MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
SECRETARIA DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

JOÃO PAULO DE BRITO GONÇALVES

DIFERENCIAÇÃO DE SERVIÇOS EM AMBIENTES VIRTUAIS COLABORATIVOS EM GRANDE ESCALA

Rio de Janeiro
2006

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

C2006

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro - RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita à referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

G635 GONÇALVES, João Paulo de Brito
Diferenciação de serviços em ambientes virtuais colaborativos em grande escala / João Paulo de Brito Gonçalves. - Rio de Janeiro: Instituto Militar de Engenharia, 2006.

85p. : il., fig.

Dissertação (mestrado) – Instituto Militar de Engenharia, 2006.

1. Ambientes virtuais colaborativos 2. Diferenciação de Serviços 3. Qualidade de Serviço I. Instituto Militar de Engenharia

CDD 004.5

INSTITUTO MILITAR DE ENGENHARIA

JOÃO PAULO DE BRITO GONÇALVES

DIFERENCIAÇÃO DE SERVIÇOS EM AMBIENTES VIRTUAIS COLABORATIVOS EM LARGA ESCALA

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Paulo César Salgado Vidal, D. Sc.

Co-orientador: Jauvane C. de Oliveira, Ph.D.

Aprovada em 04 de agosto de 2006 pela seguinte Banca Examinadora:

_____

Prof. Paulo César Salgado Vidal, D. Sc. do IME - Presidente

_____

Prof. Artur Ziviani, Ph.D. do LNCC

_____

Prof. Edison Ishikawa, D.Sc. do IME

Rio de Janeiro

2006

Dedico esse trabalho ao meu orientador,
TC Vidal, cuja boa vontade e amizade
foram imprescindíveis para seu término

AGRADECIMENTOS

Primeiramente a Deus, por ter me dado saúde, paciência e perseverança para concluir meu trabalho.

À minha namorada Aline, cujo sorriso foi uma luz-guia em meus sombrios dias de trabalho na dissertação.

À minha mãe, Sonia, pelo amor, compreensão e apoio.

Aos meus queridos irmãos: Karla e Carlinhos.

Ao professor Paulo Rosa pela compreensão e constante vontade em ajudar.

À amiga Michelle, pela preciosa ajuda.

Aos companheiros de Mestrado, Evandro, Fábio, Márcio, Melina e outros, que sempre se preocuparam com os rumos que meu trabalho tomava.

Ao Instituto Militar de Engenharia pela oportunidade.

À CAPES pela ajuda indispensável para a realização deste trabalho.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AF	Assured Forwarding
AVC	Ambiente Virtual Colaborativo
ATM	Asynchronous Transfer Mode
BA	Behavior Aggregate
CORBA	Common Object Request Broker Architecture
CBQ	Class Based Queuing
DIFFSERV	Differentiated Services
DIS	Distributed Interactive Simulation
DIVE	Distributed Interactive Virtual Environment
DSCP	DiffServ CodePoint
DSMCast	DiffServ Multicast
EF	Expedited Forwarding
FTP	File Transfer Protocol
HLA/RTI	HighLevel Architecture/Runtime Infrastructure
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
IETF	Internet Enginnering Task Force
IntServ	Integrated Services
ISTP	Interactive Sharing Transfer Protocol
LAN	Local Área Network
LSRs	Label-Switching Routers
LSPs	Label Switched Paths
LDP	Label Distribution Protocol
MASSIVE	Model, Architecture and System for Spatial Interaction in Virtual Environments
MERL	Mitsubishi Electric Research Labs
MF	Multi-Field
MPLS	Multiprotocol Label Switching Architecture

NS	Network Simulator
OSI	Open Systems Interconnection
PDUs	Protocol Data Units
PFIFO	Priority First In First Out
PHB	Per-hop Behavior
CBQ	Class Based Queuing
QoS	Quality of Service
RED	Random Early Detection
RSVP	Resource reSerVation Protocol
RTP	Real Time Protocol
SimNet	Simulator Networking
SPLINE	Scalable Platform for Large Interactive Networked Environments
SRM	Scalable Reliable Multicast
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TOS	Type of Service
VELVET	Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs
VESIR-6	Virtual Environment Supporting Multi-user Interaction over IPv6.
VINT	Virtual InterNetwork Testbed
VPN	Virtual Private Network
VRTP	Virtual Reality Tranfer Protocol
WAN	Wide Area Network
WWW	World Wide Web

RESUMO

Os ambientes virtuais colaborativos (AVCs) são aqueles que possibilitam compartilhamento do mundo virtual por usuários que não precisam estar fisicamente no mesmo local. Para a comunicação entre usuários, utiliza-se uma rede de comunicação.

Ambientes Virtuais Colaborativos baseados na Internet apresentam potenciais problemas de desempenho, já que a Internet não provê garantias quanto à largura de banda e atraso. Como aplicações multimídia, eles necessitam de certas garantias mínimas para executarem com um desempenho adequado.

O presente trabalho apresenta as etapas de desenvolvimento de um modelo de simulação para um protocolo de comunicação para AVCs. Para aumentar a confiabilidade deste protocolo, ele foi combinado com técnicas de qualidade de serviço e posteriormente modelado em um simulador, para que seu desempenho fosse validado através de simulações.

Estudos sobre plataformas para AVCs existentes na literatura relacionada são apresentados bem como os resultados obtidos com as simulações executadas. O principal objetivo foi avaliar o desempenho da arquitetura na transmissão de mensagens em relação a atrasos e perda de pacotes.

ABSTRACT

Collaborative Virtual Environments (CVEs) make possible the sharing of virtual world for users who do not need to be physically in the same place. For the communication between users, a communication network is used.

Internet-based Collaborative Virtual Environments present potential performance problems, as the Internet does not provide any guarantees of bandwidth or delay. As multimedia applications, they must have certain minimum guarantees to execute with an adequate performance.

This work presents the stages of development of a model of simulation for a protocol of communication for AVCs. To increase the reliability of this protocol, it was combined with techniques of quality of service and later shaped in a simulator, for performance validation through simulation.

Related work on existing platforms for AVCs are presented along with the results obtained with the executed simulations. *The main goal is to evaluate the performance of the arquitecture for message transmission* in relation to delay and packet lost.

1. INTRODUÇÃO

Este capítulo apresenta o contexto e a motivação para a escolha do tema apresentado, objetivo e a organização desta dissertação.

1.1 CONTEXTO E MOTIVAÇÃO

Os ambientes virtuais colaborativos (AVCs) são aqueles que possibilitam compartilhamento do mundo virtual por usuários geograficamente dispersos (ERASLAN et al., 2003). Enquanto a dispersão geográfica entre usuários possa envolver uma distância consideravelmente longa, os ambientes virtuais executando em diferentes lugares devem estar operando e interagindo como se estivessem compartilhando o mesmo espaço físico.

Os AVCs que possuem um grande número de usuários ou uma extensão elevada são conhecidos como Ambientes Virtuais Colaborativos em Grande Escala. Muitos destes ambientes utilizam a Internet como meio de comunicação e troca de mensagens devido ao grande número de usuários que a utilizam em todo mundo. Atingir a interação multi-usuário nestes ambientes na Internet, onde a largura de banda disponível é geralmente escassa é considerado um grande desafio.

Os AVCs são aplicações multimídias críticas que necessitam de garantias específicas em relação à largura de banda e atraso para que possam manter a consistência e interatividade e, por conta disso, precisam utilizar mecanismos de qualidade de serviço (*Quality of Service* - QoS). A Qualidade de Serviço é uma área de pesquisa que estuda técnicas para garantir às aplicações os recursos que elas necessitam para executarem adequadamente, mesmo quando a rede está em condições não favoráveis.

Uma técnica adequada de qualidade de serviço para AVCs, dada sua grande diversidade de fluxos de dados, é a diferenciação de serviços (*DiffServ – Differentiated Services)*

(BLAKE et al., 1998). Nesta arquitetura, o protocolo da camada de rede deve ser capaz de associar pacotes com classes de serviços particulares e provê-las com os serviços associados a cada classe. Para tal, não é necessário o uso de mensagens adicionais para prover sinalização de qualidade de serviço. Ao invés disso, são usados os campos Classe de Tráfego no cabeçalho IPv6 ou o campo Tipo de Serviço (*Type of Service-TOS*) do cabeçalho IPv4 como mecanismos de classificação de qualidade de serviço. Dessa forma, se consegue garantir recursos sem aumentar o número de mensagens em trânsito e a latência da rede.

Em AVCs é desejável tratar pacotes de tipos de mensagens diferentes de formas diferentes na rede. Por exemplo, atualizações de estado relacionadas a novos usuários se unindo ao mundo virtual são mais importantes e urgentes no encaminhamento que atualizações de eventos e podem ser relacionadas a classes de tráfego com tratamento mais prioritário nos roteadores. Níveis de serviço diferentes devem ser providos para fluxos de tráfego diferentes e para alguns objetos se necessário.

1.2 OBJETIVOS

Podemos observar que:

- Dada a heterogeneidade e o volume dos dados em trânsito na rede durante a execução de um AVC verifica-se a necessidade da utilização de técnicas que forneçam garantias específicas em relação a atraso e largura de banda para estas aplicações;
- Como a execução de uma sessão de um AVC já implica em um grande volume de dados transitando na rede, é interessante que as técnicas usadas para se fornecer tais garantias não usem mensagens adicionais que aumentem ainda mais a quantidade de dados em trânsito;
- Dentre as técnicas para Qualidade de Serviço existentes atualmente a que mais se aproxima do adequado para os AVCs são as técnicas que permitem diferenciação de serviços.

Este trabalho tem por meta avaliar o desempenho de técnicas de diferenciação de serviços em uma plataforma para AVCs em grande escala através de simulações. Com isso, este trabalho é composto dos seguintes objetivos:

a) estudo da integração de técnicas de diferenciação de serviços em uma plataforma para AVCs em grande escala já existente. Para este estudo, foi escolhida a plataforma SPLINE, por apresentar relativa simplicidade de implementação e ser conhecida no meio acadêmico;

b) definição e especificação de um subconjunto de funções da plataforma SPLINE utilizando diferenciação de serviços. Também serão feitas modificações na plataforma no que diz respeito aos protocolos da camada de transporte utilizados para transmissão dos dados. Isso se deve ao fato de que na SPLINE são utilizados os protocolos TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*) para transmitir tipos de mensagens diferentes. Neste estudo é proposto que todas as mensagens sejam enviadas utilizando o protocolo UDP para se evitar atrasos, e utilizar a diferenciação de serviços para se atribuir prioridades diferentes para tipos de mensagens diferentes;

c) implementação de um modelo deste subconjunto de funções no NS (*Network Simulator)* para validação através de simulações. O NS é um simulador de eventos discretos focado para o desenvolvimento de pesquisas em redes de computadores e amplamente usado em trabalhos acadêmicos(FALL; VARADHAN, 1997). Utilizando o NS, serão criados vários scripts de simulação objetivando a análise do desempenho da plataforma em vários cenários. Em caráter de comparação, a plataforma SPLINE sem modificações também é implementada no simulador.

1.3 ORGANIZAÇÃO DO TEXTO

Esta dissertação está organizada em cinco capítulos, de tal forma que:

- No capítulo 2, são apresentados os conceitos de realidade virtual, ambientes virtuais colaborativos e arquiteturas de rede para os mesmos. Ainda é mostrada uma classe

especial destes ambientes, e vários modelos desenvolvidos e suas características particulares.

- No capítulo 3 será apresentada uma visão geral das Arquiteturas de Qualidade de Serviço, e algumas propostas de técnicas de qualidade de serviço para Ambientes Virtuais Colaborativos.
- No capítulo 4 são apresentados uma visão do protocolo ISTP, usado pela plataforma SPLINE e a modelagem criada para o protocolo.
- No capítulo 5 são mostrados os experimentos conduzidos com a implementação do protótipo e os resultados obtidos.
- No capítulo 6 são apresentadas as conclusões obtidas no trabalho e as propostas para trabalhos futuros.

2. AMBIENTES VIRTUAIS COLABORATIVOS

Os sistemas de realidade virtual buscam a imersão de seus usuários em um ambiente tridimensional gerado por computador. Neste ambiente, também chamado de mundo virtual, os usuários podem interagir entre si e com o próprio ambiente de forma similar à interação no mundo real ou através de outros tipos de interação que não são viáveis na realidade. Assim, a realidade virtual permite uma interação homem-máquina mais natural e poderosa, tornando-se uma ferramenta bastante atrativa para as mais variadas áreas do conhecimento.

Alguns autores diferenciam o termo realidade virtual de ambiente virtual. Neste caso, o termo realidade virtual significa uma combinação de tecnologias cujas interfaces com os usuários humanos podem dominar seus sentidos de forma que eles interajam intuitivamente com um ambiente gerado por computador, imersivo e dinâmico. Por sua vez, ambiente virtual é uma representação tridimensional gerada por computador que apenas sugere um espaço real ou imaginário, não tendo o realismo fotográfico e o senso total de imersão como objetivos principais (RODRIGUES, 2003).

2.1 INTRODUÇÃO

A realidade virtual pode ser utilizada em diversas aplicações, como educação (RODRIGUES, 2003), (ALLISON, 1997), treinamento (CADORIN; OLIVEIRA, 2003), (GOBEL, 1996), telemedicina (ALBERIO; OLIVEIRA, 2003), arquitetura, trabalho cooperativo, visualização científica, entre outras. Estas aplicações podem ser classificadas de acordo com:

- o tipo de ambiente, se o ambiente é puramente virtual ou se há alguma interação com o mundo real;
- a participação do usuário, se o usuário é ativo - modificando o ambiente - ou passivo;

- a distribuição do sistema, se o sistema está concentrado em uma única estação ou se está dividido em estações distintas;
- e o número de usuários participando simultaneamente do mundo virtual.

Esta classificação visa determinar os tipos de aplicações de realidade virtual.

Os ambientes virtuais colaborativos (AVCs) são aqueles que possibilitam o compartilhamento do mundo virtual por usuários geograficamente dispersos (enquanto dispersão geográfica entre usuários possa ser uma distância consideravelmente longa, os ambientes virtuais executando em diferentes lugares devem estar operando e interagindo como se estivessem compartilhando o mesmo espaço físico). Este compartilhamento é importante por duas razões principais. Primeiro, a capacidade de interação com outros usuários em um mesmo ambiente virtual aumenta o envolvimento e a credibilidade do usuário neste ambiente. Segundo, este compartilhamento permite ampliar o alcance das aplicações de realidade virtual, possibilitando sua utilização nas mais diversas áreas do conhecimento.

Em educação à distância pode-se obter a interação entre alunos e professores, enriquecendo o aprendizado através de um ambiente relacionado ao tema da aula. Para treinamento, podem-se usar o ambiente virtual tanto para treinar situações de perigo ou em lugares de difícil acesso, como para treinar, em conjunto, membros de uma equipe que estão em lugares diferentes. Na área de trabalho cooperativo, há possibilidade de realização de tarefas em comum utilizando a interação com o ambiente como ferramenta (análise de um protótipo, a visualização do interior de um ambiente sendo projetado). Para entretenimento, além de diversos tipos de jogos, pode-se utilizar a realidade virtual para o turismo.

Um mundo virtual é composto por objetos. Os usuários deste mundo são representados por objetos chamados de avatares. Os avatares são imagens tridimensionais – que podem incluir um vídeo - e servem como entrada padrão para o usuário que o controla. Um avatar não é necessariamente a imagem de um ser humano. Em uma aplicação militar, por exemplo, um usuário pode estar controlando um avião e, portanto, seu avatar é a representação deste avião. Em um jogo, o avatar pode ser uma criatura fantástica ou um animal.

Nem todos os objetos de um mundo virtual são controlados pelos usuários. Existem objetos chamados agentes ou robôs, que são controlados por simulações de computador e

operam de forma relativamente autônoma. Agentes que exibam comportamento similar ao humano são importantes para a criação de ambientes mais complexos. Eles permitem a sensação de um número maior de participantes, sem o custo de um ambiente com muitos usuários. Além destes dois tipos de objetos, há também objetos fixos que ajudam a compor o ambiente virtual.

Estes objetos estão representados em um ambiente compartilhado que deve apresentar as mesmas características para todos os usuários. Ao entrar em um mundo virtual, um participante pode ver o avatar dos outros que estão neste espaço e os outros podem ver o seu avatar, percebendo o comportamento uns dos outros imediatamente. As informações que descrevem um objeto e seu comportamento são o seu estado. Como um participante pode se movimentar pelo mundo virtual, interagir com ele (criando ou modificando objetos) e interagir com outros participantes, seu estado não permanece fixo. Por isto, torna-se necessário que os sistemas, para suportar este tipo de aplicação, notifiquem as modificações de estado a todos os participantes do ambiente virtual, garantindo que todos os usuários tenham uma visão consistente deste ambiente compartilhado.

Idealmente, os sistemas de realidade virtual devem oferecer uma interação com o usuário completamente imersiva. Porém, ainda existem muitas restrições que devem ser contornadas para que esta sensação de imersão seja satisfeita. Os sistemas distribuídos de realidade virtual são sistemas complexos, que representam vários sistemas agregados em um só. Eles são sistemas distribuídos, precisando gerenciar a utilização dos recursos da rede de comunicação, contornando as perdas de dados, as falhas na rede e a concorrência. São também aplicações gráficas, precisando manter uma taxa de exibição de quadros em tempo real e dividir o tempo de processamento entre a exibição de imagens gráficas e as outras tarefas. Além disto, são aplicações interativas, precisando processar em tempo real as entradas dos usuários.

Os ambientes virtuais colaborativos que possuem uma extensão geográfica elevada ou um grande número de usuários são conhecidos como Ambientes Virtuais Colaborativos em grande escala. A Internet é a plataforma ideal para os ambientes virtuais colaborativos em grande escala, devido ao grande número de usuários que a utilizam em todo mundo. A infra-estrutura necessária para a representação dos mundos virtuais está sendo disponibilizada nos navegadores (*browsers*) da World Wide Web (WWW). Por exemplo, a linguagem VRML

(*Virtual Reality Modeling Language*) (VRML, 2006), permite aos usuários carregar modelos tridimensionais interativos pela WWW. Entretanto, atingir interação multi-usuário em ambientes virtuais colaborativos usando a *Internet*, onde a largura de banda disponível é geralmente escassa, é um grande desafio.

