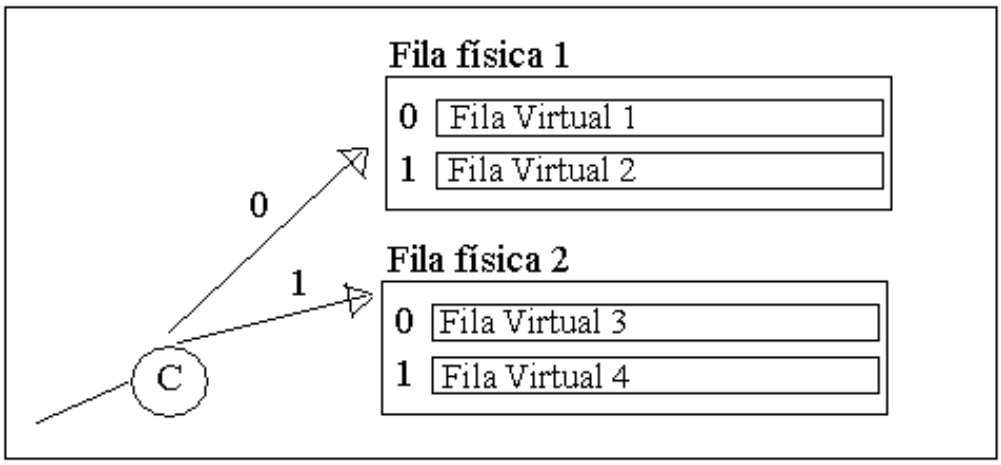

Figura 4.2. Estrutura de filas dos nós *DiffServ*.

Como só foram utilizados fluxos multimídia nas simulações, todos os fluxos de vídeo estão sendo enviados para a fila 01 (significando a fila virtual número 2 dentro da fila física número 1) conforme mostrado na Figura 4.2. As outras filas estão sem uso, pois não definimos nenhum outro tipo de tráfego. Foi utilizada uma política de policiamento do tipo balde de fichas *(*chamada *tokenBucketPolicer)* nos nós de fronteira (*edge)*. Para este policiamento, deve-se definir a taxa de comprometimento do fluxo, chamada de *CIR (Committed Information Rate*) e o tamanho da rajada, chamado de *CBS (Committed Burst Size*). A taxa de comprometimento é informada pelo próprio fluxo ao *BB* no momento em que este negocia sua admissão e o tamanho da rajada foi definido com um valor de 5000 bytes. A definição destes dois parâmetros não tem nenhum efeito prático nas simulações, pois se um fluxo exceder seu perfil de tráfego definido, nenhuma marcação ou remarcação foi definida.

Um parâmetro importante nas simulações foi o tamanho das filas e foi definido através de simulações preliminares, ou seja, por tentativa e erro buscou-se um tamanho de fila que não fosse nem muito grande a ponto de não haver perdas e nem muito pequeno a ponto de haver muitas perdas. Foram feitas 10 simulações com valores de tamanho de fila variando entre 200 e 500 pacotes. No final deste processo uma fila com o tamanho de 400 pacotes foi escolhida para servir de parâmetro para as simulações.

Foi utilizada uma política de escalonamento entre as filas denominada *MDRR (Modified Deficit Round Robin)* desenvolvido por [Bouras 2007]. Caso um fluxo exceda a taxa de comprometimento definida, nenhuma remarcação foi

utilizada, logo o fluxo continuará na mesma fila. O descarte de pacotes é feito pelo controle de congestionamento definido pela fila *RED (Random Early Detection)*, definida pela implementação da *Nortel* [Pieda 2000], sendo utilizados os parâmetros *default* definidos por este *framework*.

4.4. Processo de codificação dos vídeos

Os vídeos utilizados neste trabalho foram codificados e armazenados em duas formas: (1) disponibilizados em apenas um fluxo de vídeo de 30 *fps* (*frames per second*) e (2) disponibilizados através de dois fluxos de vídeo independentes com 15 *fps* cada, utilizando-se *MDC* para a quebra e codificação dos mesmos. A Tabela 4.2 mostra detalhes específicos da codificação feita.

Tabela 4.2. Detalhes de codificação das seqüências de vídeo.

Nome do Arquivo	Formato do Arquivo (pixels)	Disponibilizados em apenas um fluxo			Disponibilizados através de dois fluxos		
		N° de fluxos	Taxa (*fps*)	Banda passante média	N° de fluxos	Taxa (*fps*)	Banda passante média
Akiyo	CIF(352x288)	1	30	526 Kbps	2	15	263 Kbps
Foreman	CIF(352x288)	1	30	372 Kbps	2	15	186 Kbps
Hall	CIF(352x288)	1	30	274 Kbps	2	15	137 Kbps

Quando o *BB* admite uma requisição utilizando apenas um caminho, o mesmo seleciona e transmite o vídeo com apenas um fluxo. Isto é feito lendo-se um arquivo de perfil de tráfego pré-armazenado. O mesmo acontece quando uma requisição é admitida utilizando-se dois caminhos, o vídeo é transmitido utilizando-se dois fluxos lendo-se dois arquivos de perfil de tráfego também pré-armazenados.

O *framework EvalVid* [Klaue 2003] foi utilizado para a quebra e codificação dos vídeos. O processo de codificação foi realizado através de vários programas de codificação. Todas as ferramentas utilizadas se encontram disponibilizadas em [Chih-Heng 2006]. O processo de codificação e decodificação utilizado neste trabalho foi:

- primeiro houve a transformação do vídeo sem compressão para o formato H.264, utilizando-se um programa chamado *x264 encoder*;
- em seguida um programa chamado *MP4Box* foi utilizado para a transformação do vídeo em H.264 para o padrão MPEG4, adicionando-se neste processo um conjunto de informações que descrevem como os quadros devem ser transportados através do protocolo *RTP*;
- o Analisador (*Parser*) utilizado foi o programa *mp4Trace*;
- o processo de envio e recepção gera vários arquivos como escritos na seção 2.5. O programa *etmp4* foi utilizado para a "montagem" dos subfluxos de vídeo que seriam encontrados do lado do receptor, em um formato MPEG4;
- e finalmente o programa *ffmpeg* foi utilizado para a descompressão dos subfluxos para que pudessem ser utilizados pelo processo de junção (*Merger*).

Dentre as etapas citadas na seção 2.5, o processo de codificação é o mais importante, pois os vídeos podem ser codificados de diversas formas diferentes que terão um impacto diferente no resultado da simulação. O formato utilizado neste trabalho foi o MPEG4.

O padrão MPEG (*Motion Picture Experts Group*) faz a compressão retirando dois tipos de redundância existentes em fluxos de vídeo: a espacial e a temporal [Tanenbaum 2003]. A redundância espacial é alcançada simplesmente codificando-se cada quadro de um fluxo de vídeo utilizando o padrão *JPEG* (*Joint Photographic Experts Group*).

Compressão adicional pode ser atingida removendo-se a redundância temporal. Esta compressão se aproveita do fato de que dois quadros consecutivos de um fluxo de vídeo geralmente apresentarem muitos pontos (*pixels*) em comum. Logo, uma seqüência de vídeo codificada no formato MPEG consistiria, de forma bem resumida, em um quadro base, chamado de quadro I (*Intracoded*) e uma seqüência de quadros que só teriam a diferença em relação ao quadro anterior, chamados de quadros P (*Predictive)* ou B (*Bidirectional),* dependendo da

codificação suportada. A freqüência em que quadros I e quadros P aparecerão, dependerá da codificação utilizada.

A utilização do programa *x264* para a transformação do vídeo sem compressão para o formado H.264 foi escolhido devido à flexibilidade que este programa oferece na codificação do vídeo. É possível fixar vários parâmetros como a banda passante, a taxa de quadros por segundo e o intervalo entre quadros do tipo I. Obviamente, ao se fixar um determinado parâmetro, outro é flexibilizado, por exemplo, dependendo da banda passante e do intervalo entre quadros I, o número de pacotes será distribuído em função de tempo de forma diferente.

Uma questão importante a se considerar na codificação dos vídeos é o perfil de comportamento do tamanho da rajada de pacotes e seus efeitos nas variações de funcionamento do *MDCAC*. A distribuição e o tamanho das rajadas de pacotes podem influenciar muito o resultado obtido pelas variações V2, V3, V3A e V4.

Para se explicar o porquê, considere que em seu estado inicial, as variações V1 e V2 aceitem três requisições de fluxos de vídeos idênticas e as enviem no mesmo instante de tempo. Considere que para este vídeo, cada quadro I mande 20 pacotes para a rede e que cada vídeo mande um quadro I a cada 30 segundos. A variação V1 enviaria o primeiro fluxo de vídeo pela rota padrão, o segundo fluxo pelo *relay* R1 e o terceiro fluxo de vídeo pelo *relay* R2. Logo, a cada 30 segundos aconteceria uma rajada de 20 pacotes em cada *link*. Agora imagine a mesma situação acontecendo para a variação V2, então o mesmo quebraria o primeiro vídeo em dois subfluxos e os enviaria pela rota padrão e pelo *relay* R1, quebraria o segundo vídeo e o mandaria pelo *relay* R1 e pelo *relay* R2 e finalmente quebraria o terceiro vídeo e o mandaria pela rota padrão e pelo *relay* R2. Como em V2 os quadros do tipo I de cada subfluxo também são enviados utilizando-se 20 pacotes a cada intervalo de 30 segundos, acabaria existindo uma rajada de 40 pacotes a cada 30 segundos em cada *link*, o que poderia ser desastroso em termos de perda de pacotes.

Como na simulação 2 (será abordada em maiores detalhes adiante) são utilizados somente três vídeos diferentes com no máximo 120 segundos, a chance de se admitir dois ou mais fluxos de vídeos iguais no mesmo instante de tempo é alta. Para se amenizar este problema, quando V2, V3, V3A ou V4 enviam um vídeo utilizando-se dois caminhos, estes enviam cada um dos dois subfluxos com um valor de intervalo entre quadros I que seja a metade do valor utilizado pela variação V1.

Por exemplo, se o intervalo entre quadros I dos vídeos enviados por V1 for de 30 segundos, os subfluxos enviados por V2 terão um quadro I a cada 15 segundos. Isso faz com que o número de pacotes enviados por V2, ao enviar um quadro I, seja menor do que o número de pacotes enviados por V1, minimizando desta forma o problema.