Existem diversos fatores a serem considerados em tal cenário. A maior limitação é a utilização de largura de banda da rede, particularmente quando o sistema incorpora novos usuários. Muitos usuários são conectados à *Internet* via linhas telefônicas de baixa capacidade, com uma largura de banda de não mais de 56 kbps. Apesar das redes locais possuírem velocidade o bastante para suportar tais ambientes virtuais, deve-se tomar cuidado no projeto de uma arquitetura de comunicação, para permitir que tais aplicações funcionem na *Internet* com os usuários conectados ao sistema usando linhas com baixa capacidade. Um segundo problema relacionado aos AVCs é a latência da rede. Quando uma atualização chega a um receptor particular, ela pode estar várias centenas de milisegundos atrasada, e a fonte já pode ter colocado uma nova atualização em trânsito, o que pode causar inconsistências no sistema. Outro fator que pode gerar inconsistências nos AVCs é a perda de dados na rede. Para solucionar tal problema, deve-se empregar protocolos confiáveis, mas que geram como uma conseqüência indesejada, um aumento ainda maior da latência da rede.

Sabe-se que a quantidade, o tipo e o nível da sensibilidade da informação que deve ser trocada entre AVCs guia os requerimentos de comunicação e dependem do número e do tipo de usuários participantes, e da natureza e nível de interação entre estes usuários. A troca de pacotes entre usuários poderia incluir a localização de cada objeto, sua interação, orientação ou mensagens baseadas em texto. Um objeto móvel típico gera cerca de 10 pacotes por segundo. Com tanta informação sendo trocada entre usuários em diferentes localidades, é necessário o uso de comunicação *multicast* para minimizar a largura de banda da rede usada e para reduzir os dados de entrada para cada usuário, de forma que cada um receba apenas aqueles pacotes que sejam de seu interesse. Isso acontece porque com a separação dos usuários do mundo virtual em grupos *multicast*, apenas os usuários dentro de um grupo irão receber mensagens daquele grupo.

Um requisito importante dos AVCs é a manutenção de consistência do mundo virtual. Por exemplo, em um ambiente virtual para treinamento de motoristas, todos os motoristas

devem perceber os movimentos de todos os objetos imediatamente para controlar o veículo de forma correta. As inconsistências poderiam acarretar atropelamentos e batidas sem que o motorista tivesse meios para reagir evitando este acidente.

Outro requisito destes sistemas é o controle de concorrência no acesso aos objetos do ambiente. Além de atualizar o estado de todos os objetos constantemente, um sistema distribuído de realidade virtual é responsável por controlar o acesso a estes objetos. Se dois participantes tentarem acessar simultaneamente um mesmo objeto, apenas um deve ser capaz de concluir a ação. Suponha um ambiente virtual que representa um jogo de guerra em que dois avatares representam seres humanos que lutam por uma arma, por exemplo. Para que a representação do mundo não fique inconsistente, apenas um deles deve conseguir a arma. Logo, o ambiente virtual deve ter alguma política de acesso aos objetos, mantendo a exclusão mútua.

Além destes dois requisitos, também é importante para este tipo de sistema a viabilidade de comunicação direta entre os participantes. Esta comunicação pode ser realizada através de textos, gestos ou voz. A capacidade de comunicação adiciona uma sensação de realismo necessária para qualquer ambiente simulado.

2.2 MODELOS DE ARQUITETURAS DE REDE PARA AMBIENTES VIRTUAIS COLABORATIVOS

O fluxo de informação e a colaboração entre usuários em AVCs são controlados pela infra-estrutura de comunicação subjacente. Na fase de inicialização, ou seja, quando o usuário está tentando se conectar ao sistema, o mundo virtual junto com seu conteúdo deve ser enviado para o novo usuário. Uma vez que o usuário se une ao AVC, atualizações de eventos são geradas quando um usuário cria/remove um objeto, transfere sua propriedade, deixa o sistema, ou se move dentro do AVC. A arquitetura de comunicação de rede define como os dados irão fluir entre os participantes do AVC e pode ser classificada dentro de três modelos diferentes de arquitetura de rede, segundo o trabalho de (ERASLAN et al., 2003).

2.2.1 ABORDAGEM CLIENTE-SERVIDOR

Em um sistema cliente-servidor, há um banco de dados central que guarda o conteúdo do mundo virtual e controla os movimentos dos objetos e seus comportamentos. Quando um novo cliente se une à sessão, o servidor o acrescenta a seu banco de dados. As mensagens de controle ou de atualização são enviadas para este servidor, e então o servidor as repassa para os outros membros.

Servidores podem reduzir o tráfego de mensagens aplicando alguns mecanismos de filtragem e comprimindo vários pacotes em mensagens únicas, eliminando os fluxos de mensagens redundantes. Entretanto, essa arquitetura é um gargalo tanto para a rede quanto para o processamento se o número de participantes e o conteúdo do mundo virtual aumentam. A inclusão de um servidor em cada transação significa que a mensagem pode não ser capaz de pegar o caminho mais curto entre nós da fonte ao destino. Além disso, as mensagens podem gastar tempo sendo processadas no servidor, o que aumenta o tempo total de transmissão. O servidor é o ponto de falha da rede. Usar um servidor centralizado para passagem de mensagens dentro dos mundos virtuais o limita a dezenas de usuários por causa da contenção de dados de entrada e saída. Um exemplo desse tipo de arquitetura é a de jogos *em LAN Houses*, onde um servidor coordena as simulações sendo executadas em poucas máquinas.

2.2.2 COMUNICAÇÃO MULTIPONTO

Arquiteturas de comunicação multiponto usam o protocolo IP *Multicast* (DEERING, 1989) para aumentar a eficiência do sistema evitando duplicação desnecessária de pacotes. Com este método, pacotes são enviados para todos os membros de uma vez e a latência é reduzida para quase a metade comparada à abordagem cliente-servidor, onde mensagens devem primeiro viajar até o servidor que as redistribui para os outros membros. Em comunicação multiponto, usuários primeiro se unem a um grupo *multicast* para se tornarem um membro do sistema e então todas as mensagens são enviadas para este endereço *multicast.*

Esta arquitetura tem boa escalabilidade para um grande número de usuários já que ela utiliza a largura de banda disponível de forma eficiente. Entretanto, sabemos que os usuários, cuja entrada é posterior a dos demais devem ser considerados na utilização desta arquitetura. Um problema, seria como os usuários com entrada posterior estarão cientes dos usuários que já se uniram ao AVC. Já que não há repositório central, um usuário pode não ser capaz de conseguir o conhecimento sobre os outros participantes do AVC. Fazer com que todos os usuários periodicamente enviem mensagens para indicar que estão ativos, resolveria essa questão.

A comunicação *multicast* está se tornando a forma padrão de se estabelecer à comunicação em ambientes virtuais colaborativos em grande escala. Ela provê uma eficiente utilização dos recursos de rede, e também ajuda o AVC a diferenciar vários tipos de dados através do uso de vários endereços *multicast* diferentes. Os grupos *multicast* têm importância fundamental na redução do número de mensagens em trânsito em AVCs em grande escala. Eles participam de uma técnica chamada de filtragem de dados.

A filtragem de dados é um mecanismo que explora o fato de um participante não interagir com todos os objetos de um mundo virtual simultaneamente (GREENHALGH; BENFORD, 1997). Nesse mecanismo, os participantes do mundo virtual são agrupados definindo escopos de interação, de forma que as interações entre os participantes fiquem confinadas nestes escopos. O escopo de interação pode ser visto como um filtro para as informações a serem recebidas.

Tais escopos podem ser definidos por afinidades espaciais, temporais ou funcionais. Dentre elas, a mais usada é a filtragem espacial, que agrupa os participantes que compartilham a mesma região do mundo virtual. Ela limita a troca de mensagens a um escopo de interação, reduzindo o número total de mensagens trocadas no sistema de comunicação. A implementação deste mecanismo pode ser feita através de comunicação multiponto, onde os escopos de interação são mapeados em grupos *multicast*. Dessa forma, os participantes só recebem mensagens dos grupos *multicast* dos quais fazem parte, ou seja, eles só receberão mensagens dos usuários que se encontram em áreas do mundo virtual que sejam de seu interesse.

2.2.3 COMUNICAÇÃO MULTIPONTO COM UM REPOSITÓRIO CENTRAL

Apenas o uso de comunicação *multicast* não é suficiente para fazer um AVC consistente por duas razões. Primeiro, membros desconectados devido à falhas continuariam aparecendo no mundo virtual como observadores do sistema que não executam nenhuma ação, mas observam outros usuários e objetos. Segundo, quando um membro se une a um grupo *multicast*, ele pode não ter conhecimento de alguns membros e objetos no sistema até que eles enviem uma mensagem de que estão ativos ou executem alguma ação gerando um evento de atualização. As mensagens de estado ativo, entretanto, consomem uma significativa porção de largura de banda e com isso não são a opção mais viável.

Vários sistemas usam um repositório central de forma a manter o controle dos usuários no AVC, provendo-os com uma visão consistente do ambiente, e ajudando-os a se unirem ao sistema ou se recuperarem de desconexão temporária rapidamente. Estes sistemas utilizam um repositório central enquanto tomam vantagem da eficiência provida pelo *multicast* da camada de rede. Utilizando um endereço *multicast* conhecido, participantes de um AVC podem anunciar sua presença e também aprender sobre a presença de outros participantes.

2.3 PLATAFORMAS PARA AMBIENTES VIRTUAIS COLABORATIVOS EM GRANDE ESCALA

Nesta seção serão mostradas várias plataformas para AVCs existentes, bem como suas características particulares, vantagens e desvantagens.

2.3.1 SIMNET E DIS

Um dos primeiros ambientes virtuais colaborativos criados foi o SimNet (*Simulator Networking*) (GARVEY; MONDAY, 1988), um sistema desenvolvido na década de 80 para o Departamento de Defesa americano com o objetivo de promover o treinamento militar

utilizando uma rede de simuladores. O SimNet se baseia em uma arquitetura objeto-evento e a noção de nós de simulação autônomos.

A arquitetura objeto-evento modela o mundo virtual como uma coleção de objetos. As interações entre estes objetos são realizadas através de um conjunto de eventos, que são mensagens trocadas pela rede indicando uma alteração no estado do objeto. Os nós de simulação são autônomos porque são responsáveis pelo envio de mensagens periódicas (*heartbeats*) atualizando o estado dos objetos sob seu controle. Os receptores, por sua vez, devem receber estas mensagens e atualizar o estado dos objetos. As PDUs (*Protocol Data Units*) definidas para o SimNet são específicas para sistemas militares. Por exemplo, a PDU *Fire* representa o disparo de uma arma, enquanto a PDU *Impact* comunica que a arma acertou um objeto. O SimNet, no entanto, não define um conjunto de PDUs que supram os requisitos de qualquer tipo de simulação militar. Além disso, o formato das PDUs e a arquitetura não estão suficientemente documentados.

Por isto, após o desenvolvimento do SimNet sentiu-se a necessidade de padronização do protocolo utilizado por ele, possibilitando a interoperabilidade entre qualquer tipo de participante e a máquina. Esta preocupação deu origem ao desenvolvimento do protocolo para Simulações Distribuídas Interativas (*Distributed Interactive Simulation – DIS*), padrão IEEE 1278. A arquitetura objeto-evento foi projetada para suprir um maior número de requisitos de simulações militares que o SimNet.

O protocolo definido para o DIS especifica vinte e sete PDUs, dentre as quais apenas 4 para interação entre nós no ambiente virtual: *Entity State* – enviada para a atualização de estado e *Fire*, *Detonation* e *Collision*, todas ligadas a eventos nas simulações de batalhas. A maioria dos sistemas compatíveis com o padrão DIS implementa apenas estas 4 PDUs. No DIS, a troca de pacotes é realizada através do protocolo UDP (*User Datagram Protocol*), que não é confiável. As perdas de pacotes são recuperadas na recepção do próximo pacote, para isto, mesmo objetos que não existem mais no ambiente devem enviar PDUs *Entity State*. A especificação do DIS estabelece o atraso de 100 ms para simulações fortemente acopladas e de 300 ms para simulações fracamente acoplada. As simulações implementadas com o DIS, realizadas na rede do Departamento de Defesa Americano para Simulações (*Defense Simulation Internet* -DSI) suportam até 300 participantes simultâneos.

Porém, existem alguns problemas associados a escalabilidade do protocolo DIS, como a necessidade de difusão periódica de atualizações de estado, mesmo para objetos estáticos, e a replicação dos modelos e banco de dados do mundo em cada simulador. Com o objetivo de tornar escaláveis os sistemas colaborativos de realidade virtual, surgiram novas propostas baseadas no que foi aprendido com os erros e acertos do SimNet e do DIS.

2.3.2 RTI

Atualmente, a especificação do DIS está parada e o Departamento de Defesa americano redirecionou seus esforços para o desenvolvimento de um modelo cliente/servidor distribuído de barramentos de objetos, a chamada *HighLevel Architecture/Runtime Infrastructure (*HLA/RTI*)* (HOOK et al., 1996). A HLA/RTI é um sistema que oferece serviços necessários para simulações distribuídas. O objetivo da HLA é o suporte adequado a todos os tipos de simulação do Departamento de Defesa americano, aumentando o grau de interoperabilidade e reutilização destas simulações. O serviço de distribuição de dados da RTI visa uma maior eficiência e flexibilidade de forma a suportar uma maior variedade de aplicações.

Os serviços da infraestrutura RTI são descritos pela especificação de interface da arquitetura HLA. Estes serviços incluem o gerenciamento do tempo, das federações (conjunto de simulações), dos objetos, do acesso aos objetos e da declaração de dados. A distribuição e o controle dos objetos do RTI são feitos através da arquitetura para comunicação distribuída CORBA (*Common Object Request Broker Architecture*) (DERRIGI et al., 1999). Para modificar o estado de um objeto, basta alterar o atributo necessário. Apenas este atributo vai ser enviado para as outras simulações. A infra-estrutura separa os participantes em regiões, definindo as regiões em que eles têm interesse em receber ou enviar atualizações. Através das interseções nestas regiões, a infra-estrutura determina quais participantes estão interagindo, devendo trocar mensagens.

2.3.3. NPSNET-IV

O NPSNET-IV (MACEDONIA et al., 1994), desenvolvido pela *Naval Postgraduate School*, é um sistema de simulação para treinamento militar. As versões anteriores do NPSNET utilizavam redes locais e o protocolo SimNet para interoperabilidade. O principal objetivo desta última versão é obter um ambiente virtual mais escalável, permitindo o treinamento de muitos (mais de 100.000) usuários simultâneos. Atualmente, este sistema suporta centenas de participantes utilizando a *Internet.*

A fim de obter um sistema mais escalável, foram adotados o protocolo DIS, para comunicação entre simuladores independentes, e o protocolo IP *multicast*, para suportar simulações distribuídas escaláveis em redes de longa distância.

O NPSNET- IV utiliza um gerente de área de interesse, dividindo o ambiente virtual em um conjunto de partições menores (classes) para reduzir a carga computacional nas estações, minimizar a troca de informações na rede e restringir os problemas de confiabilidade. Os objetos do mundo virtual são divididos em classes espaciais, temporais e funcionais. Em relação às classes espaciais, o ambiente virtual é dividido em áreas hexagonais de tamanho fixo, chamadas células. Estas classes são associadas a grupos IP *multicast*. O modelo de comunicação suporta tipos diferentes de interações, para garantir um certo grau de confiabilidade sem prejudicar as interações em tempo real.

O grupo que criou o NPSNET propôs, recentemente, o protocolo VRTP (*Virtual Reality Transfer Protocol*) com o objetivo de viabilizar ambientes virtuais na Web. A idéia é fazer uma extensão para o protocolo HTTP (*Hypertext Transfer Protocol*), oferecendo um suporte de rede para a linguagem VRML. O VRTP deve oferecer suporte aos requisitos dos ambientes virtuais escaláveis, permitindo desde comunicação cliente/servidor até comunicação direta entre dois participantes.

Foi lançada a quinta versão da plataforma NPSNET, o NPSNET-V, com algumas inovações em sua estrutura. Com uma arquitetura baseada em componentes de software e a capacidade de introdução de novos módulos em tempo de execução, essa nova versão traz

uma maior flexibilidade na construção de ambientes virtuais colaborativos, tirando proveito das características da linguagem Java.

2.3.4 DIVE

DIVE (*Distributed Interactive Virtual Environment*) (HANGSAND, 1996), desenvolvido no *Swedish Institute of Computer Science*, tem como principal enfoque a interação entre os usuários. Por isso, as aplicações visadas são as que exigem colaboração entre os usuários. Basicamente, a DIVE representa uma sala de conferência para interação compartilhada à distância.

Este sistema busca a escalabilidade através da utilização de um protocolo multidestinatário confiável com reconhecimentos negativos e a replicação parcial da base de dados. O particionamento dos participantes em grupos é em um modelo de dois níveis, onde existe um único grupo *multicast*, no topo da hierarquia, ao qual todos os participantes podem se associar no primeiro nível, e um grupo *multicast* para cada usuário no nível mais baixo. Quando ocorre o envio de uma mensagem de atualização, tenta-se primeiro achar os grupos referentes aos usuários. Caso não seja possível, o endereço do grupo no nível superior é usado para transmissão.

A base de dados do DIVE é dinâmica. O sistema tem a capacidade de adicionar novos objetos e modificar a base de dados de maneira confiável. O controle de concorrência dos acessos aos objetos do mundo virtual é feito através de um mecanismo de bloqueio (*locking*) do objeto acessado. Alguns testes realizados com o DIVE mostram que ele funciona em redes de longa distância para 20 participantes com atrasos menores que 200 ms.

2.3.5 MASSIVE

O sistema MASSIVE (*Model, Architecture and System for Spatial Interaction in Virtual Environments*) (GREENHALGH; BENFORD, 1997) é um sistema colaborativo de realidade virtual que suporta atividades colaborativas entre usuários com equipamentos heterogêneos. Ele introduz um modelo espacial de interação com diferentes níveis de percepção entre os participantes.

Neste modelo espacial, é chamado de aura o volume do espaço dentro do qual a interação é possível. A interação entre dois objetos se torna possível quando as suas auras colidem. Portanto, os objetos só trocam notificações quando há uma colisão de auras, ou seja, é feita uma filtragem dos dados de forma que apenas objetos que estão em um mesmo escopo espacial troquem informações.

Outro componente do modelo espacial, o conhecimento, trata do controle de interação ou comunicação entre dois objetos uma vez que a interação foi permitida devido a uma colisão de auras. O conhecimento de um objeto sobre o outro quantifica a importância deste ultimo objeto. O nível de conhecimento é baseado na distância relativa entre os participantes e mediado por um mecanismo de negociação, feito por um gerente de colisão. Quando o gerente de colisões identifica uma colisão os participantes são notificados e estabelecem uma conexão pontoaponto.