Por exemplo, o vídeo *Akiyo* ao ser codificado utilizando-se o programa *x264* com um intervalo entre quadros I de 30 segundos, envia cada quadro I com 20 pacotes. Ao se alterar o intervalo entre quadros I para 15 segundos, passa-se a enviar cada quadro I utilizando-se 12 pacotes. É claro que isso não elimina o problema, mas apenas o ameniza. Também não existe uma regra para se determinar a relação entre o valor do intervalo entre quadros I e número de pacotes que cada quadro I irá utilizar, pois isso depende do vídeo utilizado e de outros parâmetros utilizados na codificação.

4.5. Análise do número de requisições aceitas e do *profit*

Conforme dito anteriormente, foi definida uma simulação para a análise do número de requisições aceitas e do *profi*t, denominada simulação 1. Para esta simulação, o gerador de carga de requisições foi configurado utilizado um arquivo de configuração fornecido junto com o *framework* desenvolvido por Tang (2007). Este arquivo gera uma carga de requisições de tal forma a simular um dos servidores de mídia corporativo da *HP*. Para a simplificação das simulações, selecionou-se um intervalo de 4 horas no qual havia um maior número de pedidos de requisições por segundo.

Observe que para a análise do *profit* e do número de requisições aceitas não é necessário a simulação da transmissão de vídeo na rede, mas sim uma simulação que envolva o recebimento e o processamento de requisições por parte do *BB*. Logo, foi utilizada uma banda passante de 100 Mbps em todos os *links* da topologia definida pela Figura 4.1 e um grande número de requisições.

A Tabela 4.3, a seguir, mostra os resultados da avaliação das variações de funcionamento dos algoritmos em termos do número de requisições aceitas e do *profit*. Foi utilizado um valor de *p* igual a 30% para a variação 3A, conforme definido na seção 3.3.5. As simulações foram repetidas 30 vezes com sementes diferentes para se aumentar a confiabilidade, sendo os resultados da Tabela 4.3, a

média dos resultados das 30 repetições. O intervalo de confiança utilizado foi de 95%.

Tabela 4.3. Resultados dos algoritmos de controle de admissão.

	Número de requisições aceitas (em porcentagem)	*Profit* (x 10^11)
Variação0	17,8520	1,5358
Variação1	48,2004	4,5492
Variação2	47,3339	4,5641
Variação3	47,5378	4,5685
Variação3A	47,5927	4,5678
Variação4	47,9249	4,5513

Como na topologia utilizada existem três caminhos entre A e B, se espera que o ganho de desempenho das variações seja da ordem de três vezes se comparado a uma abordagem que utilize um só caminho. Contudo, como é mostrado na Tabela 4.3, nenhuma abordagem atinge tal meta em relação ao número de requisições aceitas, ou seja, nenhuma delas é três vezes melhor do que a variação zero, que utiliza só a rota padrão. A Tabela 4.4 mostra os ganhos para os valores do número de requisições aceitas das variações V1, V2, V3, V3A e V4 em relação a V0. Este ganho de refere ao valor ideal, que seria três vezes o valor obtido para a variação V0.

Tabela 4.4. Ganhos em relação ao número de requisições aceitas.

	Ganho do número de requisições aceitas em relação a V0
V1	90,00%
V2	88,38%
V3	88,76%
V3A	88,86%
V4	89,48%

Diferente do número de requisições aceitas, o *profit* consegue atingir um valor muito próximo do ideal, como é mostrado na Tabela 4.5, onde a variação V3 tem o maior ganho. V3 consegue este resultado porque primeiro tenta enviar fluxos utilizando só um caminho alternativo e depois verifica se “sobrou” banda passante suficiente para a admissão de um fluxo através de dois caminhos alternativos.

Tabela 4.5. Ganhos em relação ao *profit*.

	Ganho do *profit* em relação a V0
V1	98,73%
V2	99,06%
V3	99,15%
V3A	99,14%
V4	98,78%

Para uma análise mais detalhada das abordagens, observe a Tabela 4.6. Como mostra a tabela, não existe uma variação melhor que deva ser usada sempre, porque os valores observados são similares. Logo, outros fatores devem ser considerados para a comparação das variações, o que serviu de motivação para a análise de perda de pacotes e a análise da rajada de perda de pacotes que serão mostradas nas seções que se seguem.

Tabela 4.6. Analise das abordagens em relação ao *profit* e ao número de requisições aceitas.

	Ganho do número de requisições aceitas em relação a V0	Ganho do *profit* em relação a V0
V1	90,00%	98,73%
V2	88,38%	99,06%
V3	88,76%	99,15%
V3A	88,86%	99,14%
V4	89,48%	98,78%

4.6. Análise da taxa de perda de pacotes

Para a análise da taxa de perda de pacotes, foi definida uma simulação denominada simulação 2. O arquivo de configuração do gerador de requisições foi alterado e simplificado, utilizando-se agora uma simulação com apenas 500 segundos e apenas três seqüências de vídeo disponibilizadas pelo *Video Traces Research Group* [Seeling 2008], com 120s de duração cada. Cada vídeo foi codificado conforme definido na seção 4.4. A simulação 2 é mais complexa do que a simulação 1 por que envolve a transmissão de vídeos na rede e a análise do número de pacotes perdidos em cada nó da rede.

Uma análise detalhada da taxa de perda de pacotes foi feita para as variações 1, 2 e 3 (denominadas V1,V2 e V3), pois se desejava saber se havia alguma diferença na utilização destas abordagens. Para isso a taxa de perda de pacotes foi observada à medida que se aumentava a banda passante disponível nos *links* definidos na Figura 4.1. É de se esperar que o controle de admissão aceite mais fluxos à medida que a banda passante aumente, e como a alocação é feita pela banda passante média dos fluxos, é de se esperar também que haja alguma perda.

Como existe um *overhead* na quebra e codificação dos vídeos, o número médio de pacotes enviado por V2 e V3 é 2,2% e 1,7% maior do que o número de pacotes enviados por V1, respectivamente. O número de pacotes enviados por cada abordagem pode ser visto na Figura 4.3. Todas as simulações subseqüentes foram repetidas 10 vezes com sementes diferentes para se aumentar a confiabilidade, sendo os resultados a média das 10 repetições e o intervalo de confiança utilizado foi de 95%. A Figura 4.4 mostra a taxa de perda de pacotes em função da banda passante disponível para V1, V2 e V3.

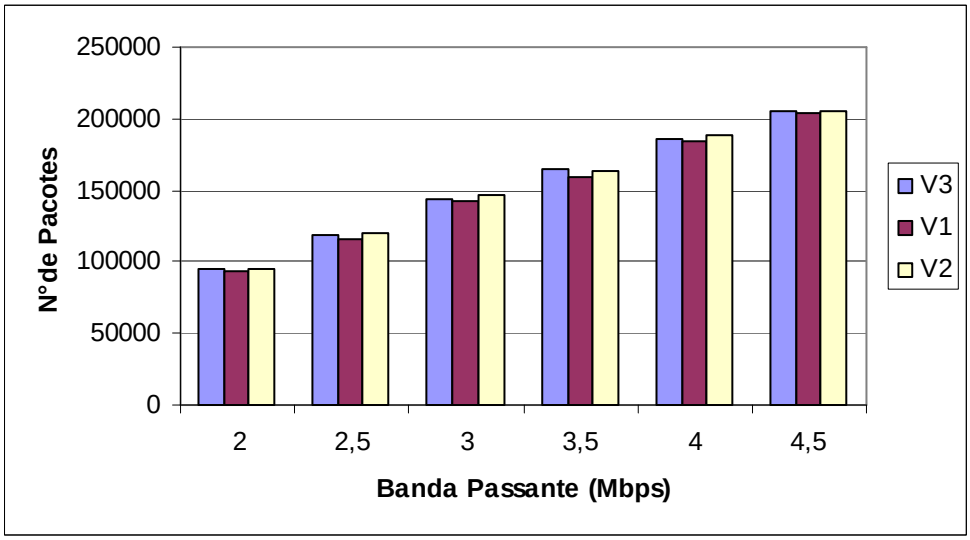

Figura 4.3. Número de pacotes enviados por cada abordagem.

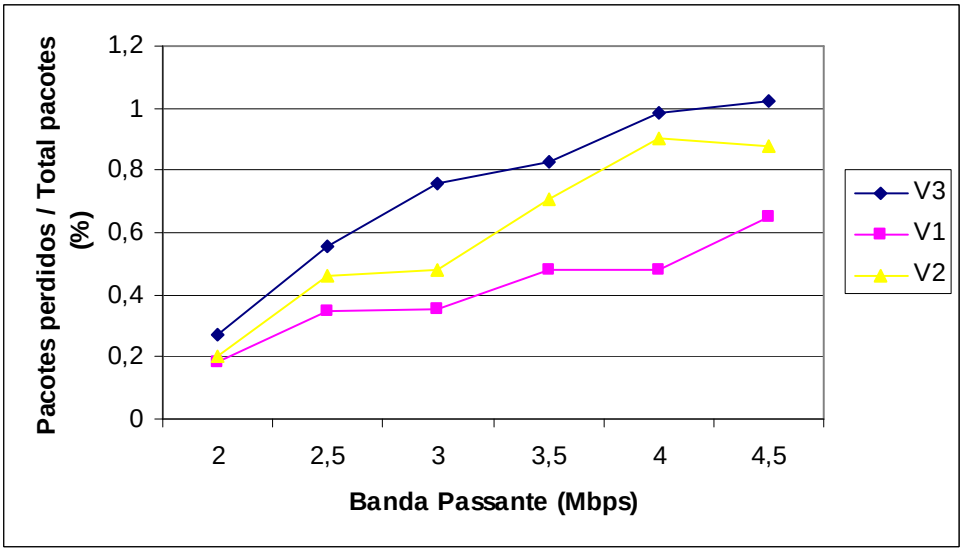

Figura 4.4. Taxa de perda de pacotes em função da banda passante.

Observa-se na Figura 4.4 que à medida que se aumenta a banda passante dos *links* da topologia da Figura 4.1, as variações V2 e V3 apresentam uma taxa de perda maior em relação à variação V1. Deve ser observado que em todas as variações a taxa de perda de pacotes é sempre baixa (abaixo de 1%), o que é um reflexo do controle de admissão de fluxos utilizado. Esta baixa taxa de perda de pacotes só será significativa caso aconteça perda de pacotes em rajadas na rede.

Algumas hipóteses que podem explicar porque estas variações perdem mais pacotes são: (1) devido à configuração da alocação de recursos, que neste caso se refere à banda passante; (2) devido ao tamanho da rajada de pacotes nas diferentes variações; (3) devido a um envio maior de pacotes ao se admitir uma requisição utilizando-se dois fluxos em vez de um só e (4) uma combinação das hipóteses citadas. Logo, as três primeiras hipóteses serão investigadas nas subseções que se seguem.