A responsabilidade de detecção de colisão é dividida entre vários gerentes de colisão, buscando melhorar a escalabilidade do sistema.

Uma importante contribuição feita por MASSIVE-2 foi o gerenciamento de conhecimento e comunicação entre participantes de uma multidão. A estrutura para tratar multidões é baseada no modelo espacial de interação e numa extensão para este modelo chamada objeto mediador. Colisões de auras levam a estabelecimento de conexão. A qualidade da informação que é transmitida depende do nível de conhecimento que um observador tem do observado. Um objeto mediador é um objeto independente que afeta o conhecimento entre dois outros objetos.

As multidões são uma classe específica de objetos mediadores que suportam grupos de pessoas potencialmente grandes. As multidões devem ter efeito assimétrico no conhecimento. Vistos do lado de fora, indivíduos na multidão estão escondidos e são substituídos por uma visão agregada da multidão como um todo. Dentro da multidão, indivíduos podem interagir normalmente uns com os outros. Tipicamente, eles também vão ter conhecimento dos objetos fora da multidão individualmente.

A primeira versão de MASSIVE usava um protocolo orientado a conexão confiável para carregar mensagens entre agentes. Entretanto, isto resultava em um desempenho em tempo real ruim e grande uso de largura de banda, o que causava uma baixa escalabilidade com o crescimento do número de habitantes no mundo virtual. Para resolver estes problemas, o MASSIVE agora usa um protocolo *multicast* e representa grupos *multicast* como entidades dentro do ambiente. Por exemplo, uma sala de conferência poderia ser modelada como uma entidade feita de paredes, janelas, uma porta e um grupo *multicast* que é usado por entidades dentro da sala para se comunicarem entre si. Entidades que contêm grupos podem ser controladas da mesma forma que outras entidades e podem ser feitas para seguirem multidões, por exemplo, para proverem alta largura de banda para comunicação entre os membros da multidão.

2.3.6 SPLINE

A SPLINE (*Scalable Platform for Large Interactive Networked Environment*) (BARRUS et al., 1996), foi desenvolvida no *Mitsubishi Electric Research Labs* (MERL).

Na plataforma SPLINE, o mundo virtual é modelado por uma base de dados que contém informações sobre esse mundo, chamada Modelo de Mundo. Este modelo está parcialmente distribuído entre os processos SPLINE. As aplicações interagem umas com as outras através da modificação desta base de dados e da observação das modificações feitas pelos outros usuários.

O mundo é dividido em áreas chamadas *locales,* que não possuem tamanho nem formas fixas e podem ser processadas separadamente. Cada *locale* está associado a um grupo *multicast* e possui um sistema de coordenadas próprio. Cada participante possui uma cópia parcial da base de dados de acordo com o seu *locale*. Servidores de *locale* são responsáveis por manter um registro do estado de todos os objetos em um determinado *locale.* Dessa forma, um processo SPLINE que se interessa por um *locale* pode receber a informação atualizada sem ter que contatar cada processo pertencente ao *locale*.

A SPLINE é uma plataforma que representa bem as plataformas para AVCs em grande escala que utilizam filtragem de dados espacial. Além disso, ela possui eficiência comprovada, é conhecida no meio acadêmico (ANDERSON et al, 1995) e de implementação não muito complexa. Por conta de sua importância para este trabalho, ela será melhor descrita em um capítulo específico.

2.3.7 VELVET

Todas as plataformas apresentadas até agora tentaram reduzir a carga computacional nas estações e a quantidade de informação a ser transmitida através dos enlaces usando uma divisão do Mundo Virtual em partes, cada uma associada a um grupo *multicast*, de forma que os usuários só recebam informações dos usuários que se encontram na mesma partição do Mundo Virtual. Ainda que esta abordagem seja eficiente para várias situações, ela se mostra inadequada para alguns cenários. Suponha que tenhamos um Museu Virtual, o qual algumas centenas de usuários estão visitando. Se todos eles decidirem ver o mesmo quadro, todas as estações terão que lidar com todas as centenas de fluxos de dados, já que todos os usuários se encontrarão na mesma partição do Mundo Virtual. Para este tipo específico de situação, tal abordagem falharia no objetivo de reduzir a carga computacional nas estações.

Recentemente, a plataforma VELVET (*Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs*) (OLIVEIRA; GEORGANAS, 2003), trouxe uma nova proposta para solução deste problema. Usando um grupo *multicast* para cada objeto do Mundo Virtual e um modelo de Mundo Virtual Paralelo, que particiona o mundo virtual seguindo um conjunto de

métricas que não necessariamente estão relacionadas com distância geográfica, ela propõe uma solução para o problema citado acima e ainda trata de outro de grande relevância, a heterogeneidade de hardware e largura de banda entre os participantes. Se em um grupo de participantes, alguns possuírem conexões de rede e capacidades computacionais muito superiores às de outros participantes, tal situação gerará um problema para os participantes mais deficientes em termos de largura de banda e processamento, pois eles não conseguirão processar os dados da simulação da mesma forma que os outros usuários, comprometendo assim a experiência interativa. Para solução deste problema, a plataforma VELVET provê um modelo no qual, cada usuário pode receber as informações do Mundo Virtual de acordo com a sua capacidade, permitindo que ele participe da simulação em um nível menor, mas ainda sim satisfatório.

3. QUALIDADE DE SERVIÇO

Na *Internet* tradicional, todas as conexões ganham o mesmo tratamento da rede. Tal prática se encontra em contraste com outros conceitos de redes, como as redes ATM (*Asynchronous Transfer Mode*) (KAMIENSKI; SADOK, 2000) que pode oferecer requisitos de qualidade de serviço para as conexões ao custo de uma grande quantidade de sinalização e processamento relacionados à aceitação de novas conexões e mantendo as garantias das conexões já estabelecidas. Entretanto, como os recursos da rede são limitados, oferecer garantias de desempenho requer a rejeição de novas conexões se não existirem recursos disponíveis. Tal prática está em contraste com a natureza de melhor esforço, ou seja, sem garantias, da *Internet* de hoje.

A Qualidade de Serviço pode ser definida como uma linha de pesquisa que busca o desenvolvimento de técnicas para prover classes de serviços diferenciados e níveis de prioridade para fluxos de dados. Seu objetivo é garantir que os tráfegos de aplicações críticas como os AVCs, por exemplo, tenham um desempenho aceitável na rede, mesmo em períodos de congestionamento. Em situações onde a largura de banda é limitada e os AVCs e outras aplicações multimídia como vídeo-conferência por exemplo, concorrem por recursos escassos, a Qualidade de Serviço se torna uma ferramenta fundamental para garantir que todas as aplicações possam coexistir e funcionar com níveis de desempenho adequados.

As técnicas de Qualidade de Serviço têm como objetivo comum acelerar a entrega de pacotes de aplicações críticas, enquanto compartilham os recursos da rede com aplicações não-críticas. Elas também dão aos gerentes de rede controle sobre as aplicações de rede, aumento na relação custo-eficiência de redes WAN(*Wide Area Network*) e permitem uma diferenciação de serviços permitindo que desenvolvedores de aplicações façam um balanceamento dos níveis de prioridade dos serviços para a satisfação dos usuários com eficiente utilização de acesso para minimizar a latência da rede. Eficientemente priorizando uma variedade de tipos de tráfego e requerimentos para tráfego de aplicações críticas e sensíveis ao tempo como os AVCs, tais técnicas trazem uma série de benefícios como melhor

controle sobre recursos, uso dos recursos da rede de forma mais eficiente, coexistência entre aplicações críticas e as fundações para uma rede totalmente integrada no futuro.

Em uma arquitetura de qualidade de serviço é importante que se procure respeitar as características de tráfego das diferentes aplicações. Os requisitos importantes para um tipo de aplicação podem não ser fundamentais para outras. Assim, os parâmetros de qualidade de serviço menos importantes, aqueles que influenciam menos o desempenho percebido por um dado usuário podem ser explorados. Nesse caso, podem ser planejados serviços que realizem uma degradação em alguns parâmetros, quando houver uma sobrecarga do sistema, para otimizar o atendimento de uma forma global. Isso permitiria que um maior número de usuários se mantivesse satisfeito, mesmo que uma pequena queda na qualidade de serviço fosse necessária. Nesse momento, deve ser identificado o que é mais importante para cada tipo de aplicação: a confiabilidade, a disponibilidade/conectividade do serviço, etc.

A qualidade de serviço sobre o protocolo IP se refere ao desempenho do fluxo de pacotes IP através da *Internet*. Ela é caracterizada por uma série de métricas, incluindo disponibilidade de serviço, atraso, variação de atraso, vazão e taxa de perda de dados. As aplicações são classificadas de acordo com as suas necessidades específicas, baseando-se nesses parâmetros. A maioria das aplicações é extremamente sensível a alguns desses parâmetros de qualidade de serviço, mas por outro lado há quase sempre uma flexibilidade com relação aos parâmetros restantes. O transporte de dados, por exemplo, é geralmente influenciado pela taxa de perdas no meio, apresentando uma maior tolerância ao atraso. Tráfegos de tempo real como voz, comportam-se de forma oposta, sendo altamente sensíveis a atrasos e relativamente robustos a perdas. De uma forma geral, o atraso é importante para aplicações que necessitam de interatividade. Os fluxos que representam mídias contínuas (voz e vídeo) a serem reproduzidas no receptor (os chamados *streams* de dados) necessitam de pequenos valores de *jitter,* de forma a reduzir o tamanho dos *buffers* de reprodução da mídia. A taxa de perdas deve ser mantida baixa nas aplicações onde a confiabilidade na transmissão é fundamental. Assim, as retransmissões podem ser evitadas e o desempenho do sistema é elevado. A alta vazão é significativa quando se realizam transferências de grande volume de dados, assim como para o transporte de mídias que demandam alta utilização dos enlaces.

Duas arquiteturas de qualidade de serviço foram propostas pelo *Internet Enginnering Task Force* (IETF) para a Internet: A Arquitetura de Serviços Integrados (também conhecida como *IntServ*) e a Arquitetura para Serviços Diferenciados (também conhecida como *DiffServ*).

3.1 ARQUITETURA DE SERVIÇOS INTEGRADOS

A arquitetura *IntServ (*BRADEN et al., 1997), estende o modelo arquitetural IP para suportar tanto fluxos de tráfego transmitidos por melhor esforço quanto de tempo real. *IntServ* sugere que para um fluxo receber um nível de serviço desejado em termos de largura de banda ou atraso, é necessário instalar e manter um estado por fluxo específico na rede. A arquitetura *IntServ* define três classes de serviço:

- Serviço Garantido: esta classe de serviço suporta fluxos de tráfego de tempo real que necessitam de largura de banda, um limite para o atraso e garantia de ausência de descarte de seus pacotes nas filas dos roteadores.
- Carga Controlada: esta classe se aproxima do serviço por melhor esforço sobre uma rede com uma carga pequena, oferecendo um serviço mais flexível e garantindo que limites de medidas estatísticas (como o atraso médio, por exemplo), não serão violados mais vezes que no caso de uma rede sem carga.
- Melhor Esforço: também conhecido com ausência de qualidade de serviço, o serviço por melhor esforço é basicamente conectividade com nenhuma garantia. A Internet, tal como ela é hoje em dia, é um bom exemplo de serviço por melhor esforço.

Ainda que a especificação original da arquitetura *IntServ* não ressalte como obrigatória à utilização de um protocolo de sinalização específico, o Protocolo de Reserva de Recursos (*Resource reSerVation Protocol - RSVP*) (BRADEN et. al, 1997), é o protocolo mais comumente usado para sinalização entre os nós componentes de um domínio *IntServ.* O protocolo assume que os recursos da rede estão reservados para cada fluxo que requeira qualidade de serviço em cada nó roteador no caminho entre transmissor e receptor usando sinalização fim-a-fim. A arquitetura *IntServ*, usa o protocolo RSVP entre emissores e receptores para sinalização por fluxo. Dessa forma, as mensagens RSVP atravessam a rede para requisitar recursos. Roteadores ao longo do caminho devem manter um estado leve (*soft-*

state) para os fluxos RSVP, ou seja, um conjunto de informações sobre a reserva que possuem um tempo máximo de validade, depois do qual se expira e os dados são apagados. Isso permite que o protocolo facilmente ofereça suporte a mudanças nos membros dos grupos, como também se adapte automaticamente a alterações no roteamento.

Dessa forma, o RSVP é o protocolo utilizado pelas aplicações para realizar pedidos de reserva de recursos da rede. A resposta da rede é a admissão ou rejeição explícita do pedido. Combinando-se a arquitetura IntServ com o protocolo RSVP, obtêm-se um modelo onde cada fluxo tem suas necessidades sinalizadas para os elementos de rede através da utilização do RSVP. Os elementos de rede então aplicam procedimentos de controle de admissão baseados em suas necessidades. Além disso, o elemento de rede configura mecanismos de controle de tráfego para garantir os serviços estabelecidos para aquele fluxo isoladamente dos restantes. Dessa forma, a sinalização através do RSVP configura classificadores de pacotes por micro-fluxo nos roteadores *IntServ* ao longo do caminho de comunicação.

A sinalização RSVP é útil para coordenar as técnicas de controle de tráfego e é uma peça fundamental na configuração de um serviço de qualidade de serviço fim-a-fim no topo de redes corporativas onde os fluxos dos usuários podem ser gerenciados no nível de usuário.

Apesar de possuir uma eficiência comprovada em redes de pequeno porte, a arquitetura *IntServ*, mostrou possuir vários problemas que não a tornam escalável o bastante para que seja usada em uma rede de grande dimensão como a *Internet* pública. Os principais problemas são:

- A quantidade de informação de estado cresce proporcionalmente ao número de fluxos que um roteador tem que tratar. Isso impõe uma grande sobrecarga aos roteadores, em termos de capacidade de armazenamento e processamento considerando que roteadores de núcleo na Internet tratam simultaneamente milhares de fluxos.
- Para cada fluxo deve haver sinalização a cada nó (sistema final ou roteador). A troca de informações de sinalização é muito grande, inclusive porque o RSVP trabalha com o conceito de *soft-state*, prejudicando a escalabilidade.
- As exigências para os roteadores são bastante altas. Todos têm que implementar RSVP, classificação, controle de admissão e escalonamento de pacotes. Entretanto, os roteadores, com uma finita quantidade de *buffers* e CPU podem não ser capazes de

manter informações de estado de milhares de fluxos RSVP enquanto processam mensagens de atualização RSVP freqüentes.

3.2 ARQUITETURA DE SERVIÇOS DIFERENCIADOS

Em oposição ao modelo *IntServ*, a arquitetura de Diferenciação de Serviços, *DiffServ* (BRADEN et. al., 1994), não busca um controle de qualidade de serviço baseado em necessidades de micro-fluxos individuais, oferecendo qualidade de serviço na Internet com escalabilidade: sem estado para cada fluxo e sem sinalização para cada nó. O provisionamento da qualidade de serviço é implementado a nível de agregação de fluxos. Isso é possível a partir da observação que os diferentes micro-fluxos das diversas aplicações podem ser classificados em algumas poucas categorias (classes de tráfego) dependendo das características das aplicações refletidas em suas necessidades de qualidade de serviço. Assim as garantias de qualidade de serviço não são por fluxo individual, mas por agregados de tráfego. As especificações dos agregados e do serviço estão descritas por contratos, chamados de Acordos de Nível de Serviço (*Service Level Agreements* – SLA).

A arquitetura DiffServ utiliza um campo do cabeçalho dos pacotes IP chamado de *DiffServ CodePoint* (DSCP) para diferenciar conjuntos de fluxos, tratando-os como um conjunto ou agregado. No interior do domínio DiffServ os agregados de fluxos de cada classe são tratados e encaminhados nó a nó de acordo com comportamentos pré-estabelecidos, os "Comportamentos por Nó" (*Per Hop Behavior* - PHB's). Um PHB é o comportamento do encaminhamento de pacotes observável externamente, aplicado por um nó DiffServ a um agregado de fluxos (BLAKE et al., 1998).

Em cada borda entre domínios *DiffServ,* os aspectos técnicos e comerciais dos serviços oferecidos são definidos na forma de SLAs, que especificam as características gerais, o desempenho que os usuários podem esperar desses serviços e formas de cobrança e tarifação. Como os serviços *DiffServ* são unidirecionais, as duas direções têm que ser tratadas separadamente em um SLA. O oferecimento de um serviço fim a fim é realizado através da concatenação de vários domínios *DiffServ*, onde os SLAs são negociados em cada uma das

bordas entre os domínios existentes. Um domínio usuário de um serviço não estabelece um SLA direto com o domínio final do serviço, a não ser que haja uma ligação direta entre eles. Caso contrário, ele negocia com o próximo domínio *DiffServ* no caminho e assim por diante até o domínio final.

Cada classe de serviço dentro de um domínio DiffServ deve estar adequadamente provisionada para que se possa oferecer as garantias de qualidade de serviço a cada agregado. Para que tal provisionamento se mantenha adequado utilizam-se mecanismos de controle de tráfego nas bordas dos domínios, onde o volume de tráfego é menor. Isso diminuí a dificuldade e a complexidade no atendimento de uma larga escala de fluxo.

Tais mecanismos incluem classificadores, medidores, e policiadores/condicionadores que compõem uma etapa do processo de provisionamento de qualidade de serviço chamada de condicionamento de tráfego.

3.2.1 CONDICIONAMENTO DE TRÁFEGO

As atividades referentes ao policiamento dos pacotes nos roteadores de borda para averiguar sua adequação ao perfil de tráfego contratado, são coletivamente chamadas de condicionamento de tráfego. O condicionamento envolve a classificação dos pacotes, medição do tráfego e uma subseqüente ação, dependendo da aderência dos pacotes ao perfil de tráfego contratado.

O Classificador é o primeiro elemento envolvido no processo de condicionamento de tráfego. Ele seleciona os pacotes recebidos nas interfaces de entrada baseado no conteúdo de alguma parte do seu cabeçalho. Foram definidos dois tipos principais de classificadores. O Classificador BA (*Behavior Aggregate*) classifica os pacotes baseado somente no conteúdo do campo DSCP. Esse caso ocorre quando o domínio anterior é compatível com *DiffServ* e os pacotes já vêm marcados. Quando o domínio anterior não é habilitado para enviar os pacotes com o campo DSCP previamente marcado, o classificador pode avaliar vários campos dos pacotes. Nesse caso, ele é chamado de Classificador MF (*Multi-Field)*. Em ambos os casos, o

resultado da classificação é o enquadramento do pacote em um BA válido no domínio e o seu encaminhamento para um processamento posterior. Por exemplo, se o classificador detectar que o pacote está usando o serviço de melhor esforço, ele provavelmente será encaminhado sem nenhum processamento adicional. Caso o pacote pertença a um BA para o qual foi definido um perfil de tráfego, ele é geralmente encaminhado para a fase de medição.