4.6.1. Investigação da alocação de reserva estatística de recursos

Para se investigar a alocação de recursos, a banda passante disponível na rota padrão para as variações V1 e V2 é exibida na Figura 4.5. Estes valores se referem à banda passante disponível após o processo de alocação de uma simulação específica, considerando-se 2,5Mbps em cada *link*. É importante observar que a alocação feita é uma alocação com garantias estatísticas, pois na verdade o recurso não é alocado, mas sim se verifica se há a disponibilidade de recursos antes da aceitação de um fluxo de vídeo.

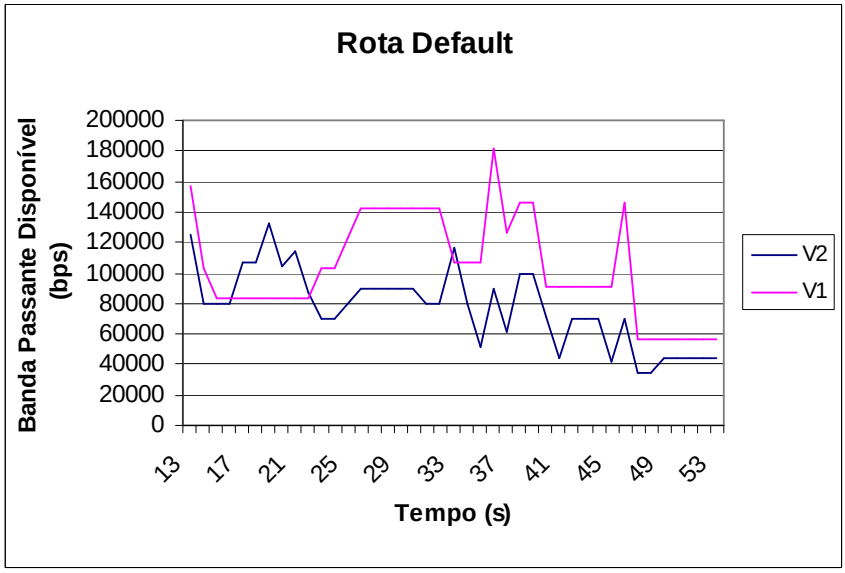

Figura 4.5. Banda passante disponível na rota padrão (Simulação específica).

Como se pode observar, a banda passante disponível após a alocação definida pela variação V2 é menor do que a disponível pela variação V1, na maior parte do tempo. Isso pode indicar que V2 aloca recursos da rede de forma mais eficiente, já que o objetivo é maximizar o uso dos recursos de rede.

Devido ao seu processo de funcionamento, a variação V2 faz a alocação estatística de recursos com valores de banda passante menores do que V1. Por exemplo, para uma requisição de 526 Kbps de banda passante, V1 usará este valor nominal para a alocação do recurso, enquanto V2 irá utilizar dois valores de 263 Kbps para fazer esta alocação. Então pode ser que V2 aloque recursos de forma mais eficiente em média, pois aproveitará recursos que não seriam possíveis de serem utilizados pela variação V1, por lidar com valores maiores. Se isso for verdade, após o processo de alocação, como a banda passante disponível definida por V1 é maior do que V2, esta abordagem (V1) apresentará uma maior capacidade de absorver um desvio de comportamento de um fluxo e definirá um menor nível de perda de pacotes.

Para se investigar o que foi dito anteriormente, foi desenvolvida uma quarta variação de funcionamento do *MDCAC* conforme definido na seção 3.3.6. Como é mostrado na Figura 4.6, V4 consegue atingir um nível de perda de pacotes menor do que a variação V1 para o perfil de tráfego usado neste trabalho, sendo que V4 manda 1,2% mais pacotes do que a variação V1.

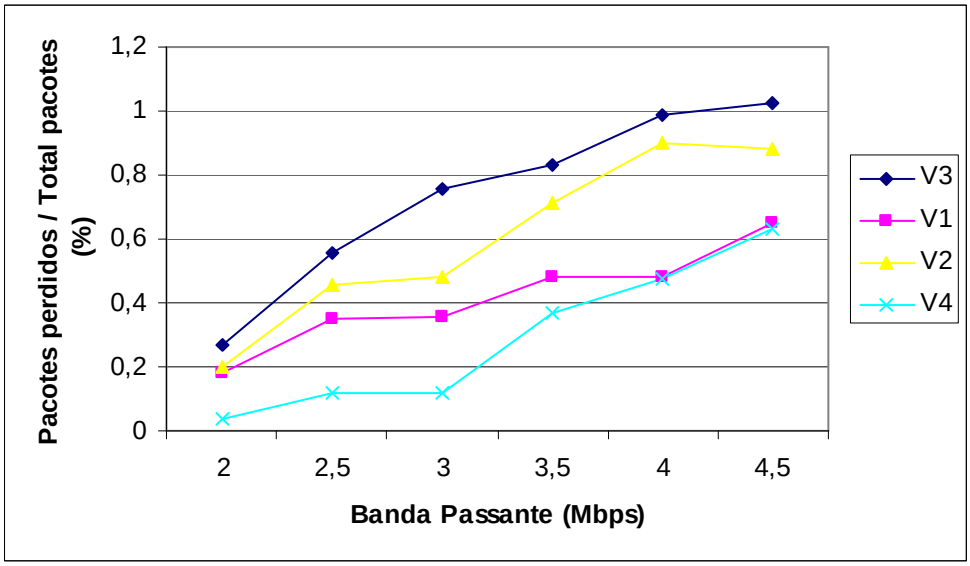

Figura 4.6. Perda de pacotes quando V4 envia 1,2% mais pacotes do que V1.

Com este resultado, o valor médio da banda passante disponível após o processo de alocação, para V1, V2 e V4 foi investigado, pois o interesse foi determinar se V2 realmente aloca a banda passante de forma mais eficiente e qual foi o comportamento da banda alocada por V4. Para isso o *link* definido pela rota padrão foi monitorado e estes resultados são mostrados nas Figuras 4.7, 4.8, 4.9, 4.10, 4.11 e 4.12 para os valores de banda passante de 2,0, 2,5, 3,0, 3,5, 4,0 e 4,5 Mbps, respectivamente.

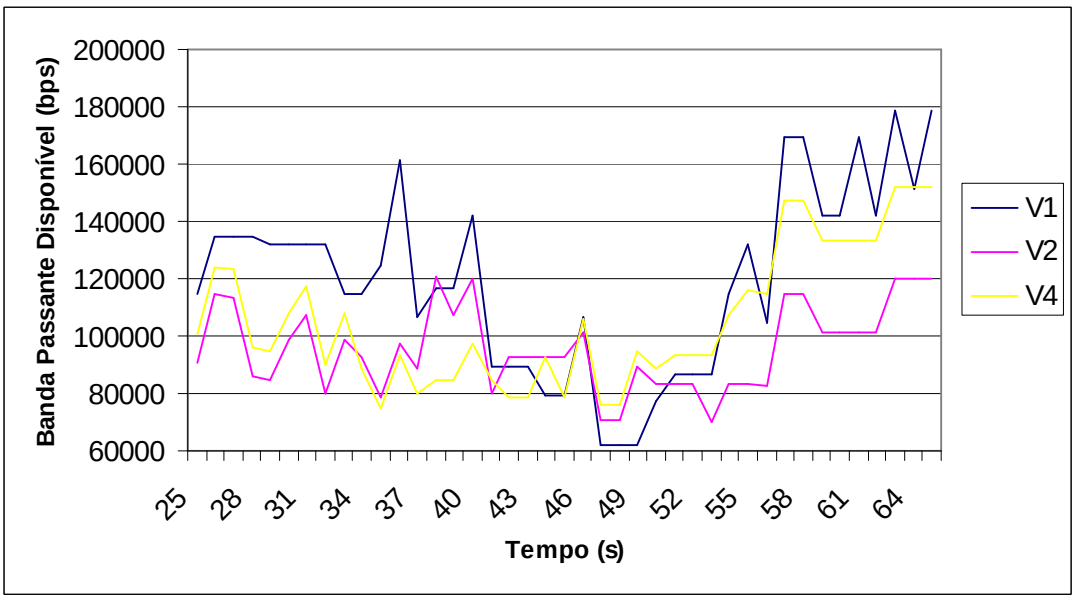

Figura 4.7. Banda passante disponível na rota padrão (*link* de 2,0 Mbps).

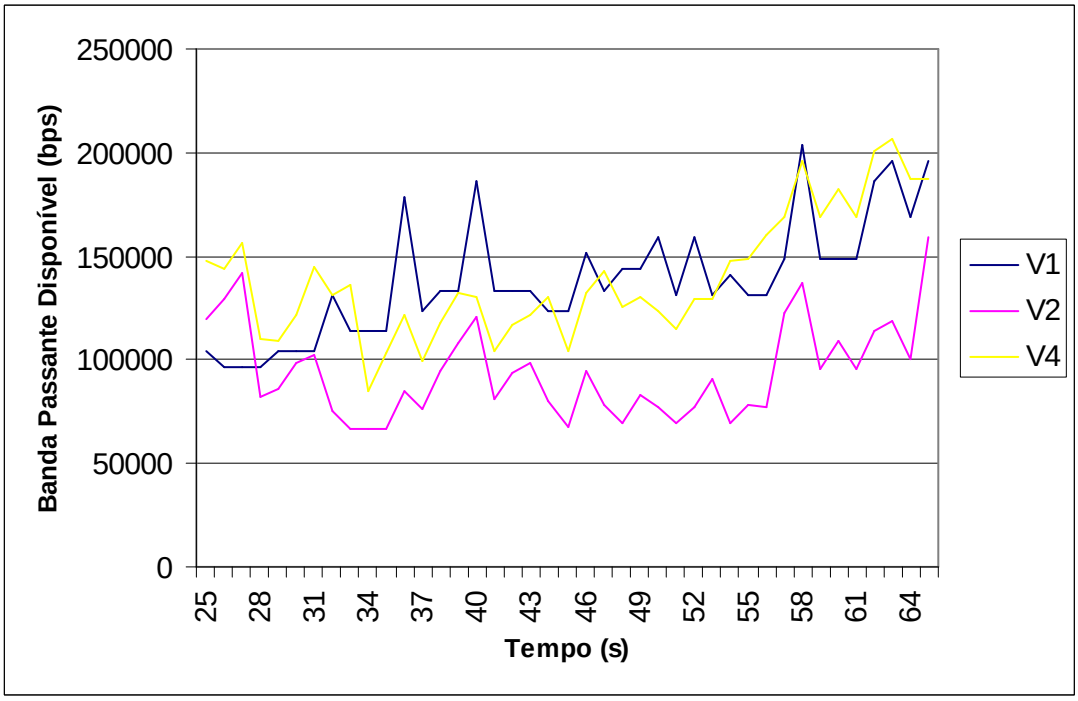

Figura 4.8. Banda passante disponível na rota padrão (*link* de 2,5 Mbps).