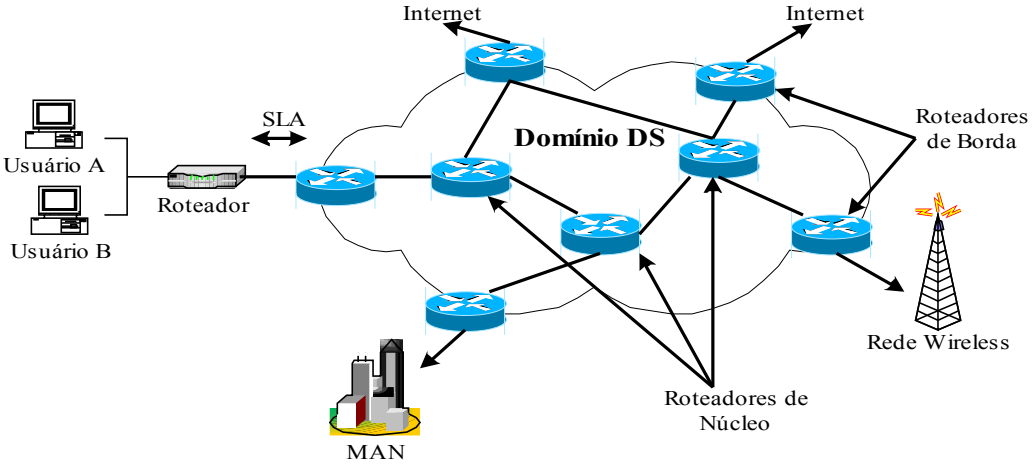

FIG. 3.1 Domínio DiffServ com roteadores de núcleo e de borda

O Medidor de tráfego mede as propriedades temporais de um fluxo de pacotes selecionados pelo classificador de acordo com o perfil de tráfego especificado. Embora vários mecanismos de medição possam ser utilizados, o mais apropriado é o balde de fichas (*token bucket*) (BLAKE et al., 1998). Este mecanismo é definido por uma taxa de dados *r* e uma rajada *b*. Uma analogia seria imaginar um balde com uma determinada capacidade máxima que contém fichas que são inseridas regularmente. Uma ficha corresponde à permissão para transmitir uma quantidade de bits, que pode ser apenas um. Quando chega um pacote, o seu tamanho é comparado com a quantidade de fichas em um balde. Se existir uma quantidade suficiente de fichas, o pacote é enviado. Caso contrário, ele é inserido em uma fila para que aguarde até haver fichas suficientes.

Feito isso, o medidor descobre quais pacotes estão dentro ou fora do perfil. Pacotes fora do perfil são aqueles que chegam quando não existem fichas suficientes no balde. Em termos mais genéricos, os conceitos de “dentro” e “fora” do perfil podem ser estendidos para

múltiplos níveis de conformidade. Por exemplo, é possível definir um medidor com várias taxas e rajadas e então comparar os pacotes com esses valores para obter uma avaliação mais precisa das características do tráfego.

Com base no resultado da medição, os pacotes são encaminhados para um estágio onde uma ação de condicionamento é realizada. A ação depende do serviço oferecido, mas, em geral, pode ser suavização, descarte, marcação ou contabilização para posterior cobrança. Geralmente não é aplicada nenhuma ação em pacotes dentro do perfil, a não ser no caso do Classificador MF, após o qual os pacotes têm que ser marcados para um DSCP específico.

O Suavizador é um elemento que introduz um atraso nos pacotes fora do perfil até que o balde tenha fichas suficientes para encaminhá-lo. Ou seja, ele molda o tráfego para que forçosamente fique dentro do perfil contratado. Um suavizador geralmente implementa uma fila com tamanho limitado. Quando a fila está cheia, os pacotes subseqüentes são descartados.

O Descartador descarta pacotes que foram considerados fora do perfil pelo medidor, para que o fluxo seja considerado dentro do perfil. Esse processo é também chamado de policiamento do tráfego. Um descartador pode ser implementado como um caso especial de um suavizador, no qual o tamanho da fila é zero.

O Marcador é um componente necessário no caso de os pacotes serem classificados por um Classificador MF, para que os roteadores de núcleo posteriores possam classificar os pacotes com um Classificador BA. Ele é utilizado também quando os dois domínios utilizam valores do DSCP diferentes para o mesmo BA. Outra função dele é rebaixar de classe pacotes fora do perfil.

Um Monitor é um elemento que pode ser utilizado para contabilizar a utilização para cobrança em caso de tráfego fora do perfil. O SLA pode prever que todo tráfego fora do perfil será encaminhado, mas haverá uma taxação extra. Esse componente, no entanto, não é padronizado pela arquitetura DiffServ.

Em um domínio DiffServ, em geral o condicionamento de tráfego é necessário somente nos roteadores de borda. Como estes roteadores geralmente têm um volume de tráfego menor

que os roteadores de núcleo, o impacto no desempenho não é significativo. Roteadores de núcleo, que recebem maior volume de tráfego, dedicam a maior parte dos seus recursos na atividade de encaminhar pacotes, de acordo com o seu PHB. A combinação da aplicação do PHB no centro da rede com o condicionamento de tráfego na borda, permite a criação de vários serviços em uma rede DiffServ.

3.2.2 BAs e PHBs

Um BA (*Behavior Aggregate*) é definido formalmente como "uma coleção de pacotes com o mesmo DSCP que estão cruzando um enlace em uma determinada direção em um Domínio DiffServ" (BLAKE et al, 1998). A quantidade de pacotes que pertencem a um determinado BA pode aumentar ou diminuir nos vários roteadores, à medida que fluxos iniciam e terminam neles. Em uma analogia com o sistema de tráfego rodoviário, o conjunto de carros que trafega de um ponto a outro se altera conforme surgem novos cruzamentos onde vários carros entram e saem do caminho principal. A definição de BA é diferente da definição de serviço. BAs são blocos de construção técnicos para construir serviços fim a fim, que incluem regras, PHBs e configurações específicas. Os usuários vêem somente os serviços, não os BAs.

A implementação de um BA requer que todos os pacotes recebam o mesmo tratamento de encaminhamento (PHB) nos roteadores por onde passam. Um PHB (*Per-Hop Behavior*) é uma descrição de um comportamento de encaminhamento observável externamente de um roteador *DiffServ*, aplicado a um determinado BA. O PHB é a maneira como um roteador aloca recursos para os BAs, e é em cima desse mecanismo local que serviços são construídos. A maneira mais simples de implementar um PHB é destinar a ele um determinado percentual de utilização da largura de banda de um enlace de saída.

Atualmente a arquitetura *DiffServ* oferece dois PHB's padronizados pelo IETF: O Encaminhamento Expresso (*Expedited Forwarding* - EF) (BLAKE et al., 1998), e o Encaminhamento Assegurado (*Assured Forwarding* – AF) (BLAKE et al., 1998). Os tráfegos

não classificados em nenhuma das classes pertencentes a esses PHBs são chamados de Melhor Esforço (BE – *Best Effort*), isto é, tráfego sem nenhuma garantia de QoS.

3.2.3 ENCAMINHAMENTO ASSEGURADO

O PHB AF procura garantir vazão de dados, permitindo a ocorrência de rajada de pacotes, desde que estes sejam de curta duração. Dessa forma, os congestionamentos de curta duração são permitidos nas filas dos nós AF. Entretanto, os congestionamentos longos são penalizados. Com isso, o serviço de Encaminhamento Assegurado não oferece garantias explícitas de retardo e variação do retardo, mas oferece garantia de banda passante, mesmo na ocorrência de congestionamentos. Ele é o serviço adequado para aplicações que exigem largura de banda sem restrições temporais como, por exemplo, a transferência de arquivos e a navegação web. O principal mecanismo dessa classe é o gerenciamento ativo de filas como RED (*Random Early Detection*) que descarta pacotes aleatoriamente, antes de ocorrer congestionamento. Este mecanismo é muito útil para tráfegos FTP e http, pois o controle de congestionamento do TCP reduz a taxa de transmissão quando a conexão perde um pacote, maximizando o aproveitamento do canal.

O serviço de Encaminhamento Assegurado possui quatro classes de serviço cada uma com três prioridades de descarte (alta, média, baixa) e cada uma dessas classes possui um *code point* próprio, resultando assim em doze *code points* diferentes compondo o serviço. A motivação para os PHBs AF é a demanda existente na *Internet* atualmente por encaminhamento assegurado de pacotes IP. Em uma aplicação típica, uma companhia usa a Internet para interconectar seus escritórios que estão geograficamente distribuídos e quer uma garantia de que os pacotes dentro dessa Intranet são encaminhados com alta probabilidade, se o tráfego agregado de cada escritório não excede o perfil contratado. Outro exemplo pode ser o de uma empresa que realiza vendas na Internet e que deseja que os pacotes de todos os usuários que a acessarem sejam encaminhados preferencialmente em relação ao tráfego de melhor esforço. O AF pode ser usado para implementar um serviço mais ou menos semelhante ao serviço de carga controlada do *IntServ*, embora com uma maior flexibilidade.

3.2.4 ENCAMINHAMENTO EXPRESSO

Através da utilização do PHB EF, pode ser obtido o serviço de "linha privativa virtual", que oferece garantias de baixo atraso, variação de retardo baixa e pequena taxa de perdas, oferecendo uma banda passante assegurada ao agregado. Para realizar isso, é necessária a garantia de que a taxa de serviço nos nós do domínio seja superior à taxa de chegada de dados. O tráfego EF deve receber esta taxa de saída independentemente da intensidade de qualquer outro tipo tráfego em um determinado roteador. Dessa forma, as filas nos roteadores estarão sempre minimizadas. Para tanto, esse serviço emprega condicionadores de tráfego nos nós de ingresso, de forma a garantir a correta relação entre as taxas. Assim, o EF pode ser utilizado para dar suporte a aplicações de tempo real, como por exemplo, videoconferência e telefonia IP.

3.3 A ARQUITETURA MPLS

Ainda que inicialmente não tenha sido concebido como um mecanismo de qualidade de serviço, MPLS (ROSEN et al, 2001) pode ser uma ferramenta importante para provedores de serviços, podendo fornecer habilidades por salto em muitas das maneiras que os modelos *IntServ* e *DiffServ* propõem.

A arquitetura MPLS (*Multiprotocol Label Switching Architecture*) é uma tecnologia de comutação de rótulos (*labels*) que tem o objetivo de integrar a flexibilidade e a funcionalidade da camada de rede (utilizando informações de roteamento) em conjunto com a capacidade e agilidade da camada de enlace (empregando comutadores de alta velocidade). A arquitetura MPLS está conceitualmente localizada entre a camada de enlace e a camada de rede segundo o modelo OSI *(Open Systems Interconnection*). A idéia básica do MPLS é adicionar um rótulo nos pacotes que entram em redes MPLS. Os roteadores de comutação de rótulos (*Label-Switching Routers – LSRs)* (ROSEN et al, 2001) fazem a classificação e o encaminhamento, além de prover os serviços para os pacotes baseados no valor do rótulo.

O roteador de ingresso LSR de um domínio MPLS classifica e encaminha os pacotes de chegada baseado na informação do cabeçalho do pacote IP e na informação armazenada na tabela de roteamento. Um rótulo é adicionado ao pacote. Dentro do domínio MPLS, este rótulo é usado como índice para a tabela de encaminhamento do roteador LSR. O rótulo de entrada é substituído por um rótulo de saída e o pacote é enviado para o próximo roteador LSR. O rótulo é removido antes do pacote sair do domínio MPLS. O caminho seguido pelos pacotes entre os roteadores de ingresso e egresso são chamados de caminhos comutados por rótulos (*Label Switched Paths* –LSPs). Os caminhos LSPs são configurados usando um protocolo de sinalização, como o protocolo RSVP ou pelo protocolo LDP (*Label Distribution Protocol*) ((ROSEN et al, 2001).

3.3.1 APLICAÇÕES EM REDES DA ARQUITETURA MPLS

Uma vantagem direta da utilização de MPLS é substituir o encaminhamento tradicional IP pelo encaminhamento baseado em rótulos, que é um processo consideravelmente mais rápido. Entretanto, somente isso não seria um motivo suficiente para a adoção de MPLS, uma vez que a tecnologia de roteadores de alta velocidade atual já permite fazer buscas rápidas na tabela de roteamento.

Uma aplicação para MPLS é a emulação redes orientadas a conexão, como Frame Relay ou ATM. Uma vez que MPLS é um mecanismo orientado a conexão, ele pode ser utilizado para emular qualquer serviço orientado a conexão não confiável com um LSP. Isso permite que uma rede integrada baseada em datagramas ofereça serviços legados aos seus usuários usando uma única infra-estrutura.

MLPS pode ser utilizado também para facilitar a integração de IP com ATM. A interligação de roteadores através de uma sub-rede ATM requer a configuração de vários circuitos virtuais, no pior caso N (N-1)/2 circuitos virtuais, onde N é o número de roteadores. Usando roteadores MPLS e fazendo uma atualização dos *switches* ATM, se torna possível fazer uma fácil integração entre ambas as tecnologias. *Switches* ATM podem se comportar como LSRs, fazendo o encaminhamento baseado em rótulos.

MPLS também permite facilmente a configuração de Redes Privadas Virtuais (VPNs). Empresas usam VPNs para criar túneis seguros entre matrizes e filiais ou entre parceiros comerciais. Normalmente esse tipo de comunicação utiliza linhas privativas, porque a Internet é considerada insegura para transportar informações confidenciais. Com MPLS pode-se criar uma VPN usando o serviço de emulação de Frame Relay, ou então configurando roteadores MPLS nas redes dos usuários para serem o início e o final de LSPs. Dentro dos LSPs, a comunicação apresenta um alto nível de segurança.

3.4 QUALIDADE DE SERVIÇO EM AMBIENTES VIRTUAIS COLABORATIVOS

As redes utilizadas pelos Ambientes Virtuais Colaborativos carregam uma grande parcela de seus dados na forma de tráfego de tempo real e interativo, os quais esgotam a capacidade e os recursos da rede. Dessa forma, o gerenciamento de tráfego em tais redes, é uma preocupação importante e envolve a gerência da alocação de recursos para transferência de dados para garantir a contínua manutenção dos níveis de qualidade de serviço requeridos. Se não gerenciada eficientemente, a rede pode facilmente se tornar um gargalo para a aplicação. Quanto mais usuários participam do AVC, maior é a quantidade agregada de informação gerada pela aplicação. Entretanto, a capacidade da rede é um recurso limitado, de forma que o sistema deve ser cuidadosamente planejado de forma a determinar como alocar a capacidade da rede para os vários tipos de informação que os diversos usuários em um AVC devem trocar. Como exemplo, podemos citar que quando um usuário se conecta a um AVC através de uma linha telefônica e um modem, os quais lhe dão recursos de rede limitados, é desejável que seu fluxo de dados seja tratado de forma diferente na rede, de forma a garantir a consistência no AVC.

Em AVCs é desejável tratar pacotes diferentes de formas diferentes na rede. Por exemplo, atualizações de estado relacionadas a novos usuários se unindo ao mundo virtual são mais importantes e urgentes que atualizações de eventos e podem se relacionadas a classes de tráfego com uma prioridade maior. Níveis de serviço diferentes devem ser providos para fluxos de tráfego individuais e para alguns objetos se necessário.

Devido à necessidade de garantias específicas em termos de largura de banda e atraso para que os AVCs executem adequadamente, várias técnicas de qualidade de serviço foram desenvolvidas para funcionarem integradas com as arquiteturas de comunicação dos AVCs em grande escala.

Em (ERASLAN et al., 2003) é proposta uma arquitetura de comunicação e gerenciamento para ambientes virtuais colaborativos. Para validar esta arquitetura, foi construído um protótipo que a usava sobre o protocolo IPv6, o qual foi batizado de VESIR-6 (*Virtual Environment Supporting Multi-user Interaction over IPv6*).

A arquitetura proposta faz distinção entre mensagens que são significantes e aquelas que são meramente contingentes e com isso, são definidos dois níveis de serviço na arquitetura.

O protótipo VESIR-6 basicamente gera dois tipos de mensagens na rede: atualizações de eventos e os menos freqüentes comandos de controle. As atualizações de eventos em AVCs são freqüentes e se algumas são perdidas ou chegam fora de ordem, as próximas atualizações irão corrigir estes erros. Atualizações de eventos ocorrem quando o usuário está se movendo dentro do mundo virtual. Enquanto este usuário vai se movendo, seu sistema envia uma série de mensagens de posição. Se uma dessas mensagens de posição é perdida, o sistema pode pular para as mensagens subseqüentes e ignorar a mensagem perdida. Em contrapartida, os comandos de controle envolvem mudanças de estado e são da mais alta importância. Com isso, os tráfegos de atualização de eventos e comandos de controle são separados em diferentes fluxos de tráfego, e é atribuída uma alta prioridade para o fluxo associado aos comandos de controle.

Os pacotes são então classificados em fluxos pelo protocolo e pelos campos com os DSCPs no cabeçalho do pacote. Cada fluxo corresponde a uma fila de saída separada. Quando um pacote é associado a um fluxo, ele é colocado na fila para aquele fluxo. Durante períodos de congestionamento, os nós IPv6 alocam uma porção de largura de banda disponível para cada fila ativa.

Dessa forma, a qualidade de serviço fim-a-fim é conseguida através da classificação, marcação, escalonamento e possível ajuste dos pacotes enviados através da rede pela aplicação VESIR-6. Uma variedade de enfileiramento, ajuste de tráfego e tecnologias de filtragem são necessárias para implementação de prioridade de tráfego e controle de congestionamento fim-a-fim através da rede.

A plataforma proposta provê qualidade de serviço dinâmica para a aplicação através da divisão do PHB *Expedited Forwarding* (EF) em duas variações. É usado um escalonador para desmembrar atualizações de eventos em diferentes fluxos. Para estes fluxos, era atribuído um conjunto de prioridades de forma a garantir uma degradação aceitável do serviço em caso de congestionamento da rede. Neste caso, os pacotes de baixa prioridade pertencentes à classe de serviço EF1 seriam descartados primeiro. Já para os pacotes da classe EF2 seria garantida uma qualidade de serviço maior.

Uma vez que os pacotes fossem classificados, o próximo passo seria marcá-los com uma identificação única para garantir que essa classificação fosse garantida fim-a-fim. Isso foi feito através do campo de classe de tráfego IPv6 no cabeçalho de um datagrama IPv6.

O propósito deste tipo de marcação de pacotes é garantir que etapas como escalonamento e enfileiramento concordem no tratamento correto dos pacotes usando a marcação.

Uma vez que o tráfego esteja classificado, o próximo passo é garantir que ele receba tratamento especial nos nós através do escalonamento e enfileiramento. As disciplinas de escalonamento de pacotes arbitram acesso à largura de banda e provêem garantia de desempenho para aplicações. O escalonador de pacotes determina a ordem na qual cada pacote é transmitido. O algoritmo de escalonamento mais simples consiste em se ordenar os pacotes como uma função de sua prioridade. Desta forma, pacotes com alta prioridade são transmitidos primeiro. Este método de classificação pode gerar um período de espera indefinido para pacotes com baixa prioridade se o tráfego de dados de alta prioridade é muito pesado. Para evitar este tipo de problema, usualmente, múltiplas filas são usadas, com uma taxa mínima de serviço associada a cada fila. Desta forma, ao testar o protótipo, foram implementadas os esquemas de gerenciamento de filas *Priority First In First Out* e *Class Based Queuing* (CBQ) (JACOBSON et al., 1999).

O escalonador de pacotes é ciente do DSCP, ou seja, ele é apto a detectar pacotes de alta prioridade marcados com precedência pela aplicação e escaloná-los mais rápido, providenciando tempo de resposta superior para este tráfego. À medida que o valor de prioridade aumenta, o algoritmo aloca mais largura de banda para aquele fluxo para se certificar que ele seja servido mais rapidamente quando acontecer um congestionamento.

Uma outra abordagem, presente na arquitetura MASSIVE (GREENHALGH; BENFORD, 1997), descrita no capítulo anterior, usa técnicas diferentes para prover qualidade de serviço. Nesta arquitetura, existem duas entradas primárias; a primeira é a matriz de conhecimento de um conjunto de usuários e objetos no mundo virtual (ou algum subgrupo se o mundo estiver particionado em regiões). Estas são as interpretações das declarações de preferências dos usuários por recursos do mundo. Elas mudam dinamicamente conforme o usuário se movimenta e ajustam os parâmetros do modelo espacial (mudando o tamanho da aura e o nível de conhecimento). A segunda entrada é a descrição de toda a informação potencialmente disponível para transferência que está associada com usuários e objetos. Com isso, esta arquitetura inspeciona os valores de conhecimento e determina a alocação ótima das mídias disponíveis (áudio e vídeo) entre os usuários. Fazendo isso, ela tenta manter o equilíbrio entre as preferências individuais e a alocação de recursos ótima para o grupo.

4. IMPLEMENTAÇÃO DO MODELO NO SIMULADOR NS

Este capítulo descreve a implementação do protocolo ISTP (WATERS et al., 1997) integrado com técnicas de diferenciação de serviços. De forma a validar a proposta da dissertação de avaliar a utilização da diferenciação de serviços em uma plataforma para ambientes virtuais colaborativos em grande escala, foi escolhida a plataforma SPLINE. A razão desta escolha, foi a vasta utilização da plataforma no meio acadêmico (ANDERSON et al., 1996), por possuir uma menor complexidade em relação a outras plataformas similares e por ser uma representante das plataformas que usam filtragem de dados espacial.

4.1 SPLINE

A plataforma SPLINE (*Scalable Platform for Large Interactive Networked Environment*) (BARRUS et al. 1996), foi desenvolvida no *Mitsubishi Electric Research Labs* (MERL). O primeiro mundo virtual construído utilizando esta plataforma foi um parque (*Diamond Park*). Vários usuários podem interagir com o parque pedalando em bicicletas controladas por computador. Quando próximos uns dos outros, os usuários são capazes de conversar. Na plataforma SPLINE, o mundo virtual é modelado por uma base de dados que contém informações sobre este mundo. Este modelo está parcialmente distribuído entre os processos SPLINE. As aplicações interagem umas com as outras através da modificação desta base de dados e da observação das modificações realizadas pelos outros usuários.

O mundo é dividido em *locales*, áreas que podem ser processadas separadamente. Cada *locale* está associado a um grupo *multicast*, possui um sistema de coordenadas próprio e seu tamanho, formato e orientação são escolhidos arbitrariamente. Cada participante possui uma cópia parcial da base de dados, de acordo com o seu *locale.* Servidores de *locale* são responsáveis por manter um registro do estado de todos os objetos em um determinado *locale*. Desta forma, um processo SPLINE que se interesse por um *locale* pode receber a informação atualizada sem contatar cada processo pertencente ao *locale*.

A comunicação entre os processos SPLINE na rede, é feita através do protocolo ISTP (*Interactive Sharing Transfer Protocol*) (WATERS et al., 1997). Ele não é particularmente ligado a SPLINE e vice-versa, ou seja, o protocolo ISTP pode ser usado por outra arquitetura de ambiente virtual. Além disso, a SPLINE poderia usar um outro conjunto de protocolos para atingir sua meta, ou seja, compartilhamento do mundo virtual. Apesar disso, ISTP possui várias características que o tornam adequado para a SPLINE, como o suporte a *locales*, por exemplo.

O objetivo do protocolo ISTP, é transportar informações sobre objetos em um modelo de mundo compartilhado sob um AVC. Tal objetivo, inclui características para gerenciar os vários objetos presentes em um Mundo Virtual. Os vários objetos que a SPLINE deve suportar carregam características bem diferentes, indo de objetos muito extensos e estáticos, como uma imagem de fundo, até objetos em tempo real, como áudio. Para conseguir atender tal gama de características, o protocolo ISTP possui cinco sub-protocolos:

- sub-protocolo de conexão ponto-a-ponto: usado para estabilizar e manter uma conexão TCP entre dois processos ISTP;
- sub-protocolo de transmissão de estado do objeto: usado para comunicar o estado de objetos de um processo ISTP para outro. Tais atualizações podem ser enviadas via conexão ponto-a-ponto ou pelo protocolo UDP Multicast;
- sub-protocolo de áudio em cadeia: usado para enviar áudio via o protocolo RTP(*Real Transfer Protocol*);
- sub-protocolo de comunicação baseada em *Locales*: é o núcleo do ISTP e suporta o compartilhamento de informações sobre objetos no modelo de mundo e;
- sub-protocolo de comunicação baseada no conteúdo: suporta comunicação de informação sobre *beacons* usando o modelo de servidor central.

Os últimos dois sub-protocolos são construídos sobre os outros três. Uma característica chave do protocolo ISTP é que ele é totalmente distribuído, ou seja, um processo ISTP tanto pode ser o cliente ou o servidor para outros processos.

O protocolo ISTP é um protocolo híbrido que é construído sobre quatro outros protocolos: TCP, UDP, RTP e HTTP.

- O protocolo TCP é usado para a comunicação confiável de informação de controle;
- os protocolos UDP e RTP são usados para a comunicação de informações críticas (mensagens UDP e RTP são ambas enviadas via *multicast* sempre que possível);
- o protocolo HTTP é usado para a distribuição de grandes ou complexas quantidades de dados.

Através desta abordagem híbrida, o protocolo ISTP atende as necessidades de comunicação para diferentes tamanhos e tipos de objetos de uma maneira bastante eficiente e provê a confiabilidade e correção de erro necessária para operações através dos vários tipos de configurações de rede.

Os processos ISTP guardam uma cópia dos dados que compartilham. Tais cópias chamadas de Modelo de Mundo (como descrito anteriormente) são limitadas pela noção de *locale*, ou seja, cada objeto apenas lida com objetos dentro do *locale* onde ele está assim como *locales* na sua vizinhança. Esta característica permite escalabilidade, enquanto reduz o poder de processamento requerido em cada estação.

4.1.1 SUB-PROTOCOLO DE CONEXÃO PONTO-A-PONTO

O sub-protocolo de conexão ponto-a-ponto é implementado através de uma conexão TCP. O processo que faz a requisição do canal de comunicação abre uma conexão TCP para outro processo ISTP e envia uma mensagem HTTP GET pedindo por uma conexão. Se a conexão é aceita, então uma resposta *ISTP Connection Status* será enviada e a conexão será então aberta. No caso da conexão ser negada, então a resposta é uma mensagem HTTP *Not Found.* Em qualquer momento, qualquer processo pode fechar esta conexão, e mensagens *keep-alive* são enviadas de uma forma regular para que os processos sejam informados de que ainda estão funcionando. A qualquer momento, um dos processos pode reabrir a comunicação. Neste caso, a outra parte se comporta como se a conexão tivesse acabado de ser criada.

4.1.2 SUB-PROTOCOLO DE TRANSMISSÃO DE ESTADO DE OBJETO

As mensagens *Object State* são usadas para comunicar o estado de objetos de um processo ISTP para todos os processos que compartilham o mesmo *Locale*. Quando uma mensagem *Object State* é recebida por outro processo, ela é usada para criar, atualizar ou remover a cópia do objeto a qual ela se refere.

Cada objeto no ISTP é identificado por um identificador global único, o qual inclui um endereço IP para evitar qualquer tipo de dependência em relação ao servidor.

4.1.3 SUB-PROTOCOLO DE ÁUDIO EM CADEIA

Este sub-protocolo é baseado diretamente no protocolo RTP para transmissão de comunicação via áudio. A comunicação via áudio na SPLINE, é tratada como qualquer outro objeto. Uma fonte de som é colocada em uma certa locação do mundo virtual e uma URL para um arquivo de som é então fornecida. Um usuário recebe tal informação com a qualidade de acordo com a sua proximidade da fonte de som.

Existem três tipos de fontes de áudio possíveis no mundo virtual: sons contínuos que geralmente são músicas de fundo do ambiente; sons especiais de eventos, como por exemplo o som resultante da colisão de dois carros no mundo virtual e por último som em tempo real, geralmente proveniente da comunicação entre dois usuários.

4.1.4 SUB-PROTOCOLO DE COMUNICAÇÃO BASEADA EM LOCALES

Cada *locale* possui um servidor de *locale* associado a ele, o qual pode gerenciar mais de um *locale*. Este servidor é responsável por gerenciar a entrada e saída de usuários da partição e ajudar usuários vítimas de uma perda de conexão a se atualizarem em relação à imagem mais atual do Mundo Virtual. Cada *locale* possui dois endereços *multicast:* um para a comunicação de mudanças no estado dos objetos e outro usado para a comunicação de áudio

em cadeia. Ainda que a comunicação *multicast* seja o padrão, quando ela não é possível, ISTP usa *multicast* simulado através de conexões ponto-a-ponto. Dessa forma, usuários que tenham acesso à comunicação *multicast* e usuários que fazem parte de redes que não suportem esse tipo de comunicação poderão participar do Mundo Virtual da mesma forma.

4.1.5 SUB-PROTOCOLO DE COMUNICAÇÃO BASEADA NO CONTEÚDO

Esse sub-protocolo suporta conexões de conteúdo endereçado entre processos ISTP e controla o armazenamento e publicação de informação sobre *beacons* como *tags* URL. Após criar um objeto, um processo pode publicá-lo pela criação de um objeto *beacon* e registrando-o em algum servidor baseado em conteúdo. Neste ponto, o *beacon* é visível por qualquer outro processo o qual pode acessar tal informação a qualquer momento. Sempre que um processo cria ou modifica um *beacon,* ele abre uma conexão ponto-a-ponto para o servidor especificado pela *tag* do *beacon* e executa qualquer atualização que se faça necessária.

4.2 INFRA-ESTRUTURA DA SIMULAÇÃO

Como uma forma de simplificar a etapa de implementação da simulação sem perder o caráter de generalidade, propôs-se adotar um modelo reduzido dos sub-protocolos e mensagens do protocolo ISTP, de forma que fosse possível gerar um tráfego de mensagens de controle e de atualização.

A ferramenta escolhida para simular a plataforma foi o NS (*Network Simulator*) (FALL; VARADHAN, 1997). Este trabalho foi desenvolvido em um computador AMD Athlon XP 2800 com 512 MB de memória RAM tendo como sistema operacional, o sistema Linux.

A fim de avaliar o desempenho da plataforma foram realizadas simulações em uma topologia similar a estrutura encontrada na *Internet*. O comportamento da plataforma foi avaliado visando verificar o impacto de aplicações com requisitos diferentes. Estas

simulações compreenderam as operações críticas - troca de grupos, entrada e saída do mundo, e as atualizações de estado.

Para realizar as operações críticas referentes à simulação foi necessária a implementação de três sub-protocolos. Os sub-protocolos implementados foram:

- sub-protocolo de conexão ponto-a-ponto;
- sub-protocolo de Transmissão de Estado de Objetos;
- sub-protocolo de Comunicação Baseada em *Locales*.

Com a implementação dos três sub-protocolos foi possível gerar grande parte do tráfego de mensagens necessário para gerar resultados na simulação que possam ser posteriormente analisados.

As mensagens implementadas para permitir o funcionamento dos três sub-protocolos foram:

- *Object State* – contém a descrição de um ou mais objetos que um processo possui ou as mudanças ocorridas em seu estado desde o ultimo envio de uma mensagem *Object State* por aquele processo. São mensagens enviadas via UDP para o grupo *multicast* relacionado a um determinado *locale* quando algum processo ISTP quer atualizar a informação referente a algum objeto presente no *locale.* ou via conexão ponto-a-ponto, quando o servidor de *locales* deseja atualizar algum processo recém-chegado ao *locale* em relação à visão que os outros processos têm daquele *locale*.
- *Object State Summary* – mensagens usadas para que um processo servidor possa comunicar a um processo cliente informações sobre todos os objetos que este processo conhece ou para que um processo cliente possa comunicar a um servidor todos os objetos que ele possui.
- *Locale Com Status* – são mensagens usadas para regular a comunicação baseada em *Locales.* O processo ISTP envia mensagens *Locale Com Status* para requisitar início ou mudanças na comunicação baseada em *locales* e para especificar terminação. O servidor de *locales* envia mensagens *Locale Com Status* para especificar início, mudanças ou terminação.

- Na FIG. 4.1 é mostrada a aplicação e as mensagens implementadas no simulador.

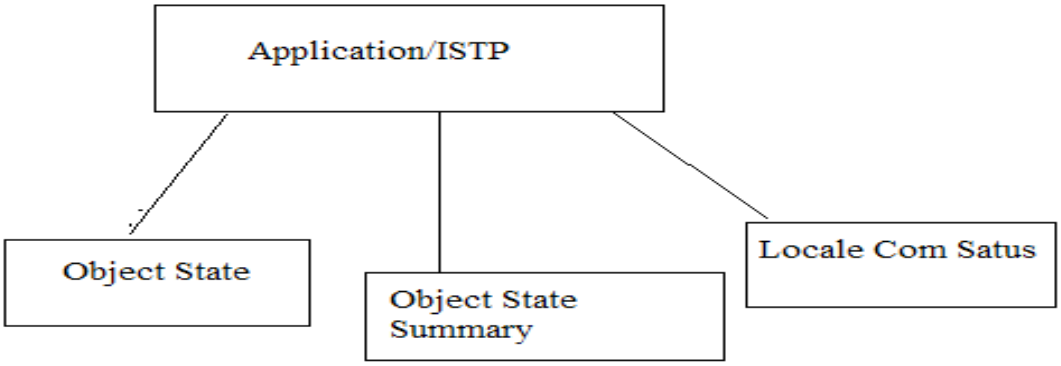

FIG. 4.1 Aplicação ISTP e mensagens

Na implementação deste modelo simplificado do protocolo ISTP no simulador NS sem o uso de técnicas de qualidade de serviço, as mensagens de controle (*Locale Com Status, Object State Summary*) são enviadas em conexões ponto-a-ponto TCP sem alterações. Já na implementação usando diferenciação de serviços todas as mensagens serão enviadas via o protocolo UDP e são usadas classes de serviços diferenciadas para prover garantias específicas para cada tipo de mensagem. No caso das mensagens de controle, elas recebem uma prioridade de descarte de pacotes menor em situações de congestionamento em relação à outra classe de mensagens, permitindo com isso, que elas tenham uma maior largura de banda e um menor atraso. O objetivo era reduzir o atraso referente ao estabelecimento da conexão TCP entre os processos e mesmo assim garantir a entrega das mensagens críticas da aplicação.

4.2.1 COMUNICAÇÃO ENTRE PROCESSOS

1) Após o inicio da conexão, uma mensagem *Locale Com Status* requisitando que a comunicação seja iniciada deve ser enviada pelo processo requisitante para o servidor de *Locales*;

2) Quando uma mensagem *Locale Com Status* for recebida pelo processo requisitante, como resposta à sua mensagem anterior, deve ser aberta uma conexão para o endereço *multicast* especificado na mensagem e que foi informado pelo servidor de *Locales* e deve-se começar a receber ou enviar informação. Esta comunicação pode ser vista na FIG. 4.2.

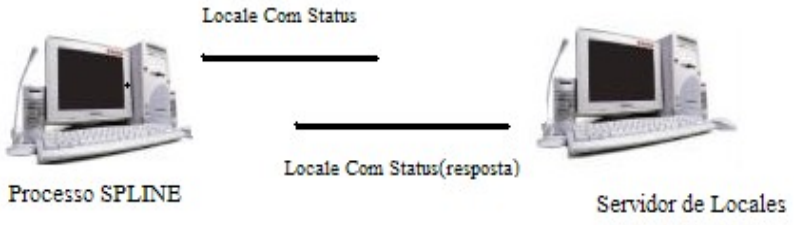

FIG. 4.2 Seqüência inicial de comunicação entre processos

3) A partir disso, mensagens *Object State* e *Object State Summary* serão enviadas pelo processo servidor, como todas as informações que ele possui sobre os objetos existentes naquele *Locale*;

4) Periodicamente (uma vez a cada 30-100 milisegundos), um processo envia uma ou mais mensagens *Object State* descrevendo todos os objetos que ele possui naquele *Locale* que mudaram desde a ultima vez que ele enviou mensagens. Tais mensagens são enviadas via *multicast*. Da mesma forma, este processo deverá processar as mensagens *Object State* recebidas de outros processos. A seqüência das mensagens pode ser visualizada na FIG. 4.3.