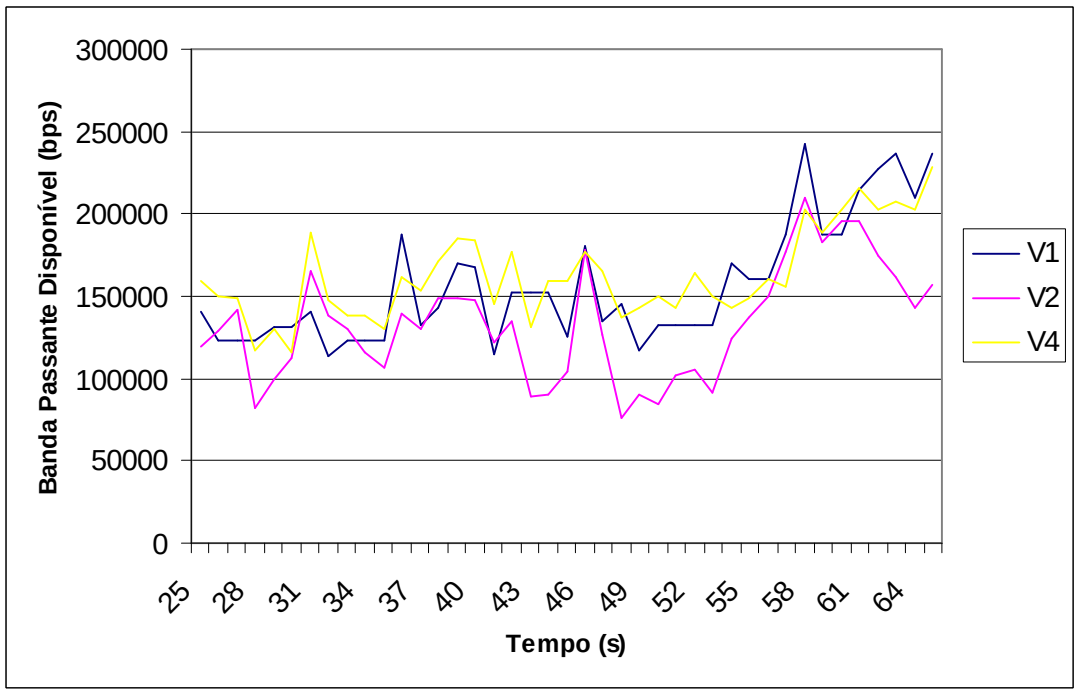

Figura 4.9. Banda passante disponível na rota padrão (*link* de 3,0 Mbps).

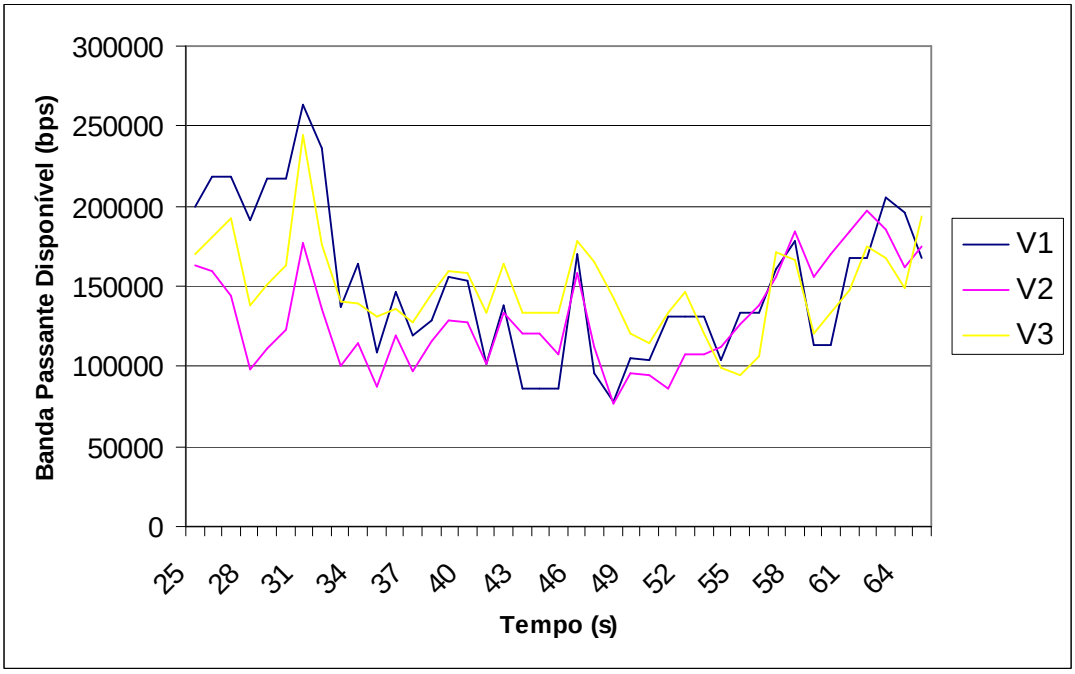

Figura 4.10. Banda passante disponível na rota padrão (*link* de 3,5 Mbps).

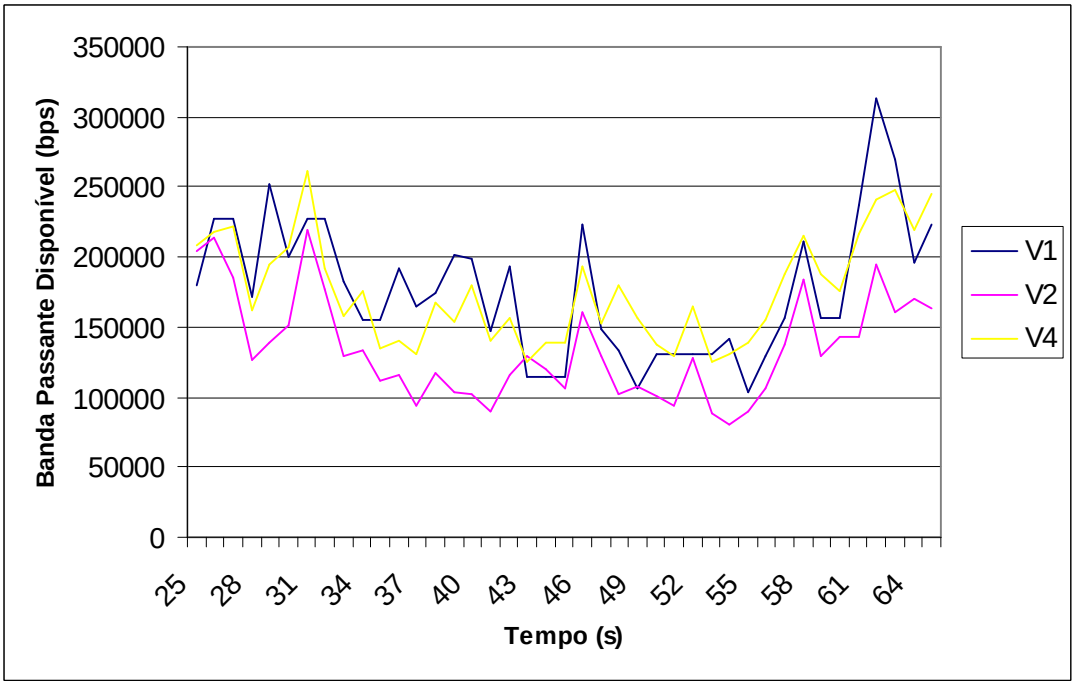

Figura 4.11. Banda passante disponível na rota padrão (*link* de 4,0 Mbps).

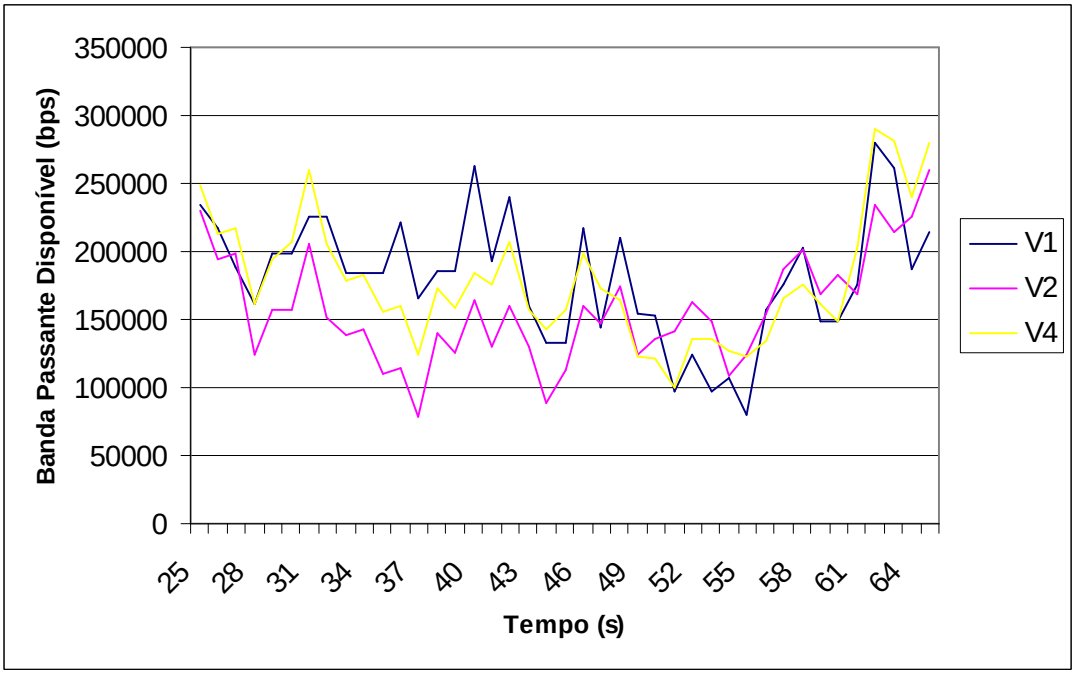

Figura 4.12. Banda passante disponível na rota padrão (*link* de 4,5 Mbps).

Nestes gráficos é possível observar que a variação V2 realmente aloca os recursos da rede de uma forma mais eficiente e que a abordagem V4 apresenta um resultado que é um meio termo entre as variações V1 e V2. O resultado mais importante dessas simulações foi verificar que a configuração da alocação estatística de recursos influencia o resultado como um todo, no que se refere à taxa de perda de pacotes, como é possível se observar no resultado de Figura 4.6.