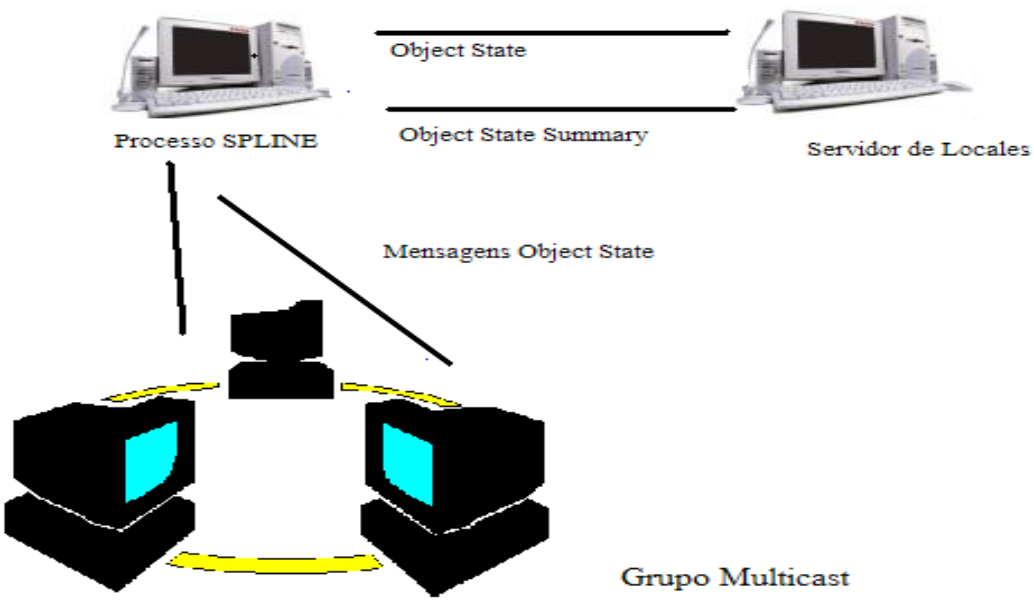

FIG. 4.3 Seqüência de interação entre processos

4.2.2 COMUNICAÇÃO ENTRE PROCESSOS USANDO DIFERENCIAÇÃO DE SERVIÇOS NA SIMULAÇÃO

Nesta abordagem não há transmissão de mensagens via TCP e as mensagens de controle recebem uma prioridade maior que as mensagens de atualização. Com isso, não haverá mensagens para abertura de conexões ponto-a-ponto. A comunicação entre os processos é composta das seguintes etapas:

1) Uma mensagem *Locale Com Status* será enviada via UDP pelo processo requisitante com uma alta prioridade requisitando que a comunicação deve ser iniciada.

2) Quando uma mensagem *Locale Com Status* enviada pelo servidor de *Locales* for recebida pelo processo, uma conexão para o endereço *multicast* especificado na

mensagem deve ser aberta e deve-se começar a receber ou enviar informação. Esta mensagem deverá possui uma alta prioridade.

3) Após isso, mensagens *Object State* e *Object State Summary* serão enviadas pelo servidor, com todas as informações que aquele servidor de *Locales* em particular possui sobre os objetos existentes no *Locale* a que serve. Estas mensagens, dada sua importância para a entrada do novo usuário no mundo virtual também são ser enviadas com alta prioridade.

4) De uma forma bastante freqüente (uma vez a cada 30 a 100 milisegundos, dependendo do tipo de aplicação com uma distribuição uniformemente distribuída), um processo envia uma ou mais mensagens *Object State* descrevendo todos os objetos que ele possui no *Locale* que mudaram desde a última vez que este processo enviou mensagens. Tais mensagens são enviadas via *multicast*. Da mesma forma, este processo devera processar as mensagens *Object State* recebidas de outros processos. Devido a grande freqüência destas mensagens, elas não são críticas como as mensagens anteriores. Sendo assim, estas mensagens receberão uma prioridade menor que as outras.

O objetivo do uso da Diferenciação de Serviços no lugar do protocolo TCP para transmissão das mensagens de controle visa uma diminuição no tempo para entrar e sair de um determinado *Locale*, com isso reduzindo o tempo de *handoff*[1] e conseqüentemente aumentando o desempenho da aplicação.

4.2.3 IMPLEMENTAÇÃO DO PROTOCOLO ISTP

[1] *Handoff* é o tempo decorrido desde o momento que um usuário sai de um escopo de interação, ou seja de um grupo *multicast* até o momento em que ele entra em outro grupo.

No simulador NS, os agentes, representados pela classe *Agent*, são abstrações que representam pontos finais, onde pacotes, da camada de rede, são construídos ou consumidos. A classe *Agent* oferece funções para o desenvolvimento de outros protocolos da camada de rede e transporte. Quando um usuário deseja criar um novo protocolo, esta nova classe será derivada da classe *Agent*.

Da mesma forma, a classe *Application,* engloba implementações de tráfegos de aplicações, que se apóiam em agentes de camada inferiores para transmitir dados. O subconjunto do protocolo ISTP foi implementado no NS como uma aplicação que apenas simula o comportamento do protocolo ISTP, e com isso as mensagens usadas para comunicação não carregam dados, e são representadas apenas por seu tamanho. A especificação da classe implementada no simulador pode ser vista na figura 4.6.

```
class ISTPApp : public Application {
 public:
            ISTPApp();
            void send_istp_atual();
            void send_istp_control();
            void send_istp_control2();
};
```

FIG. 4.4 Especificação da classe criada para a simulação

Nesta especificação, está sendo declarada a classe ISTPApp que herda os atributos da classe *Application* em conjunto com a declaração dos métodos para envio das mensagens de atualização e de controle na simulação.

O protocolo ISTP foi implementado de duas formas no simulador NS: sem o uso de técnicas de diferenciação de serviços e com o uso de técnicas de diferenciação de serviços. Na figura 4.7 pode ser vista a classe implementada derivada da classe *Application.*

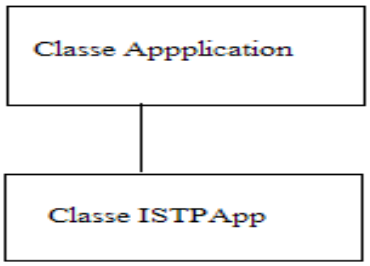

FIG. 4.5 Diagrama com a classe criada para a simulação

Na simulação, as mensagens têm um tamanho definido como um parâmetro que será uma estimativa do tamanho em bytes de cada mensagem, respeitando a definição do tamanho de cada campo apresentada em (WATERS et al., 1997), entretanto, para fins de simplificação, os campos de cada mensagem não foram implementados. O tamanho de cada mensagem é mostrado na tabela em TAB. 4.1

TAB 4.1 Tabela com o tamanho em bytes das mensagens da simulação

Mensagem	Tamanho(bytes)
Locale Com Status	126
Object State	100
Object State Summary	1000

Os pacotes gerados por essa aplicação carregam o endereço destino (nesse caso, um número que identifica o agente destino no NS), o endereço fonte e o tamanho dos dados. Os agentes que representam os protocolos de transporte no NS apenas simulam o envio de um pacote de um determinado tamanho. Na simulação sem diferenciação de serviços, são usados os protocolo TCP para envio de mensagens de controle e o protocolo UDP para envio de mensagens de atualização. Na simulação usando diferenciação de serviços apenas o protocolo UDP é usado para o envio das mensagens. O NS possui ambos os protocolos já implementados. Além disso, ele possui vários agentes que implementam variações do protocolo TCP, modelando diferentes tipos de controle de congestionamento. No entanto, apenas o agente *FullTCP* é bidirecional, ou seja, é capaz de receber ou enviar dados. Assim este agente se aproxima do comportamento do TCP como é usado na *Internet* e por isso foi escolhido para ser usado na simulação. Vale dizer também, que como o NS já possui todos os protocolos de transporte e várias técnicas de diferenciação de serviços já implementadas, sendo que a ligação destes protocolos com a aplicação é feita a nível de *script.* Na figura 4.8 pode ser visto uma parte de um script usado no trabalho, mostrando a aplicação implementada e sua ligação com os protocolos da camada de transporte. Neste *script* são criados dois grupos

multicast e dois agentes UDP que são conectados a dois objetos ISTP. Ao fim do trecho de *script*, é mostrada a ligação entre os agentes UDP aos agentes destino *Null*.

A fim de se realizar a simulação, foi necessária também a escolha de algoritmos de roteamento, tanto ponto-a-ponto quanto *multicast.* Como a infraestrutura oferecida pelo NS abrange modelos para os principais protocolos encontrados na Internet, pode-se escolher um ambiente mais próximo do real.

O roteamento dos datagramas IP foi realizado através do algoritmo SPF (*Shortest Path First*) (Huitema, 1995) de Dijkstra. Neste algoritmo as rotas são calculadas usando uma matriz de adjacência e custo dos enlaces para todos os enlaces da topologia. Nas simulações realizadas não foram simuladas mudanças na topologia, por isso, a estratégia adotada foi a estática, em que o algoritmo de computação de rotas é executado somente uma vez, no início do algoritmo. O protocolo de roteamento *multicast* do NS utilizado foi o modo denso estático(DM- *Dense Mode*) baseado no protocolo DVMRP(*Distance Vector Multicast Routing Protocol*)(Deering et al, 1988).

```
set mproto DM
set mrthandle [$ns mrtproto $mproto {}]
set group0 [Node allocaddr]
set group1 [Node allocaddr]

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$udp0 set dst_addr_ $group0
$udp0 set dst_port_ 0
$ns attach-agent $n0 $udp0

set istp0 [new Application/ISTP]
$istp0 attach-agent $udp0


set udp1 [new Agent/UDP]
$udp1 set dst_addr_ $group1
$udp1 set dst_port_ 0
$udp1 set class_ 2
$ns attach-agent $n1 $udp1

set istp1 [new Application/ISTP]
$istp1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

set null1 [new Agent/Null]
$ns attach-agent $n2 $null1

#Connect the traffic sources with the traffic sink
$ns connect $udp0 $null0
$ns connect $udp1 $null1
$ns connect $udp0 $null1
$ns connect $udp1 $null0
```

FIG. 4.6 Exemplo de script mostrando a aplicação implementada

5. SIMULAÇÕES E RESULTADOS

A simulação é um recurso bastante utilizado para a realização de estudo sobre determinadas idéias/conceitos, sem que seja necessário testar esta idéia em um meio real. Na área de redes de computadores, este recurso é extremamente utilizado, pois facilita o projeto e avaliação, estudo de protocolos novos e existentes. Avaliar um determinado protocolo, em uma rede real, pode ser bastante oneroso, como por exemplo, avaliar o desempenho de modificações na dinâmica de comunicação de centenas de nós entre sub-redes.

Para que fosse possível criar simulações que fossem executadas utilizando a aplicação implementada no trabalho, foram criados vários *scripts* que simulam o comportamento dos usuários participando da sessão simulada. Nestes *scripts* estão configurados os parâmetros definindo as características do ambiente virtual, em que nós da topologia estão os usuários, os processos que atuam como servidores, o comportamento dos usuários, o número de *locales* nos quais o mundo virtual será particionado, e no caso da implementação utilizando a diferenciação de serviços, para quais classes de tráfego, as diferentes mensagens foram mapeadas.

Os objetivos das simulações executadas são: avaliar a quantidade de mensagens perdidas nos dois modelos, tanto a perda total de mensagens quanto a perda das mensagens de controle da simulação e a quantidade média de mensagens necessária para a abertura de uma sessão no mundo virtual.

A partir da mudança nos valores dos parâmetros usados, foram simulados vários cenários, e seus resultados posteriormente analisados.

No cenário com o uso da diferenciação de serviços, foi usado o algoritmo RED (Braden et al., 1998).

5.1 O MECANISMO DE FILAS RED

O mecanismo de gerenciamento de filas RED, tem como idéia básica que não é necessário se esperar até que o *buffer* esteja cheio para que se detecte contenção de pacotes, ou seja, deve-se detectar congestionamentos antes que o buffer fique cheio. Os sinais de congestionamento podem ser materializados através de descarte de pacotes, mas também podem ser demonstrados através de uma marcação, sem que haja a real necessidade de descartá-los. Uma versão básica do algoritmo para roteadores é conhecida como *Drop Tail*. Filas *Drop Tail* simplesmente aceitam qualquer pacote que chegue quando existe espaço no buffer e descartam qualquer pacote que chegue quando existe espaço insuficiente no *buffer*.

Uma forma de usar o mecanismo RED na diferenciação de serviços, é através da variante RIO C(*Rio Coupled*), onde os pacotes podem ser marcados como dentro do perfil ou fora do perfil.

5.2 TOPOLOGIA USADA

As simulações realizadas priorizaram a análise da perda de pacotes em cenários diferentes. A primeira simulação visa analisar a perda de pacotes usando os dois modelos, com diferenciação de serviços e sem diferenciação de serviços.

A topologia criada é composta de uma árvore *multicast* cheia com 16 nós, sendo que o servidor de *Locales* é ligado à um roteador interno a rede e esse enlace é o gargalo da rede com uma largura de banda de 1Mbps. Os demais enlaces possuem uma largura de banda de 10 Mbps e atraso de 2 milisegundos. A topologia pode ser visualizada na figura 5.1. Tal topologia foi escolhida porque a árvore binária com enlace de gargalo representa uma

situação crítica, onde existem enlaces que concentram a maioria do tráfego em uma rede. Estes enlaces estão localizados próximos da fonte de um grupo *multicast,* o qual é um cenário passível de existência na *Internet.* Além disso, o enlace com gargalo também é considerado um ponto crítico com relação à perda de mensagens, pois a perda de uma mensagem de controle prejudica a entrada de um nó na simulação do ambiente virtual.

No cenário que não utilizava diferenciação de serviços, foram usadas filas *DropTail*. O tempo máximo de simulação foi definido como sendo de 5 minutos, com um grupo *multicast* e um servidor de *locale* no grupo. No cenário com diferenciação de serviços foram usadas filas RIO, para que os pacotes que estivessem dentro do perfil de tráfego configurado obtivessem um tratamento diferenciado em relação aos pacotes fora do tráfego.

5.3 MODELO DE TRÁFEGO

O modelo de tráfego utilizado para simular a aplicação é composto de mensagens de controle (*Locale com Status, Object State Summary*) que são enviadas no inicio de cada interação de cada usuário com a simulação. Estas mensagens são trocadas entre o usuário que deseja participar da simulação ou que deseja sair e o servidor de *Locales* que irá aprovar ou rejeitar sua requisição. Após isso, são geradas mensagens de atualização com 100 bytes de tamanho a cada 50 milisegundos em média, gerando um tráfego constante no período que essas mensagens de atualização(*Object State*) estão sendo enviadas. Usando a semente aleatória do NS, foi criada uma dinâmica de entrada e saída dos nós do mundo virtual. Dessa forma, quando um usuário deseja se conectar ao mundo virtual, ele deve enviar uma mensagem *Locale Com Status* com tamanho de 126 bytes para o servidor de *Locales*. Quando a conexão é aceita (não foi incluído na simulação o caso no qual o servidor de *Locales* rejeita o pedido do usuário), o servidor de *Locales* também envia uma mensagem de resposta com um tamanho médio de 126 bytes. Após a conexão ser aberta, o servidor envia mensagens *Object State Summary* com um tamanho aproximado de 1000 bytes para atualizar o nó que está entrando na simulação em relação a todos os eventos acontecidos no ambiente virtual até então e dos quais ele não participou.

Após estes passos iniciais, o tráfego se torna constante, com mensagens de atualização sendo enviadas a cada 50 milisegundos em média. Quando um usuário deseja se desconectar

do ambiente virtual, ele também deve enviar uma mensagem *Locale Com Status* para o fechamento de sua sessão. O servidor de *Locales* deverá enviar uma mensagem de resposta autorizando a desconexão e atualizando as bases de dados dos outros usuários para que eles retirem o avatar do usuário desconectado de seus Modelos de Mundo. Com isso, durante a participação do usuário no Mundo Virtual existe um tráfego constante de 2000B/s gerado por cada nó participante.

Foram criadas duas classes de tráfego: uma com maior prioridade, que corresponde à taxa de transmissão dos pacotes das mensagens de controle e outra classe correspondente à taxa de transmissão dos pacotes das mensagens de atualização. Com isso, o fluxo correspondente às mensagens de controle teria um tratamento prioritário em relação aos pacotes correspondentes ao fluxo das mensagens de atualização. Dessa forma, a taxa para que os pacotes sejam considerados dentro do perfil era de até 1000B/s. Como as mensagens de atualização possuem uma taxa de transmissão muito maior que as mensagens de controle, sua taxa de transmissão facilmente ultrapassa as mensagens de controle, e com isso estes pacotes são marcados como fora do perfil, recebendo um tratamento menos prioritário em relação aos pacotes dentro do perfil.

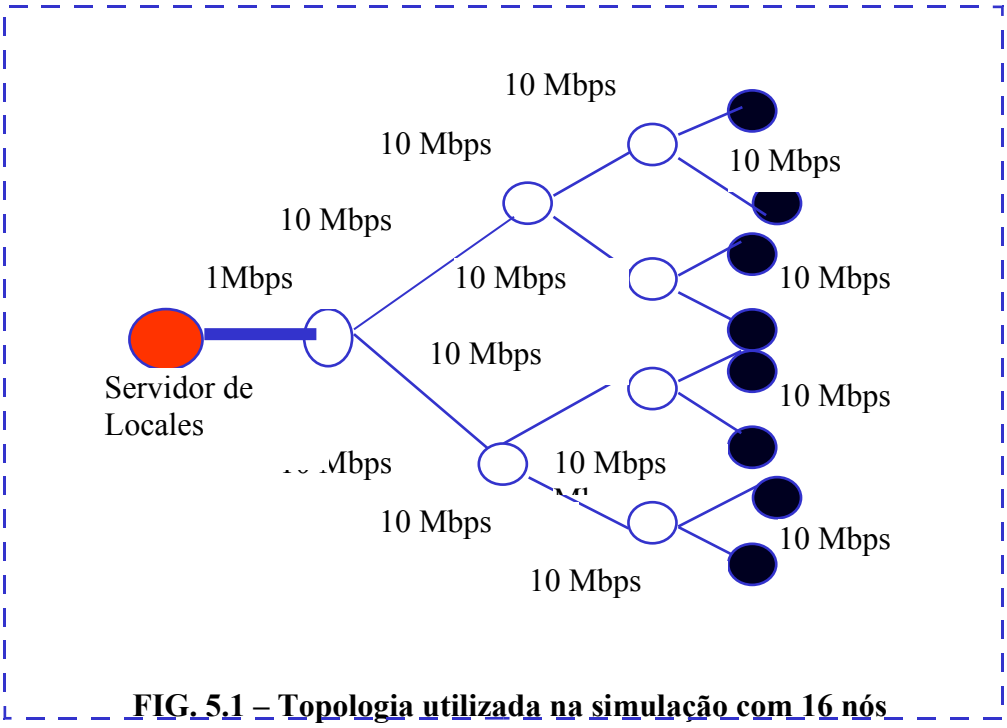

FIG. 5.1 – Topologia utilizada na simulação com 16 nós

5.3.1 APLICAÇÃO DO MODELO DE ERRO NO NS

Para o estudo de vários cenários de congestionamento diferentes, foi usado o modelo de erro do NS. O modelo de erro é baseado na taxa de erro por pacotes, isto é, no percentual de pacotes perdidos em cada direção do enlace. Um trabalho onde a aplicação do modelo é feita pode ser visto em (VIDAL & DUARTE, 2002).

Nesta segunda etapa de simulações, foi usado o modelo de erro do NS para simular vários cenários de congestionamento para a aplicação.

A aplicação do modelo variou de uma taxa de 1% a 5% de perda de pacotes. No gargalo que liga o servidor de *Locales* ao resto da topologia, foi mantida uma taxa de erro constante de 5%. Estes valores representam uma taxa de perda de pacotes considerada média e podem representar uma variação de taxa de perda durante 1 hora em redes de alta velocidade ou durante um dia no MBONE (Zhang et al., 2000). Os dois modelos criados, com diferenciação de serviços e sem, foram submetidos às mesmas taxas de perdas e seus resultados posteriormente comparados.

5.4 SIMULAÇÕES REALIZADAS E INTERVALO DE CONFIANÇA

Na primeira simulação não foi usado nenhum modelo de taxa de erros, por isso a situação de congestionamento gerada na simulação foi decorrente do próprio envio de dados das aplicações em enlaces com capacidade menor que as taxas de transmissão da aplicação.

O principal parâmetro analisado foi a perda de pacotes, tanto de mensagens de atualização quanto de mensagens de controle. Foi analisada a perda de pacotes média coletada a cada 30 segundos da simulação. Os scripts foram executados com valores aleatórios diversas vezes, e a taxa média de perda foi calculada baseada nos resultados destes *scripts.*

O intervalo de confiança foi calculado para o nível de confiança de 95%, usando-se uma amostra de trinta repetições e usando-se uma distribuição Normal. Os resultados obtidos podem ser visualizados na TAB. 5.1.

TAB. 5.1 – Intervalos de confiança para a simulação de perda de pacotes

Intervalo da Simulação	Intervalo de Confiança
0.5	338,9 ± 13,8172
1.0	934, 2 ± 10,22525
1.5	1223,9 ± 10,78454
2.0	1608,9 ± 42,90923
2.5	2386,9 ± 22,71724
3.0	3749,8 ± 21,22799
3.5	5945,6 ± 29,04184
4.0	6911,5 ± 20,62931
4.5	7450,6 ± 21,07746
5.0	8480 ± 18,51561

5.5 RESULTADOS COLETADOS

Nesta seção serão apresentados os resultados obtidos com as simulações criadas a partir da topologia e parâmetros previamente descritos neste trabalho. Os resultados se dividiram basicamente em dois grupos: os resultados obtidos sem a utilização do modelo de erro do NS e os resultados obtidos utilizando este modelo. Além disso, foi analisado também o número médio de mensagens de controle necessárias para abrir uma conexão nos dois modelos.

5.5.1 SIMULAÇÕES SEM O USO DO MODELO DE ERRO

A FIG. 5.2 mostra o comportamento da plataforma simulada sem o uso de diferenciação de serviços e com o uso. Ela mapeia a perda de pacotes no decorrer dos 5 minutos da simulação. Foram enviadas aproximadamente 80.420 mensagens, somadas as mensagens de atualização, controle e próprias à comunicação *multicast.* É possível observar que o número

de pacotes perdidos tem um crescimento mais intenso a partir do 2,5 minuto de simulação, onde um número maior de fontes de tráfego transmite simultaneamente seu tráfego constante de mensagens de atualização. Pode ser observado também, que para quase todo o tempo da simulação, existe uma tendência de que a quantidade média de perda de pacotes para a plataforma sem o uso da diferenciação de serviços foi maior, com 12,5 % de aumento em relação ao modelo que usava esta técnica. A alta perda de pacotes pode ser explicada por uma baixa capacidade nos *buffers* das filas de mensagens, que em uma situação de congestionamento acabam por descartar um número excessivo de mensagens.

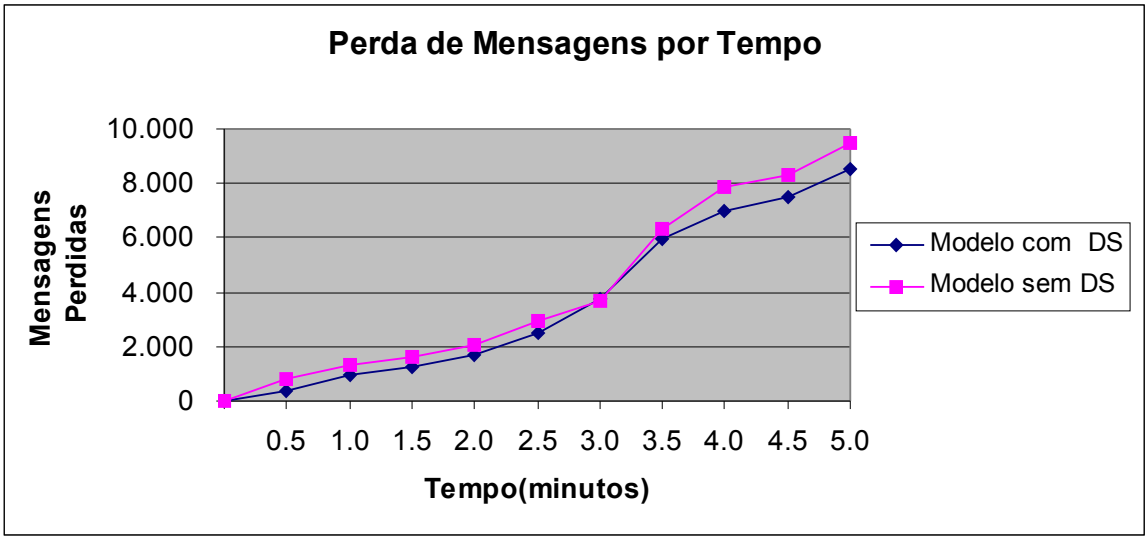

FIG. 5.2 – Número de mensagens perdidas com e sem o uso da diferenciação de serviços

Já na FIG. 5.3 é mostrado o comportamento das duas plataformas em relação à perda das mensagens de controle. Nas simulações deste tipo, houve uma transmissão média de 215 mensagens de controle. Para este caso, as simulações realizadas sugeriram que o uso da diferenciação de serviços também se mostrou eficaz, com uma tendência de taxa de perda 5,67% menor que o modelo sem o uso. Aproximadamente entre o 3º e 4º minuto de simulação, o modelo sem diferenciação de serviços apresenta uma taxa de perda menor que o

modelo que usa a técnica, mas no resto do período da simulação fica clara a vantagem do modelo que adota a técnica em relação ao outro modelo.

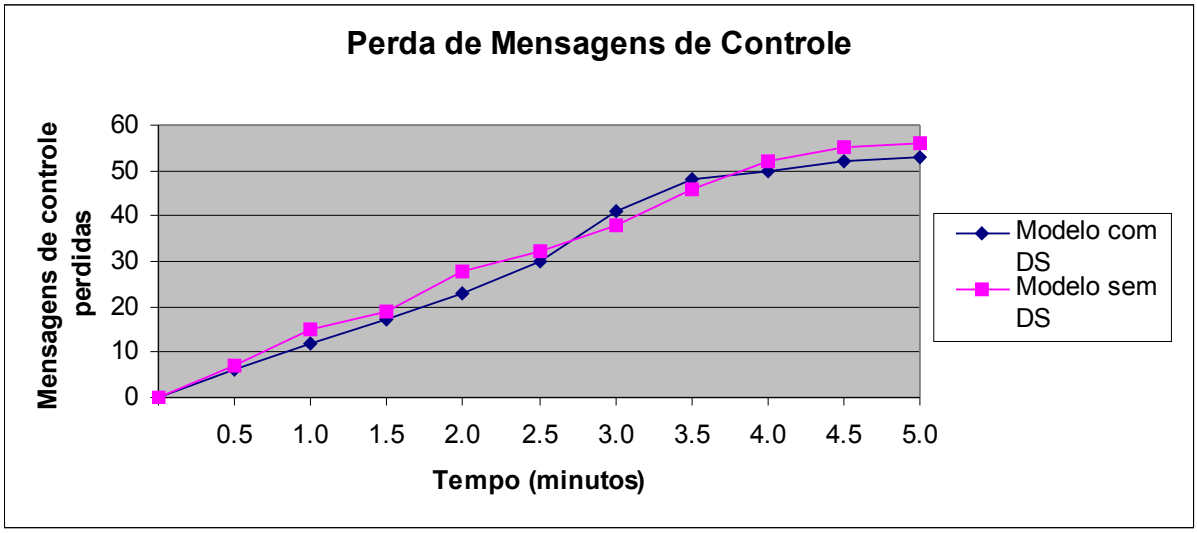

FIG. 5.3 - Variação da perda de mensagens de controle com e sem o uso da diferenciação de serviços

5.5.2 SIMULAÇÕES COM O USO DO MODELO DE ERRO

Após os resultados coletados sem o uso da diferenciação de serviços, foi aplicada uma taxa média de perdas, variando de 1% a 5% de perdas para que fosse possível analisar o comportamento da plataforma em um cenário de congestionamento. As simulações foram executadas diversas vezes para cada taxa de perda e depois foi tirada uma média dos resultados obtidos. A figura 5.4 mostra o comportamento da plataforma para os dois modelos com a aplicação da taxa média de perdas. Pode ser observado que a taxa média de perda de pacotes, calculada com base em todos os resultados gerados pelos scripts sugere um aumento de 9,13% no modelo sem o uso de diferenciação de serviços.

Em relação às mensagens de controle, também acontece uma tendência à uma melhora significativa com a aplicação da diferenciação de serviços. Na figura 5.5, é possível observar que quanto maior vai se tornando a taxa de erro dos enlaces, maior se torna a diferença na

taxa de perda entre os dois modelos. Neste caso, a taxa média de perda de pacotes sugere cerca de 10,2% de aumento nas simulações que não usam a diferenciação de serviços.

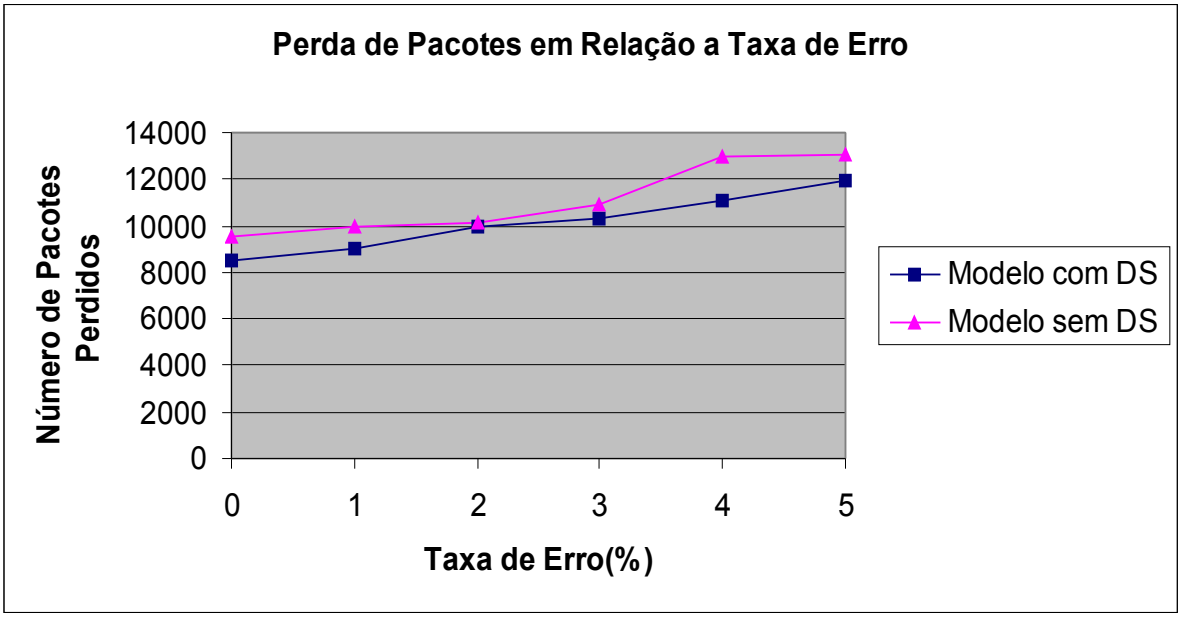

FIG. 5.4 - Variação da perda de mensagens diferentes taxas de erro nos dois modelos

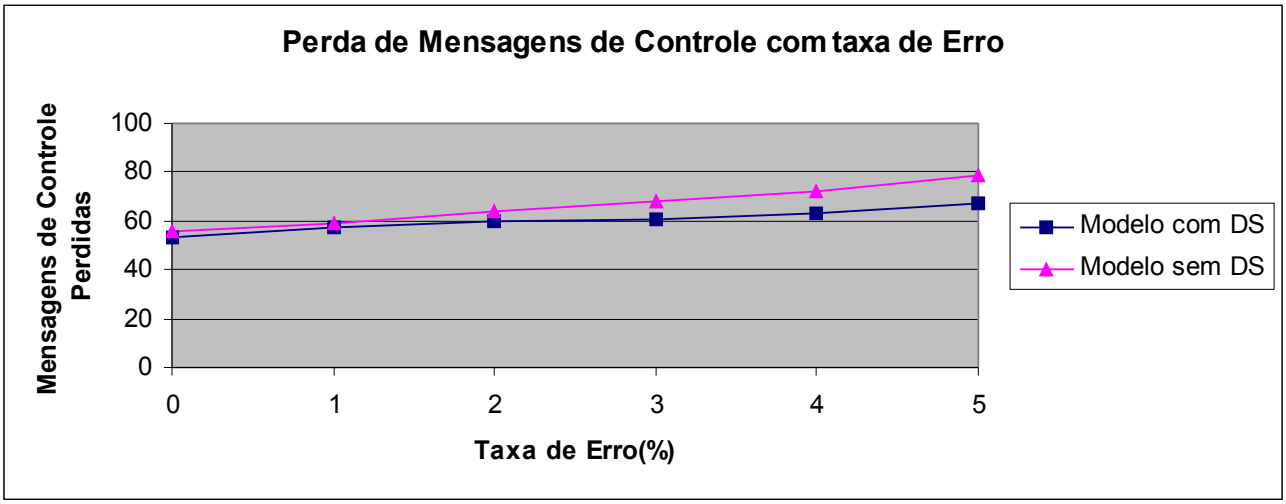

FIG. 5.5 - Variação da perda de mensagens de controle com diferentes taxas de erro nos dois modelos

5.5.3 NÚMERO MÉDIO DE MENSAGENS PARA ABERTURA DE CONEXÃO

Outro aspecto analisado pelas simulações foi o número médio de mensagens necessárias para se abrir uma conexão com o mundo virtual nos dois modelos.

Sabe-se que, para a abertura de um canal de comunicação entre um nó que deseja entrar na simulação e o ambiente virtual, é necessário o envio de uma mensagem *Locale Com Status* do nó requisitante para o servidor de *Locales*. Essa mensagem de requisição deve ser respondida com uma mensagem de aceitação do pedido por parte do servidor, onde será informado ao nó qual o endereço *multicast* do grupo com o qual ele deve interagir recebendo e enviando mensagens de atualização.

No caso do modelo sem o uso da diferenciação de serviços, o protocolo TCP é utilizado para a transmissão de mensagens de controle. Com isso, acontece uma retransmissão das mensagens perdidas enviadas tanto pelo nó quanto pelo servidor. Já no modelo que usa apenas o protocolo UDP, não acontecem essas retransmissões, e com isso o nó emissor deve recomeçar o processo de abertura de conexão com o servidor novamente.

Desta forma, foram feitas várias simulações com os dois modelos, e foi possível calcular uma tendência para o número médio de mensagens necessárias para a abertura da comunicação com o ambiente virtual utilizando os dois modelos. O gráfico da FIG. 5.6 apresenta os dados coletados nas simulações para diferentes taxas de erros.

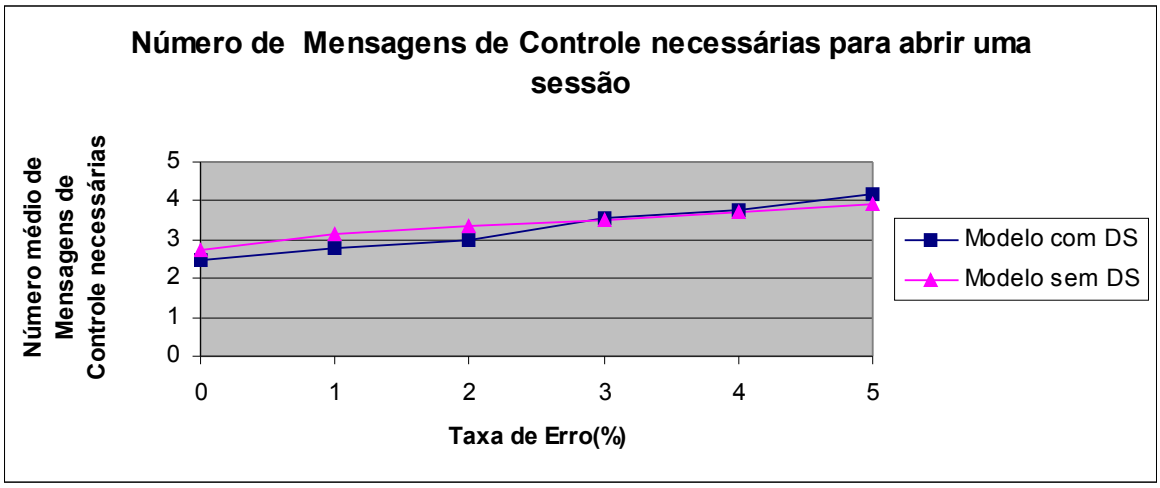

FIG. 5.6 - Número médio de mensagens de controle necessárias para abertura de uma sessão na simulação nos dois modelos

Foi possível inferir deste gráfico, que inicialmente, a abordagem utilizando diferenciação de serviços leva uma pequena vantagem no número de mensagens necessárias que o modelo que usa o protocolo TCP, devido ao tratamento prioritário que as mensagens de controle recebem nas filas RED/RIO, que fazem com que exista uma perda menor deste tipo de mensagem.