Uma questão que se levanta é se V4 aceita mais ou menos requisições do que V1 para a simulação 2. As Figuras 4.13 e 4.14 exibem o nível de requisições aceitas e o *profit* para todas as variações analisadas na simulação 2. Os valores são a média de 10 simulações e nelas fica claro que não existe uma diferença significativa entre as variações observadas.

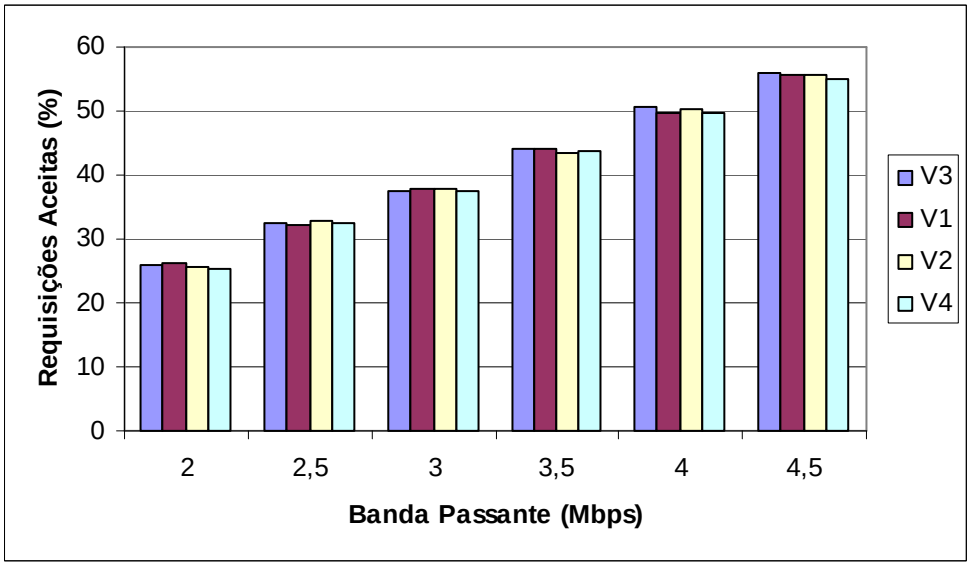

Figura 4.13. Porcentagem de requisições aceitas em função da banda passante.

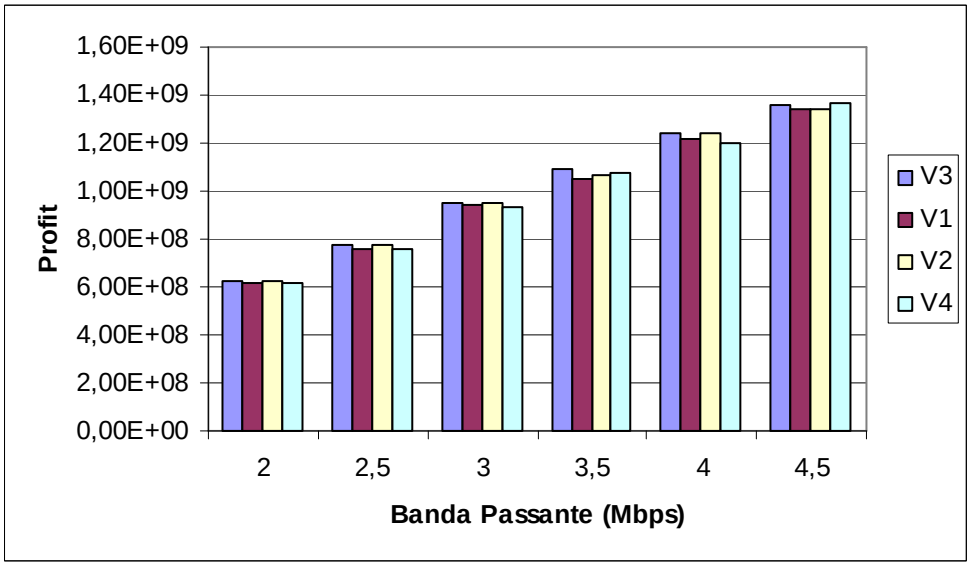

Figura 4.14. *Profit* em função da banda passante.

4.6.2. Investigação do tamanho da rajada de pacotes (e da rajada de perda de pacotes)

Outro ponto que deve ser analisado é o comportamento da rajada de pacotes nas diferentes variações. Por exemplo, pode ser que V4 seja melhor por selecionar vídeos que possuam rajadas de pacotes mais suaves em relação as outras abordagens. Para se investigar esta questão, a rajada de pacotes foi monitorada de uma forma sistêmica, analisando-se as rajadas de pacotes na rota padrão, independente da sua fonte de origem. Em outras palavras, o número de pacotes que

chegavam à fila do roteador C em função do tempo foi medido e apresentado nas Figuras 4.15, 4.16 e 4.17, se referindo a *links* com banda passante de 2,0 e 2,5 Mbps, 3,0 e 3,5 Mbps e 4,0 e 4,5 Mbps, respectivamente.

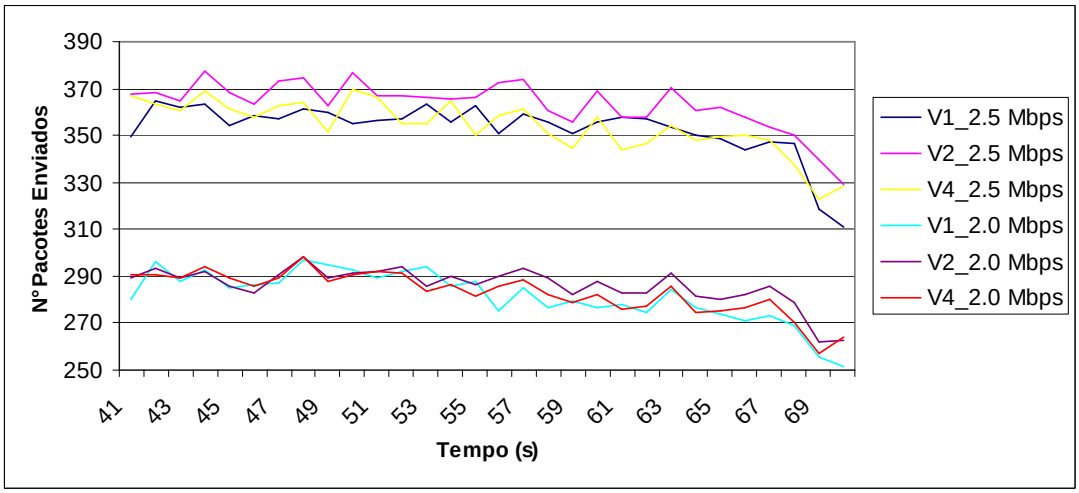

Figura 4.15. Rajada de pacotes na rota padrão (*links* de 2,0 e 2,5 Mbps).

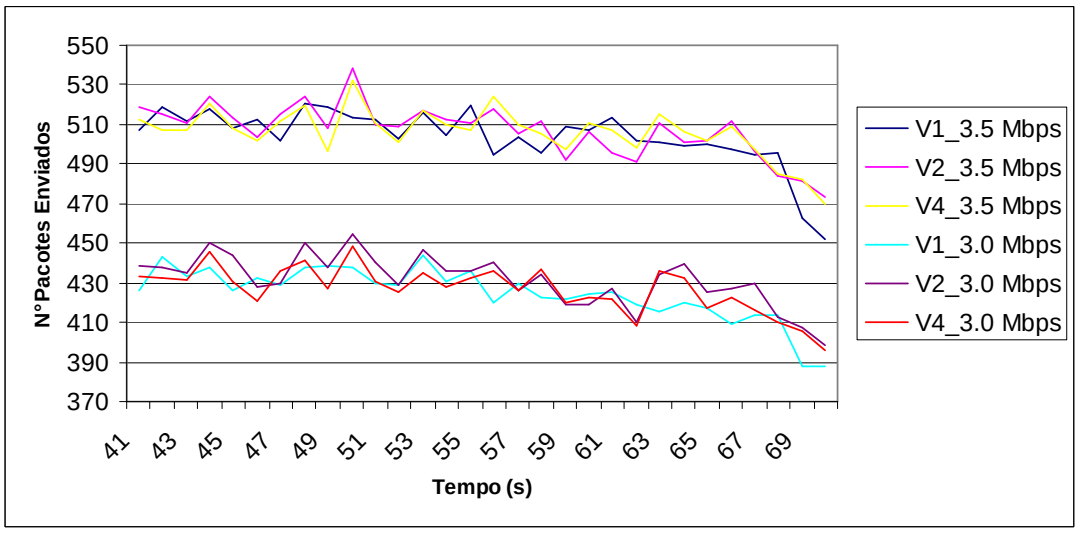

Figura 4.16. Rajada de pacotes na rota padrão (*links* de 3,0 e 3,5 Mbps).

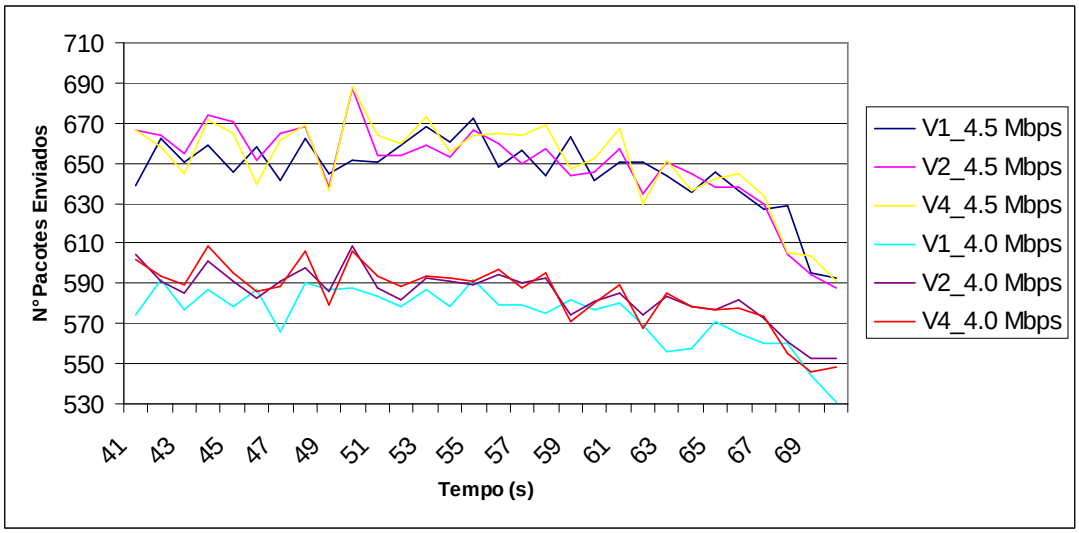

Figura 4.17. Rajada de pacotes na rota padrão (*links* de 4,0 e 4,5 Mbps).

Como mostrado nos gráficos, os valores das rajadas das diferentes variações se entrelaçam e variam em função do tempo e também da banda passante. Desta forma, não é possível, pela a análise dos gráficos, dizer que uma abordagem é melhor do que a outra (apesar de sabermos que V4 perde menos pacotes). O resultado mais importante destas simulações foi verificar que não existe uma discrepância evidente entre as rajadas de pacotes das várias variações observadas.

Observe que até neste ponto foi investigado a alocação estatística de recursos e o comportamento das rajadas de pacotes, não sendo possível explicar porque V2 e V3 perdem mais pacotes do que V1 e porque V4 perde menos pacotes do que V1. Para tornar a análise mais simples, o tamanho da fila de entrada do roteador C foi monitorado, sendo os resultados exibidos nas Figuras 4.18, 4.19 e 4.20, se referindo a *links* com banda passante de 2,0 e 2,5 Mbps, 3,0 e 3,5 Mbps e 4,0 e 4,5 Mbps, respectivamente.

Os valores exibem o tamanho da fila em função do tempo sem se considerar as perdas de pacotes, ou seja, este seria o comportamento médio do tamanho da fila se não houvesse perda de pacotes. Para se entender como estes gráficos foram construídos observe a Tabela 4.7.

Tabela 4.7. Estatísticas de uma fila da topologia da Figura 4.1.

Tempo (s)	Número de pacotes que entram na fila	Número de pacotes que saem da fila	Saldo do número de pacotes que entram e que saem da fila	Pacotes acumulados na fila sem se considerar a perda de pacotes	Perda acumulada de pacotes	Tamanho real da fila
10	278	278	0	0	0	0
11	309	298	11	11	0	11
12	323	293	30	41	0	41
13	319	286	33	74	0	74
...	…	…	…	…	…	…
27	298	278	20	347	0	347
28	311	280	31	378	5	373
29	296	288	8	386	14	372
30	303	288	15	401	30	371
31	302	293	9	410	30	380
32	278	284	-6	404	58	346

A Tabela 4.7 se refere a valores coletados de uma fila da topologia 4.1 e só serve de exemplo para o entendimento dos gráficos apresentados nas Figuras 4.18, 4.19 e 4.20. Os valores apresentados nestes gráficos se referem aos “Pacotes acumulados na fila sem se considerar a perda de pacotes” conforme é mostrado na Tabela 4.7 e é a soma acumulada da quarta coluna. Por exemplo, o terceiro valor da coluna “Pacotes acumulados na fila sem se considerar a perda de pacotes” será a soma dos três primeiros valores da coluna “Saldo do número de pacotes que entram e que saem da fila”. O tamanho real da fila é exibido na última coluna considerando-se a perda de pacotes.

Os gráficos apresentados nas Figuras 4.18, 4.19 e 4.20 foram exibidos para ser possível ter uma idéia de como a fila do roteador C enche em média em função do tempo se a fila fosse infinita, ou seja, se a mesma não perdesse pacotes. Obviamente, existe perda de pacotes e estes gráficos servem para fins de análise.

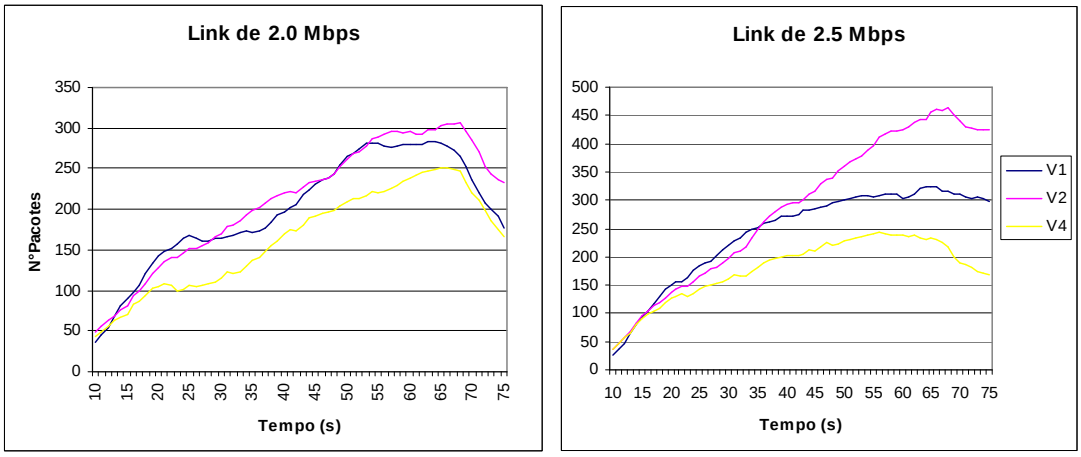

Figura 4.18. Tamanho da fila no roteador C (*links* de 2,0 e 2,5 Mbps).

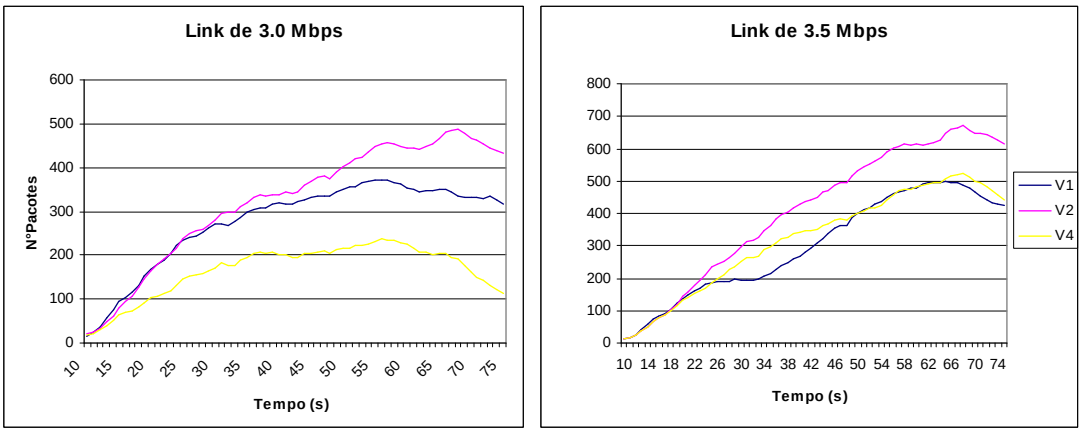

Figura 4.19. Tamanho da fila no roteador C (*links* de 3,0 e 3,5 Mbps).

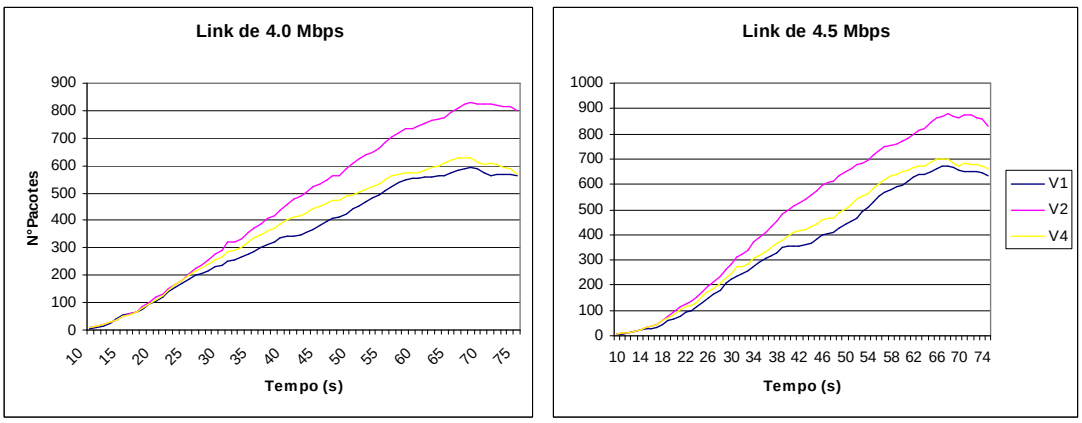

Figura 4.20. Tamanho da fila no roteador C (*links* de 4,0 e 4,5 Mbps).

As Figuras 4.18, 4.19 e 4.20 são as que melhor explicam o resultado obtido na Figura 4.6. Nelas é possível ver que a seqüência de rajadas de pacotes definidas por V4 fazem com que o *buffer* da fila do roteador C fique menos cheio se comparado com as outras variações. Na verdade o *buffer* definido por V4 fica menor considerando-se *links* com banda passante com valores de 2,0 a 3,0 Mbps. Com

valores acima de 3,0 Mbps o tamanho do *buffer* definido por V4 começa se igualar ao valor definido por V1 e eventualmente passa a ter um valor maior do que V1. Isso explicaria porque as abordagens V1 e V4 apresentam um nível de perda de pacotes parecidos a partir do valor de 4,0 Mbps como mostrado na Figura 4.6.

Também foi feita uma análise do tamanho da rajada de perda de pacotes ao se variar o valor da banda passante nos *links* da topologia da Figura 4.1, também utilizando a rota padrão. Estes valores foram coletados na mesma fila analisada nas Figuras 4.18, 4.19 e 4.20, ou seja, a fila do roteador C da topologia mostrada na Figura 4.1. As Figuras 4.21, 4.22 e 4.23 mostram como a rajada de perda de pacotes se comportou com o passar do tempo, considerando-se *links* com banda passante de 2,0 e 2,5 Mbps, 3,0 e 3,5 Mbps e 4,0 e 4,5 Mbps, respectivamente.

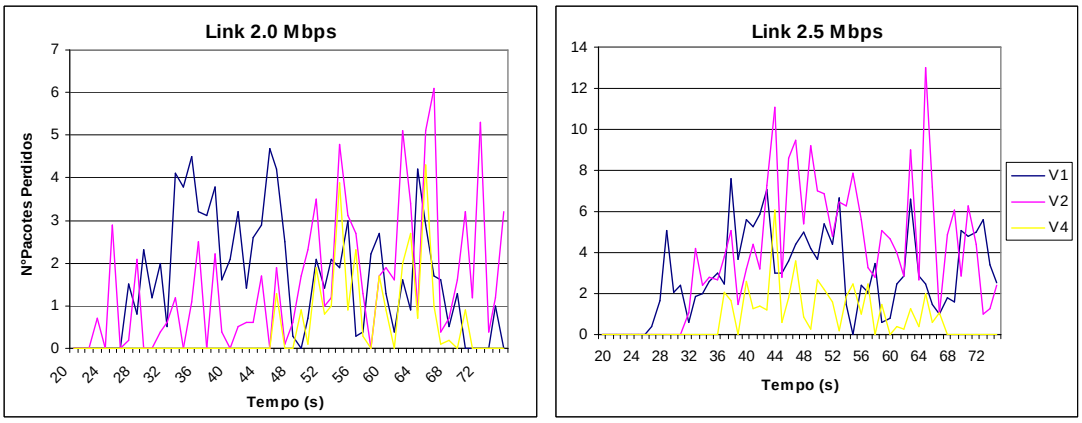

Figura 4.21. Rajada de perda de pacotes na rota padrão (*links* de 2,0 e 2,5 Mbps).

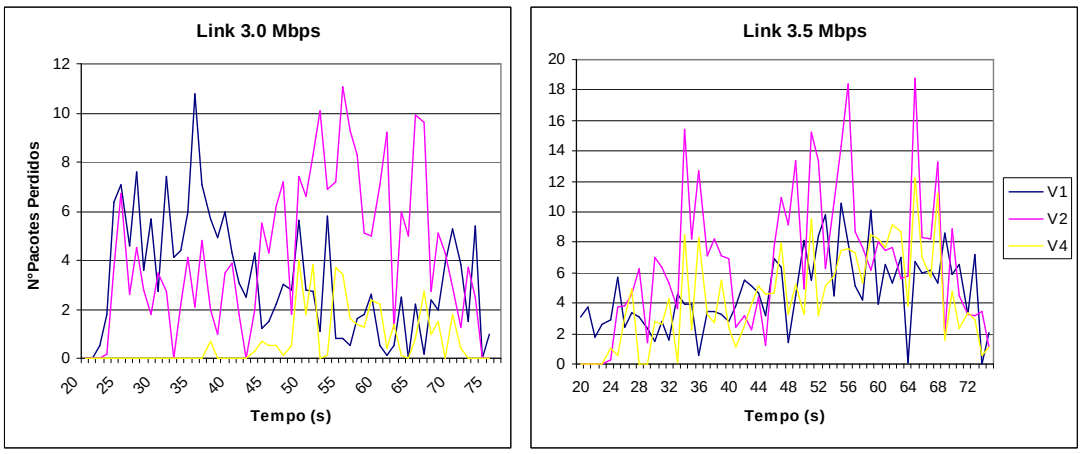

Figura 4.22. Rajada de perda de pacotes na rota padrão (*links* de 3,0 e 3,5 Mbps).

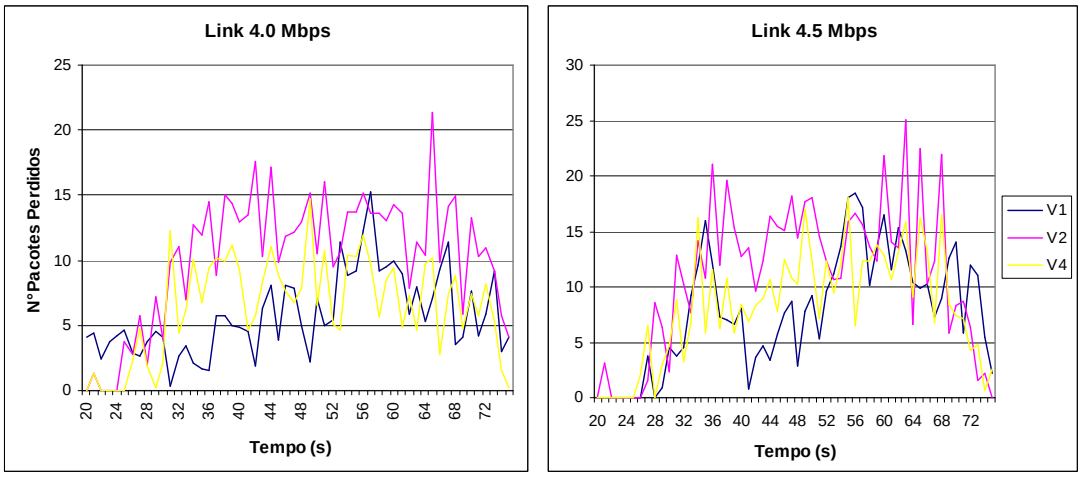

Figura 4.23. Rajada de perda de pacotes na rota padrão (*links* de 4,0 e 4,5 Mbps).

O tamanho e a distribuição da rajada de perda de pacotes é um reflexo de como estava a fila do roteador C, logo seguirá a mesma tendência apresentada na análise feita para esta fila. Como se pode observar, a rajada de perda apresentada por V4 é menor do que a rajada de perda das outras variações, considerando-se *links* de 2,0, 2,5 e 3,0 Mbps. A partir de um valor de 3,0 Mbps, a rajada de perda de pacotes apresentada por V4 começa a se aproximar da rajada de perda apresentada por V1. Contudo, o número de pacotes perdido por V4 é menor do que o número de pacotes perdidos por V1 e V2 como já foi mostrado na Figura 4.6.

4.6.3. Investigação do *overhead* gerado ao utilizar diversidade de caminhos

Para se determinar até que ponto a variação V4 é melhor do que a variação V1, o número de pacotes enviados por V4 foi aumentado de tal forma a ficar 3% maior do que o enviado por V1. Isso foi feito alterando-se os parâmetros de codificação dos vídeos, como o intervalo entre quadros I e a taxa de quadros por segundo. A Figura 4.24 mostra a taxa de perda de pacotes para este caso.

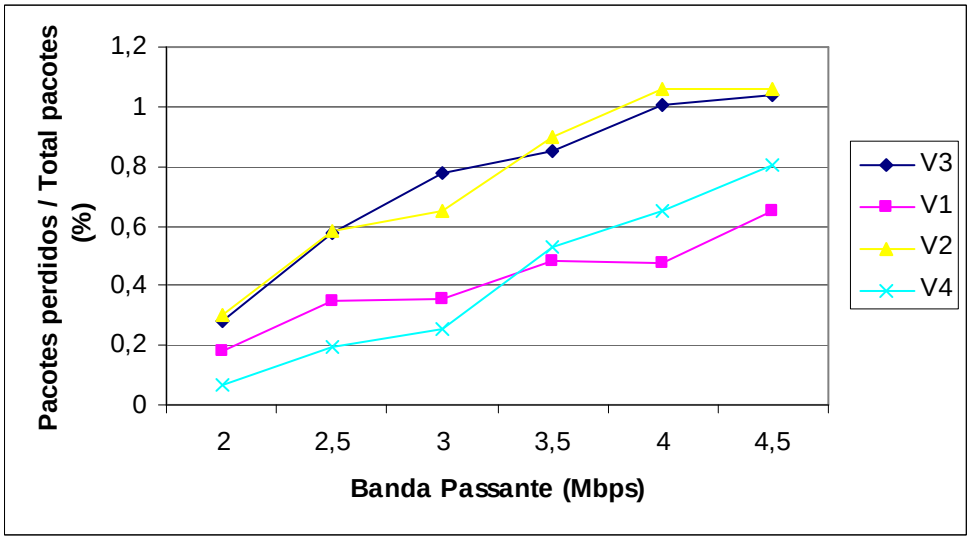

Figura 4.24. Perda de pacotes quando V4 envia 3% mais pacotes do que V1.

Observa-se que ao utilizar uma banda passante maior do que 3,25 Mbps, V4 passa a perder mais pacotes do que V1. Ao se utilizar bandas passantes maiores, o número de fluxos aceitos também é maior e conseqüentemente o número de pacotes aumentará. Com este resultado, tornou-se relevante saber como este gráfico ficaria se V4 enviasse 6% mais pacotes do que V1. Este resultado foi gerado e é exibido na Figura 4.25.

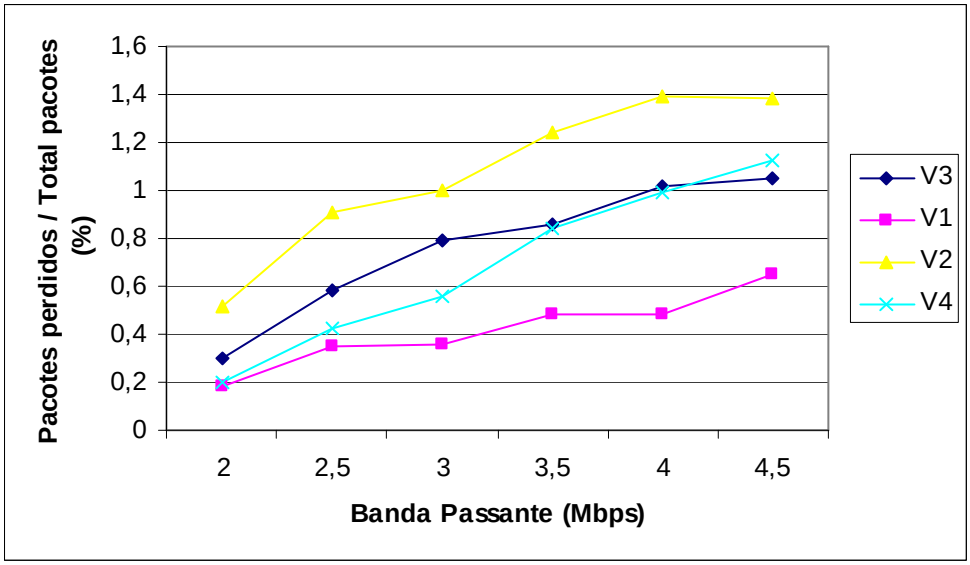

Figura 4.25. Perda de pacotes quando V4 envia 6% mais pacotes do que V1.

Obviamente, à medida que V4 manda mais pacotes, é de se esperar que a partir de um determinado momento este iria deixar de ser melhor do que V1. Como

é possível ver na Figura 4.25, V4 é pior do que V1 para um nível de 6% a mais de pacotes enviados, independente do valor da banda passante em questão.

Capítulo 5

Conclusão e trabalhos futuros

Este trabalho abordou temas relacionados com métodos que tentam solucionar o problema de congestionamento em redes IP. Várias metodologias foram apresentadas, como o *DiffServ*, o *IntServ* e algoritmos de controle de admissão. O objetivo principal foi o desenvolvimento e o estudo de disciplinas de controle de admissão utilizando o método de diversidade de caminhos juntamente com a técnica *MDC* (*Multiple Description Coding*).

Como resultado principal se observou que a utilização de caminhos alternativos para o envio de fluxos multimídia apresenta uma melhora em relação à utilização do método tradicional, no qual se utiliza o caminho definido pelo algoritmo de roteamento. Porém, esta melhora não é linear, ou seja, o uso de três caminhos entre dois pontos A e B quaisquer da rede não significa que se admitirá três vezes mais fluxos. Isso irá depender de uma série de fatores como a seqüência de requisições e o perfil de tráfego. Com relação ao número de requisições aceitas, existe uma pequena vantagem na utilização da variação V1 em relação às variações V2, V3, V3A e V4. Para a carga de trabalho utilizada neste trabalho, a variação V1 obteve o maior ganho, sendo este de 90% em relação ao valor ideal, que seria três vezes o valor obtido para a variação V0.

Já os valores obtidos para o *profit* conseguem atingir um valor muito próximo do ideal, onde V2, V3 e V3A obtiveram um ganho de 99% em relação ao valor ideal. Também foi comprovado que V2 consegue alocar a banda passante da rede de forma mais eficiente e esta abordagem seria interessante se os fluxos da rede seguissem seus perfis de tráfego acordados previamente.

A variação V4 apresenta uma vantagem no que se refere à taxa de pacotes perdidos na rede. Observa-se que mesmo havendo um *overhead* devido à quebra do vídeo em dois descritores, a taxa definida pelo número de pacotes perdidos dividido pelo número de pacotes enviados fica menor ou igual a uma abordagem que sempre manda o vídeo através de apenas um fluxo em um único caminho alternativo (V1). A abordagem V4 também apresenta uma vantagem em relação à diminuição da

rajada de perda, uma vez que uma requisição é sempre enviada através de dois fluxos independentes.

Como já mencionado, a utilização de caminhos alternativos na rede é vantajosa em relação ao método tradicional. Basta agora determinar qual variação de funcionamento seria a melhor e em que cenário. Obviamente, há um interesse de se utilizar uma abordagem que utilize dois caminhos alternativos ao se enviar um fluxo, por todas as vantagens obtidas ao se utilizar o método de diversidade de caminhos. Conforme mostrado na Tabela 4.6, considerando só o *profit* e o número de requisições aceitas, a variação V1 pode ser a solução mais adequada, pela sua simplicidade de implementação e por atingir bons resultados tanto no número de requisições aceitas quanto no *profit*.

Contudo, devemos observar outros fatores como a taxa de perda de pacotes e o tamanho da rajada de perda de pacotes. Como observado na Figura 4.6, V2 e V3 perdem mais pacotes do que a variação V1. Já a variação V4 perde menos pacotes do que V1, para um *overhead* de 1,2%, ou seja, quando V4 manda 1,2% mais pacotes do que V1. Também se observa nas Figuras 4.21, 4.22 e 4.23 que V4 apresenta um nível de rajada de perda de pacotes menor do que V1, também para um *overhead* de 1,2%.

Logo, como V4 apresenta um bom ganho no que se refere ao *profit* e ao número de requisições aceitas, como mostra a Tabela 4.6 (pois todas as versões apresentam resultados similares em relação a estes parâmetros), se considerarmos a perda de pacotes e a rajada de perda de pacotes, a variação V4 passa a ser a abordagem mais vantajosa. Contudo, esta vantagem em relação à taxa de perda de pacotes depende fortemente da codificação utilizada nas descrições dos vídeos.

Como trabalho futuro pretende-se desenvolver um algoritmo de controle de admissão que admita fluxos que não sejam de mídia contínua como de emails e transferência de arquivos utilizando-se só um caminho e que admita fluxos multimídia através de dois caminhos independentes e identicamente distribuídos. A idéia seria comparar esta abordagem com a tradicional, onde os fluxos são admitidos de forma igual.

Também há um interesse de se definir políticas dinâmicas no domínio *DiffServ* que interajam com o algoritmo de admissão. Neste caso o *BB* aplicaria

políticas no domínio *DiffServ* de forma dinâmica, de acordo com um mecanismo de realimentação que receberia informações da rede.

Referências Bibliográficas

[Apostolopoulos 2001] Apostolopoulos, J. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. Visual Communications and Image Processing (VCIP), pages 392–405, Jan. 2001.

[Black 2001] Black, D., Floyd, S., and Ramakrishnan, K. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, IETF, 2001.

[Blake 1998] Blake, S. et al. An Architecture for Differentiated Service. RFC 2475, IETF, 1998.

[Bolot 1993] Bolot, J.C. End-to-end packet delay and loss behavior in the Internet. Computer Communication Review, ACM SIGCOMM ’93, 23(4):289–298, Sept. 1993.

[Bolot 1999] Bolot, J.C., Fosse-Parisis, S., and Towsley, D. Adaptive FEC-based error control for Internet telephony. In Proceedings of IEEE INFOCOM ’99, volume 3, pages 1453–1460, Mar. 1999.

[Bouras 2007] Bouras, C., Primpas, D., and Stamos, K. Enhancing ns-2 with DiffServ QoS features. 10th International Communications and Networking Simulation Symposium (CNS 07), Norfolk Marriott Waterside, Norfolk, VA, USA, Mar. 2007.

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

[Braden 1997] Braden, R. et al. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205, Sep. 1997.

[Chih-Heng 2006] Chih-Heng, K. Multiple Description Coding (MDC). Available at: http://140.116.72.80/~smallko. Acesso em: 01 jan. 2009.

[Fitzek 2004] Fitzek, F.H. et al. Overhead and Quality Measurements for Multiple Description Coding for Video Services. Wireless Personal Multimedia Communications (WPMC), Sep. 2004.

[Golubchik 2002] Golubchik, L. et al. Multi-path continuous media streaming: What are the benefits? In Performance Evaluation, volume 49, pages 429–449, Kluver, Sep. 2002.

[Guan 2001] Guan, P. Admission Control Algorithms: A Survey. White paper, 2001.

[Guerin 1997] Guerin, R., Partridge, C., and Shenker, S. Specification of Guaranteed Quality of Service. RFC 2212, IETF, 1997.

[Hardman 1995] Hardman, V., Sasse, A., and Watson, A. Reliable audio for use over the Internet. In Proceedings of INET '95, pages 171–178, Jun. 1995.

[Heinanen 1999] Heinanen, J. et al. Assured Forwarding PHB Group. RFC 2597, IETF, 1999.

[Jacobson 1999] Jacobson, V., Nichols, K., and Poduri, K. An Expedited Forwarding PHB. RFC 2598, IETF, 1999.

[Klaue 2003] Klaue, J., Rathke, B., and Wolisz, A. EvalVid - A Framework for Video Transmission and Quality Evaluation. Proceedings of the 13th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Urbana, Illinois, USA, Sep. 2003.

[Liang 2003] Liang, J., Apostolopoulos, J., and Girod, B. Analysis of packet loss for compressed video: Does burst-length matter? Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-2003, Apr. 2003.

[Maxemchuk 1975] Maxemchuk, N.F. Dipersity routing in store and forward networks. PhD thesis, University of Pennsylvania, 1975.

[McCanne 2008] McCanne, S. and Floyd, S. Ns Network Simulator. Available at: http://www.isi.edu/nsnam/ns.

[Nagle 1987] Nagle, J. On Packet Switches with Infinite Storage. IEEE Trans. on Commun. Vol. COM-35, pages 435-438, Apr. 1987.

[Nichols 1998] Nichols, K.D. et al. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474, Technical report, IETF - Network Working Group, 1998.

[Nichols 1999] Nichols, K., Jackobson, V., and Zhang, L. A Two-bit Differentiated Services Architecture for the Internet. RFC 2638, Jul. 1999.

[Pereira 2002] Pereira, M.U., Antonini, M., and Barlaud, M. Channel adapted multiple description coding scheme using wavelet transform. International Conference on Image Processing, vol. 2, pages 197-200, 2002.

[Pieda 2000] Pieda, P. et al. A Network Simulator, Differentiated Services Implementation. Open IP, Nortel Networks, 2000.

[Postel 1981] Postel, J. Internet Protocol - DARPA Internet Program Protocol Specification. RFC 791, USC/Information Sciences Institute, Sep. 1981.

[Rahin 1998] Rahin, M.A. and Kara, M. Call admission control algorithms in ATM networks: A performance comparison and research directions. Research report, 1998.

[Ribeiro 2004] Ribeiro, B.F., Silva, E.S., and Towsley, D. Estudo da Eficácia do Método de Diversidade de Caminhos para Aplicações Multimídia. Anais do XXII Simpósio Brasileiro de Redes de Computadores (SBRC'04), pp. 307-320, Gramado, RS, Maio 2004.

[Sanneck 2000] Sanneck, H. and Le, N.T.L. Speech property-based FEC for Internet telephony applications. In Proceedings of SPIE - The International Society for Optical Engineering, volume 3969, pages 38–51, Jan. 2000.

[Savage 1999] Savage, S. et al. The end-to-end e effects of internet path selection. Proceedings of the ACM SIGCOMM , Oct. 1999.

[Seeling 2008] Seeling, P., Fitzek, H.P., and Reisslein, M. Video traces for network performance evaluation: YUV 4:2:0 Video Sequences. Available at: http://trace.eas.asu.edu/yuv/yuv.html.

[Shenker 1997] Shenker, S. and Wroclawski, J. General characterization parameters for integrated service network elements. RFC 2215, Technical report, IETF - Network Working Group, 1997.

[Somasundaram 2004] Somasundaram, S. and Subbalakshmi, K.P. MDC and path diversity in video streaming. Image Processing, 2004 (ICIP '04), Oct. 2004.

[Tanenbaum 2003] Tanenbaum, A.S. Computer Networks. Pearson Education, Inc, 2003.

[Tang 2007] Tang, W.A. et al. Modeling and generating realistic streaming media server workloads. Computer Networks: The International Journal of Computer and Telecommunications Networking, Pages: 336 - 356, Jan. 2007.

[Teixeira 2003] Teixeira, R. et al. Characterizing and measuring path diversity in Internet topologies. Proceedings of ACM SIGMETRICS, San Diego, CA, USA, Jun. 2003.

[Venkata 2002] Venkata, N.P., Wang, H.J., and Chou, P.A. Distributing streaming media content using cooperative networking. Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, Miami, Florida, USA, Pages: 177 – 186, May 2002.

[Wroclawski 1997a] Wroclawski, J. Specification of the Controlled-Load Network Element Service. RFC 2211, IETF, 1997.

[Wroclawski 1997b] Wroclawski, J. The Use of RSVP with IETF Integrated Services. RFC 2210, IETF, 1997.

[Yang, 1995] Yang, C.Q. and Reddy, A.V.S. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. IEEE Network Magazine, vol. 9, pp. 34-45, Jul./Aug. 1995.