Entretanto, é possível observar que à medida que a taxa de erro cresce na simulação, o modelo que usa o TCP começa a ter uma pequena vantagem em relação ao modelo com diferenciação de serviços. Isso pode ser explicado pelo fato que quando uma mensagem de controle é perdida com o TCP, acontece uma retransmissão apenas daquela mensagem. Já em relação ao outro modelo, quando acontece uma perda de mensagem, seja de requisição de abertura de sessão ou de resposta do servidor de *Locales,* a única alternativa é o processo requisitante começar novamente o processo, enviando uma mensagem para o servidor de *Locales*. Com isso, sempre que uma mensagem de controle é perdida neste modelo, são necessárias ao final, quatro mensagens de controle para que a conexão seja estabelecida enquanto que no outro modelo, seriam necessárias três mensagens quando uma fosse perdida. Dessa forma, ainda que a diferenciação de serviços garanta uma perda menor de mensagens de controle, quando uma delas é perdida, acontece o envio de mais duas mensagens o que ocasiona o aumento no número médio de mensagens necessárias.

Com isso, é possível constatar que ainda que as simulações tenham sugerido vantagens em relação ao uso da diferenciação de serviços nos modelos analisados, a utilização de protocolos de comunicação confiáveis com mecanismos de retransmissão para envio de

mensagens críticas à aplicação não pode ser de todo descartada no projeto da arquitetura de comunicação de um ambiente virtual colaborativo em grande escala.

6. CONCLUSÕES

Nesse capítulo será feita uma avaliação do trabalho apresentado e serão apresentadas as contribuições dadas pela elaboração desse trabalho e perspectivas para trabalhos futuros.

6.1 AVALIAÇÃO DO TRABALHO

Este trabalho teve por objetivo a avaliação do desempenho de um protocolo para ambientes virtuais colaborativos em grande escala com o uso de técnicas de Qualidade de Serviço. O objetivo do trabalho, foi analisar se a aplicação de técnicas de diferenciação de serviços em lugar do protocolo TCP para transmissão das mensagens de controle resultaria em alguma diferença substancial na perda destas mensagens e na confiabilidade da entrega das mesmas.

O protocolo usado foi o ISTP, presente na plataforma SPLINE usada na construção de mundos virtuais em grande escala. Nos primeiros capítulos da dissertação foi realizado um resumo dos conceitos envolvendo ambientes virtuais colaborativos e qualidade de serviço.

Após isso, foi exposta uma especificação de um conjunto de funções e mensagens do protocolo, e esse conjunto foi implementado no simulador NS, para que seu comportamento fosse analisado nos mais variados cenários.

No trabalho, foi apresentado um conjunto de simulações que visavam analisar o comportamento do protocolo em vários cenários diferentes. Para a simulação de cenários de congestionamentos foi usado o modelo de erro de enlace do simulador NS, para que fosse possível analisar a perda de mensagens de atualização e de controle nos dois modelos para diferentes taxas de erro.

Em cenários onde era aplicado o modelo de erro, a plataforma com o uso de diferenciação de serviços apresentou uma tendência à um comportamento mais eficaz, apresentando taxas médias de perdas de pacotes menores. Além disso, quando analisado

apenas o tráfego e descarte das mensagens de controle, o uso da diferenciação de serviços também sugeriu ser adequado, gerando uma menor taxa de perda dessas mensagens.

Dessa forma, pode-se inferir que a aplicação da diferenciação de serviços torna a plataforma mais tolerante a situações de congestionamento no que diz respeito à perda de pacotes.

Além disso, também foi analisada a quantidade média de pacotes necessária para a abertura de uma sessão com o mundo virtual nos dois modelos, e pôde ser observado que ainda que o mecanismo de diferenciação de serviços traga uma confiabilidade no tráfego de mensagens críticas, a perda de uma mensagem de controle acarreta um reenvio maior de mensagens que no modelo que usa o protocolo TCP para o tráfego destas mensagens.

6.2 CONTRIBUIÇÕES

Em decorrência do trabalho de pesquisa desenvolvido e dos resultados obtidos, podemos citar como principais contribuições desta dissertação:

- Uma extensão do simulador NS voltada para uma plataforma de um ambiente virtual colaborativo em grande escala.
- A modelagem e a implementação de um subconjunto de funções do protocolo ISTP.
- Uma análise comparativa do uso da diferenciação de serviços na plataforma SPLINE, apresentando os resultados do uso da técnica em vários cenários de congestionamento diferentes.
- Apresentação e publicação de um artigo (GONÇALVES et al., 2004) no Workshop de Teses e Dissertações do SVR 2004.

6.3 TRABALHOS FUTUROS

Diversas direções podem ser tomadas a partir dos resultados apresentados neste trabalho:

- Usar topologias maiores na simulação de forma a avaliar a plataforma de forma mais precisa.

- Execução das simulações um número maior de vezes, obtendo com isso, resultados estatísticamente mais confiáveis.

- Criação de um número maior de grupos *multicast* para representar um número maior de partições do mundo virtual aumentando assim, a dinâmica de saída e entrada de usuários dos grupos.

- Uso de técnicas de diferenciação de serviços mais sofisticadas para que se possa comparar o uso de classes com características de tráfego e atraso diferentes e com tratamentos diferenciados.

- Execução de simulações medindo o parâmetro atraso na entrada dos usuários no mundo virtual nos dois modelos, assim como o também o tempo de *handoff* da aplicação.

- Implementação de um conjunto maior de mensagens e funções do protocolo ISTP.

- A ampliação do modelo de simulação para suportar avatares com características diferentes em relação a tamanho e comportamento.

- A simulação da utilização da plataforma SPLINE em vários tipos de aplicações diferentes, comparando assim as diferenças existentes no tráfego de cada aplicação.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ALBERIO, M. V., OLIVEIRA , J. C. de. **ACOnTECe - Ambiente Colaborativo para Treinamento Cirúrgico**, SIMPÓSIO BRASILEIRO DE REALIDADE VIRTUAL (SVR2004), 2004**,** São Paulo.Workshop de Teses e Dissertações.

ALLISON, D., et al. **The Virtual Gorilla Exhibit**, IEEE Computer Graphics and Applications 17, 6, 1997, 30 – 38.

ANDERSON, D. B., et al., **Diamond Park and Spline**: **a Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability**. Disponível em: < http://www.merl.com/reports/docs/TR96-02a.pdf>. Acesso em: 25 maio 2005.

AWK, **Linguagem AWK**, disponível em: http://www.gnu.org/software/gawk/gawk.html>, Acesso em: 01 fev. de 2006.

BARRUS, J.W.; WATERS, R.C.; ANDERSON, D.B. **Locales: Supporting Large Multi-User Virtual Environments**. Disponível em: < IEEE Computer Graphics and Applications, 1996>. Acesso em: 06 abr. 2005.

BLAKE , S., et al. **An Architecture for Differentiated Services.** Disponível em: <ietf.org/rfc/rfc2475.txt>. Acesso em: 12 fev. 2005.

BRADEN, B. et al., **Reccommendations on Queue Management and Congestion Avoidance in the Internet,** abril de 1998. Disponível em: <ftp://ftp.isi.edu/in-notes/rfc2309.txt>. Acesso em: 20 de maio de 2006.

BRADEN, R., CLARK, D., SHENKER, S., **Integrated Services in the Internet Architeture:** An Overview, RFC 1633, 1994.

BRADEN, R., ZHANG, L., BERSON, S., **Resource Reservation Protocol (RSVP) – Verson 1 Functional Especification** – RFC 2205, 1997.

CADORIN, E.A.; OLIVEIRA, J.C. ACAmPE : um Ambiente Virtual Colaborativo para a Área de Petróleo. In: SIMPÓSIO BRASILEIRO DE REALIDADE VIRTUAL (SVR2004), 2004, São Paulo. Workshop de Teses e Dissertações.

COUTINHO, E.S. et al. Armazenamento e Recuperação de Objetos em Ambientes Virtuais Colaborativos para SIG - 3D. In: SIMPOSIUM ON VIRTUAL REALITY (SVR2003), 6., 2003. Workshop de Teses e Dissertações... p. 453-460

DEERING, S., PARTIDGRE, C. e WAITZMAN, D., **Distance Vector Multicast Routing Protocol,** RFC 1075, 1988.

DEERING, S., **Host Extensions for IP Multicasting**, RFC-1112, agosto de 1989.

DERIGGI JR., F. V. Et al. **CORBA Platform as Support for Distributed Virtual Environments**, IEEE VIRTUAL REALITY 99 (VRAIS,1999), 1999, Houston, Texas (USA).

ERASLAN, M. et. al. A Scalable Network Architecture for Distributed Virtual Environments with Dynamic QoS over IPv6. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 8., 2003.

FALL, K.; VARADHAN, K. **NS Notes and Documentation**, Techical Report, The VINT Project. 1997

FLOYD, S.; JACOBSON, V. Random Early Detection Gateways for Congestion Avoidance**. IEEE/ACM Transactions on Networking**, vol 1, no. 4, p. 362-373, 1993.

GARVEY, R. E.; MONDAY, P. **SIMNET SIMulator NETwork**, BBN Technical Note.1988

GONÇALVES, J. P. G.; VIDAL, P. C. S.; OLIVEIRA, J. C.; O Emprego da Diferenciação de Serviços em Ambientes Virtuais Colaborativos. In: WORKSHOP DE TESES E DISSERTAÇÕES DO SIMPÓSIO BRASILEIRO DE REALIDADE VIRTUAL (SVR2004), São Paulo, outubro de 2004.

GREENHALGH C. et al. **A QoS archtecture for collaborative virtual environments**. Proceedings of the ACM Multimedia. Orlando: [S.n.], 1999. p.121-130.

GOBEL, M., **Industrial Applications of VEs**. IEEE Computer Graphics and Applications, 17, 1996, pg 10-13.

GREENHALGH, C.; BENFORD, S. **A Multicast Network Architecture for Large Scale Collaborative Virtual Environments:** Multimedia Aplications, Services and Techniques – ECMAST'97. 1997

HAGSAND, O**.** Interactive Multiuser VEs in the DIVE System. **IEEE Multimedia**, v. 3, n.1, p.30-39, 1996.

HEINANEN, J.; et al. **Assured Forwarding PHB Group.** Disponível em: <www3.ietf.org/proceedings/98doc/I-D/draft-ietf-delfserv-af-03.txt>. Acesso em: 30 jan. 2005.

HOOK, D. J. V.; RAK, S. J.; CALVIN, J. O. **Approaches to RTI Implementation of HLA Data Distribution Management Services**. DIS Workshop on Simulation Standards, Cambridge Massachussetts, 1996. Massachussetts Institute of Technology.

HUITEMA, C. **Routing in the Internet.** [S.l.]: Prentice Hall, 1995.

JACOBSON, V.; NICHOLS, K.; PODURI, K. **An Expedited Forwarding PHB.** Disponível em:< http://rfc.sunsite.dk/rfc/rfc2598.html>. Acesso em: 25 jan. 2005.

KAMIENSKI, C. A.; SADOK, D. **Qualidade de Serviço na Internet**. Minicurso apresentado no SBRC'2000, 2000.

KUROSE, J. F., ROSS, K. W. **Redes de Computadores e a Internet:** uma nova abordagem. [S.l.]: Addison Wesley, 2003.

MACEDONIA, M. et al. NPSNET: a Network Software Architecture for Large Scale Virtual Environments. **PRESENCE: Teleoperators and Virtual Environments**, v. 3, n. 4, p.265-287, 1994.

OLIVEIRA, J. C; GEORGANAS, N. D. VELVET: an Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs. **PRESENCE: Teleoperators and Virtual Environments**, v. 12, n. 6, p.555-580, 2003.

LINGUAGEM OTCL. Disponível em:< http://otcl-tclcl.sourceforge.net/.> Acesso em: 01 mar. de 2006.

RODRIGUES, S.G.; OLIVEIRA , J.C.; PEIXOTO , M. V. ADVICE: um ambiente virtual colaborativo para o Ensino a Distância. In: IX SIMPOSIUM ON THE WEB AND MULTIMEDIA SYSTEMS (WEBMÍDIA2003), 9., 2003. Workshop de Teses e Dissertações. p.593-596.

ROSEN, E. C.; VISWANATHAN, A; CALLON, R. **Multiprotocol Label Switching Architecture, RFC3031,** Junho, 2000.
Acesso em: 20 jan. 2005.

STRIEGEL, A. **GenMCast Software Extension for ns-2**, ND CSE Technical Report, 2004.

VIDAL, P. C. S, DUARTE, O. C. M. B. Estabelecimento de Reserva de Recursos para Fluxos Multicast em Túneis IP-sobre-IP. In: V SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E HIPERMÍDIA - SBMIDIA 2002, p. 252-259, 2002.

VRML, **The Virtual Reality Modeling Language**. Disponível em: <http://www.vrml.org> Acesso em: 28 jan. 2006.

WATERS, R.C.; ANDERSON, D. B.; SCHWENKE, D. L. Design of the Interactive Sharing Transfer Protocol. In: IEEE WORKSHOPS ON ENABLING TECHNOLOGIES: infrastructure for collaborative enterprises, 6., 1997.

ZHANG, Y.; PAXSON, V. e SHENKER, S. **The Stationary of Internet Path Properties: Routing, Loss and Throughput,** ACIRI Technical Report, maio, 2000.

8. APÊNDICE

8. 1 APÊNDICE 1: O SIMULADOR NS

O NS é um simulador de eventos discretos focado para o desenvolvimento de protocolos e aplicações em redes de computadores direcionado para a arquitetura TCP/IP. O simulador é mantido pelo projeto VINT (*Virtual InterNetwork Testbed*). Sua estrutura utiliza duas linguagens: C++ e Otcl. A linguagem C++ é utilizada no desenvolvimento do núcleo de um protocolo, enquanto a linguagem Otcl é utilizada na criação e configuração dos parâmetros utilizados nos scripts de simulação.

Para criar uma simulação é preciso escrever um *script* Otcl, que será lido por um interpretador e gerará uma saída específica. No *script* de simulação diversas etapas devem ser realizadas, como descrito abaixo e exemplificado na figura 8.1

• criar a instância do simulador;
• definir uma topologia de rede adequada para o trabalho;
• definir os agentes da camada de transporte;
• criar as aplicações geradoras de tráfego;
• inserção de erros, modificação das condições da rede.

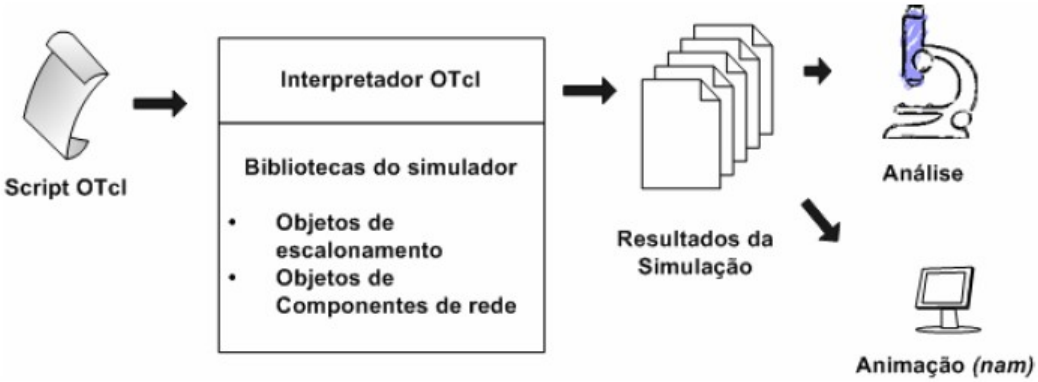

FIG. 8.1 – Etapas da simulação no simulador NS.

O NS foi escolhido para que se possa alcançar um modelo semelhante ao existente na *Internet*. Ele oferece uma infra-estrutura para simulação de protocolos de rede com abstrações para nós e enlaces. Esta infra-estrutura também compreende alguns algoritmos de roteamento utilizados na Internet, incluindo roteamento *multicas*t. Além disto, o NS engloba vários modelos de protocolos usuais na *Internet* como o TCP e o UDP. O NS permite a integração de

novos módulos em sua estrutura. Estes módulos podem representar um protocolo, uma aplicação, uma fonte de tráfego. O NS se encontra atualmente na versão 2.29 e pode ser obtido sem custos através do seu *site* na *Internet*. Desenvolvido para sistemas Linux, após instalado precisa ter seus arquivos compilados, e posteriormente testados.

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 5Mb 2ms DropTai
$ns duplex-link $n1 $n2 5Mb 2ms DropTai
$ns duplex-link $n2 $n3 1.5Mb 10ms Drop

# Some agents.
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$udp0 set class_ 0

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

$ns connect $udp0 $null0
$ns at 1.0 "$cbr0 start"

puts [$cbr0 set packetSize_]
puts [$cbr0 set interval_]

# A FTP over TCP/Tahoe from $n1 to $n3, flowid 2
set tcp [new Agent/TCP]
$tcp set class_ 1
$ns attach-agent $n1 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.2 "$ftp start"
```

FIG. 8.2 – Exemplo de script de simulação em Otcl

Na FIG. 8.2, é mostrado um script onde são criados quatro nós, de n0 a n3 e três enlaces bidirecionais entre os nós n0 e n2, n1 e n2 e n2 e n3.

Após isso, são criados um agente da camada de transporte UDP anexado ao nó n0 e uma fonte de tráfego do tipo CBR(*Constant Bit Rate*) que usa o agente UDP.

Para receber os pacotes que a fonte de tráfego irá enviar através do agente UDP, é criado um agente *Null*, ligado ao nó n3. Posteriormente é chamado o escalonador do simulador para gerar um evento de transmissão de tráfego no tempo 1.0 segundos.

Após isso é criado um agente da camada de transporte TCP que é ligado ao nó n1 e um agente destino no nó n3.

Ao fim, é criada uma fonte de tráfego do tipo FTP que é anexada ao agente TCP, com um evento de geração deste tráfego no tempo 1.2 segundos.

Após o *script* ser executado pelo módulo de simulação, começa a fase de análise de resultado. Esta análise pode ser realizada através da visualização da simulação no NAM ou através de análise do arquivo *trace*. O NAM é uma ferramenta de visualização do ambientes de simulação. Com ela é possível observar a topologia, o tráfego de pacotes nos enlaces no simulador.

O NS gera arquivos de *trace*, nos quais pode-se encontrar informações sobre tempo de chegada, tempo de partida, tempo em que o pacote é descartado em um enlace ou fila, tamanho do pacote, tipo do pacote, etc. É necessário salientar que o NS oferece uma variedade de *traces* específicos. Como por exemplo, *trace* para redes sem fio, redes fixas, etc.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo