ESPECIFICAÇÃO E ANÁLISE DE UM SISTEMA DISTRIBUÍDO DE REALIDADE VIRTUAL

Regina Borges de Araujo[1]
drea@power.ufscar.br

Dr. Cláudio Kirner
ckirner@power.ufscar.br

Grupo de Realidade Virtual
Departamento de Computação
Universidade Federal de São Carlos
Cx.Postal 676
13565-905 São Carlos - SP

RESUMO

Este artigo mostra a arquitetura, especificação e análise de um sistema distribuído de realidade virtual. Ênfase é dada à minimização do atraso *(lag)* nos vários estágios de processamento do sistema considerado. Como o atraso em um sistema de realidade virtual é altamente dependente da aplicação, um cenário de treinamento em auto-escola é utilizado como ilustração. São utilizados dados dinâmicos na análise do sistema, ao contrário dos trabalhos até então sendo discutidos na literatura, o que pretende ser uma contribuição à área emergente de pesquisa em realidade virtual distribuída.

ABSTRACT

This papers shows the architecture, specification and analysis of a distributed virtual reality system. Emphasis is given to the minimization of lag in the various processing stages of the system being considered. Lag, in a virtual reality system, is highly dependent on the application, so that a car drive training scenario is used to illustrate the system. Dynamic data is used in the analysis, as opposed to static data being discussed so far in the literature, what intends to be a contribution to the emerging area of distributed virtual reality.

1. INTRODUÇÃO

Sistemas distribuídos de realidade virtual (RV) têm surgido que suportam a interação entre usuários dispersos geograficamente, compartilhando um mesmo ambiente sintético gerado por computador. As aplicações variam desde sistemas de telecolaboração[2] até treinamento militar

[1] Doutoranda do Laboratório de Sistemas Integrados da EPUSP.

[2] Sistemas do tipo Trabalho Cooperativo Apoiado por Computador (CSCW) onde usuários remotos compartilham um ambiente virtual para realizar uma tarefa em comum.

[MACE94, CALV93]. O sentimento de "imersão" e "presença" do usuário dentro do sistema é o que caracteriza um ambiente virtual.

Para que a ilusão de "presença" do usuário no ambiente virtual seja mantida, o sistema tem que oferecer um tempo de resposta dentro de um limiar aceitável, além das ações dos usuários terem que ser refletidas de forma imediata e simultânea a todos os outros usuários que compartilham o mesmo ambiente. O ambiente tem também que ser realista. Os requisitos acima implicam nas seguintes demandas feitas sobre um sistema de RV, conforme sumarizado por Van Dam [VanD93]: taxas rápidas de atualização das imagens visualizadas pelos participantes; atraso mínimo entre uma ação efetuada pelo usuário e a visualização do resultado desta ação pelo usuário *(lag);* tratamento de múltiplos dispositivos de entrada; e a simulação de um número grande de objetos com comportamento complexo.

As taxas de atualização vem sendo incrementadas dia após dia com o surgimento de *hardware* avançado para a sintetização de imagens. Já o atraso em um sistema de RV, não diminui necessariamente com técnicas de aumento de rendimento do sistema, como a paralelização do processamento, apesar das duas medidas serem relacionadas. Por exemplo, Mathias Wloka [WLOK95] observa que o uso de múltiplos processadores numa configuração de *pipeline,* aumenta o rendimento do sistema mas mantém o mesmo atraso.

Como em um sistema de RV o atraso ocorre em vários níveis de processamento, a diminuição do atraso é um dos grandes desafios na estruturação de um sistema deste tipo. Este artigo trata da especificação e análise de um sistema de realidade vistual distribuído, visando a diminuição do atraso em seu processamento. A seção 2 sumariza a arquitetura do sistema, descrevendo os elementos envolvidos na sua estrutura. A seção 3 mostra a especificação do *software* de um sistema de RV distribuído, descrevendo as funções principais de cada módulo, a interação entre estes módulos e as suas fontes de atraso. A descrição da simulação do sistema especificado, bem como o impacto de variáveis que contribuem de forma mais significativa para o aumento do tempo de processamento do sistema, são discutidos na seção 5. A seção 6 discute brevemente os Trabalhos Relacionados, seguida das Conclusões e Apêndices A e B.

2. Um Resumo da Arquitetura do Sistema de RV Distribuído sendo Especificado

A arquitetura do sistema sendo especificado foi explorada no artigo dos autores Araujo e Kirner [ARAU95a]. Damos a seguir, um resumo do sistema considerado: um sistema de realidade virtual é dividido em sub-áreas estruturadas como um conjunto de células hexagonais. Cada sub-área é coordenada por um servidor de ambiente (SA), responsável pelo armazenamento e gerenciamento de seu respectivo conjunto de dados, além da animação e sincronização dos objetos que "habitam" esta sub-área (os objetos são descritos logo abaixo no texto). Como um sistema de RV pode compreender várias sub-áreas, o ambiente como um todo pode ser distribuído em vários servidores de ambiente. O sistema NPSNET-IV [MACE95], um dos mais significativos sistemas de RV em rede, também estrutura o ambiente em células hexagonais só que, em vez de sub-dividir o sistema em sub-áreas, como o sistema aqui especificado, particiona o ambiente em classes espaciais, funcionais ou temporais, onde cada classe agrupa, respectivamente, entidades do ambiente virtual por proximidade espacial,

por função ou por requisito de tempo que não tempo-real, além de adotar o paradigma de comunicação *peer-to-peer*, onde todos os nós do sistema interagem entre si num mesmo nível hierárquico. Cada nó no NPSNET-IV é gerenciado por um Gerente de Área de Interesse (AOIM).

No sistema de RV sendo considerado aqui, usuários remotos podem compartilhar um mesmo ambiente sintético. Estes usuários são representados no ambiente como **objetos dirigidos pelo usuário (ODU).** Estes objetos podem tomar qualquer forma, dependendo apenas da aplicação: desde figuras geométricas como círculos, quadrados, etc., até automóveis, avião, partes de um corpo humano como mãos, pés, cabeça etc. Os ODU refletem no ambiente virtual, as ações dos usuários captadas através de sensores ligados ou direcionados ao usuário. Dois outros tipos de objeto são colocados no ambiente para aumentar o realismo: os **objetos dirigidos por simulação (ODS)** e os **objetos estáticos (OE).** Os ODS são objetos animados cuja complexidade pode variar de simples objetos dirigidos por um conjunto de ações pré-definidas *(scripts)* até objetos altamente complexos com ações obtidas através de técnicas de inteligência artificial. Já os objetos OE são fixos no ambiente virtual, e caracterizam o contexto da aplicação como árvores, edifícios, avenidas, rios, montanhas etc.

Inicialmente, quando um usuário seleciona uma área de atuação, esta área é despachada de uma sub-área em um servidor de ambiente, juntamente com os objetos ODS e informações sobre ODUs remotos contendo células coincidentes, para a máquina cliente na qual o usuário está conectado. Daí em diante, o cliente passa a ter autonomia no gerenciamento dos objetos sob sua responsabilidade. A área de atuação é um conjunto de células hexagonais que vai se modificando de acordo com a direção e velocidade com as quais o usuário se movimenta na área. Se o usuário dá indícios de movimentação entre uma célula e outra de sua área de atuação - estes indícios são detectados pelo respectivo servidor de ambiente, através de *dead-reckoning*[3] - o servidor de ambiente antecipa ao cliente novas células para serem adicionadas às células já existentes na sua área de atuação. Células da área de atuação que forem sendo deixadas para trás pelo objeto dirigido pelo usuário, são então descartadas pelo cliente. Entretanto, estas células são mantidas por um determinado período de tempo no cliente para o caso de por exemplo, um ODU resolver voltar atrás, contradizendo portanto o indício de cruzamento entre células. Desta forma, a taxa de envio de células antecipadas é ditada pela velocidade com a qual usuário se move dentro das células de sua área de atuação, bem como pela extensão da célula. A figura 1 mostra um cenário da arquitetura discutida acima.

A estrutura do sistema mostrado na figura 1 reflete uma arquitetura híbrida entre *cliente-servidor*, com o servidor de ambientes centralizando o gerenciamento de uma sub-área do sistema de RV, e *peer-to-peer*, onde clientes interagem entre si num mesmo nível hierárquico. A figura 1 se constitui em apenas um dos possíveis cenários da arquitetura proposta já que, de acordo com a área de atuação selecionada por um usuário, sua máquina cliente pode se associar a qualquer um dos servidores de ambiente existentes, bem como aos clientes já lá associados. O objetivo desta estrutura é minimizar o atraso de comunicação entre os clientes,

[3] *dead reckoning* - técnica usada para minimizar o tráfego de rede no reposicionamento dos objetos de um sistema de RV, que consiste em manter dois cálculos sobre o estado de um objeto. Um cálculo é bem preciso e o outro usa um número limitado de parâmetros. Se a diferença entre estes dois cálculos for maior que um valor conhecido, mensagens de reposicionamento do objeto, alvo do cálculo, podem ser geradas e enviadas para atualização das réplicas deste objeto nos respectivos nós da rede.

entre máquinas clientes e seu respectivo servidor de ambientes e entre servidores de ambiente. A comunicação entre estes elementos é detalhada no artigo de Araujo e Kirner [ARAU95a].

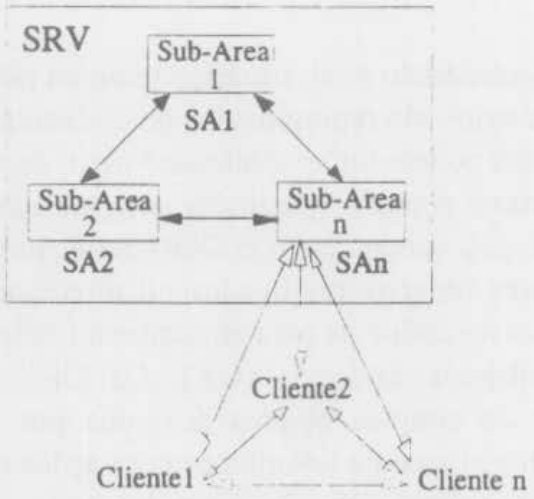


Fig.1 Um Cenário Genérico da Arquitetura Proposta para um Sistema Distribuído de RV - SRV

Um Possível Cenário de Aplicações Suportadas pelo Sistema de RV sendo Considerado

Os atrasos em um sistema de RV são altamente dependentes da aplicação e podem variar em função do número de dispositivos de entrada ligados aos sensores, do modo de operação destes dispositivos, do número de objetos ODS na área de atuação do usuário, da complexidade destes ODS, da complexidade da cena sendo vista pelo usuário, que envolve o número de objetos fixos e dinâmicos e o grau de detalhe de visualização destes objetos, do número de usuários participantes, do próprio comportamento do usuário na aplicação, através de seu ODU, que pode variar desde uma figura geométrica sem muito movimento, até um automóvel ou avião.

Uma possível aplicação suportada pelo sistema de RV sendo discutido é o de um ambiente de treinamento em uma auto-escola onde, os usuários são representados no sistema como motoristas de carros da auto-escola e onde automóveis infratores simulados podem ser inseridos no ambiente de uma cidade virtual para testarem a capacidade do candidado a motorista em lidar com situações que ele/ela certamente enfrentariam no trânsito de uma grande cidade[4]. Vários candidatos podem compartilhar uma área de atuação com células coincidentes - outros exemplos deste tipo de aplicação incluem treinamento militar motorizado, treinamento de forças policiais, por exemplo frente à perseguições motorizadas etc.

[4] Apesar de um cenário ilustrativo da aplicabilidade do sistema estar sendo mostrado, este é modelado com um conjunto de dados dinâmicos de tal forma que, variando-se os dados é possível modelar qualquer tipo de sistema de realidade virtual.

3. ESPECIFICAÇÃO DO SOFTWARE DO SISTEMA

O cliente, ligado a um usuário local e associado a um servidor de ambiente escolhido em função da área de atuação desejada pelo usuário, tem como função a execução de uma série de tarefas, não necessariamente sequenciais, que são efetuadas desde o momento em que o usuário seleciona sua área de atuação, até o momento em que o usuário sai desta área, desconectando-se portanto do servidor de ambiente. Já no servidor de ambientes, as tarefas são inicializadas uma vez e efetuadas *non-stop,* isto é, nunca terminam. As funções do cliente são representadas na figura 2 como um ciclo de tarefas e descritas logo abaixo no texto.

3.1 Especificação das Funções do Cliente

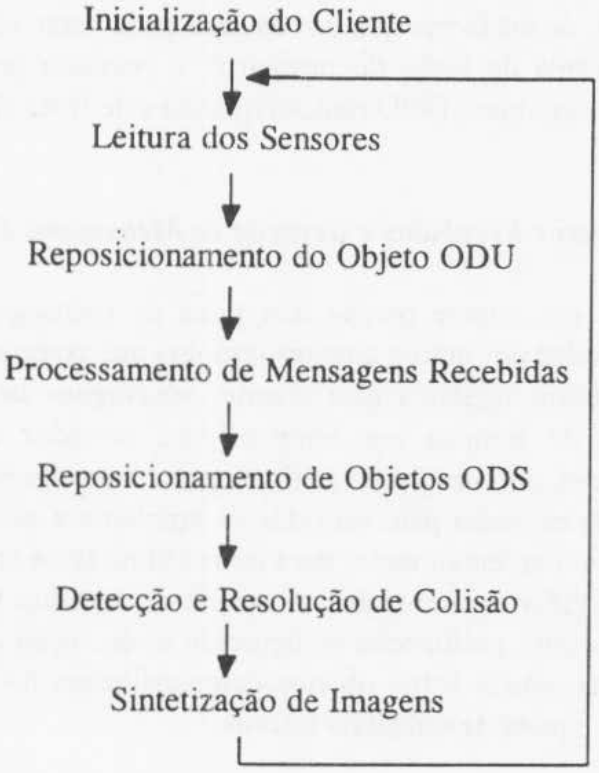


Fig. 2 Ciclo de Tarefas do Cliente

3.1.1 Inicialização do Cliente

Um usuário, ligado a um cliente, pode selecionar uma área desejada de atuação através de uma interface de entrada no sistema, que pode ser apresentada das mais diversas formas, desde uma aproximação da área desejada (através de *zooming)* até por exemplo, a requisição de uma passagem aérea, terrestre etc. para o local desejado. Selecionada a área, uma consulta é gerada, por exemplo, a um diretório global de endereços, para saber o endereço do servidor de ambiente que mantém esta área. O cliente então aguarda a chegada dos objetos estáticos ODS e ODU remotos. O cliente recebe também o endereço de outros clientes que tenham células coincidentes com as da área de atuação do usuário sendo suportado por ele. Feito isto, dá-se início ao ciclo de tarefas do cliente, juntamente com a animação dos objetos ODS que

foram carregados. O estado atual do usuário ligado a este cliente é enviado aos demais clientes remotos. Este usuário agora é um membro ativo de um grupo de usuários que compartilham células coincidentes em um ambiente virtual.

3.1.2 Leitura e Reposicionamento do objeto ODU

Os sensores são lidos através de dispositivos de entrada dedicados. Estes dados são processados pela CPU principal e são a fonte de informação para o reposicionamento do objeto ODU que representa o usuário local e seus movimentos. Como este ODU pode estar sendo visualizado por outros usuários remotos que compartilham de toda ou parte da área de atuação deste usuário local, o reposicionamento deste objeto tem que ser feito também nos clientes remotos. Na aplicação de uma auto-escola, por exemplo, dois usuários dispersos geograficamente podem estar compartilhando uma mesma célula que representa um trecho da Avenida Paulista em São Paulo, de tal forma que um usuário pode estar visualizando o outro (desde que o último esteja na área de visão do primeiro), e portanto precisa receber uma *mensagem de reposicionamento* do objeto ODU remoto quando este tiver alguma modificação significativa.

3.1.3 Processamento de Mensagens Recebidas e Geração de Mensagens Enviadas

No sistema aqui especificado, um cliente recebe três tipos de mensagens: *mensagens de reposicionamento,* que são enviadas por outros clientes remotos que compartilham de parte ou toda a área de atuação do usuário ligado a este cliente; *mensagens de sincronização do sistema distribuído,* enviadas de tempos em tempos pelo servidor de ambiente para ressincronizar os objetos dirigidos por simulação e, finalmente, *mensagens de notificação de antecipação de células,* que são enviadas pelo servidor de ambiente a um cliente cujo ODU esteja dando indicações de que vai se movimentar para outra célula (este processo é detalhado na descrição das funções do servidor de ambiente). A análise do impacto no sistema causado por outros tipos de mensagens como notificação de detecção e resolução de colisão, além de mensagens de controle de concorrência sobre objetos compartilhados no ambiente virtual e mensagens de gerenciamento[5], é parte de trabalhos futuros.

Um cliente decide se precisa ou não enviar uma mensagem de reposicionamento de seu ODU local para outros clientes remotos que estejam visualizando este ODU, quando os movimentos do usuário local, captados por sensores, fugirem ao padrão pré-estabelecido calculado através do *dead-reckoning* (descrito em rodapé da seção 2).

As mensagens são tratadas assim que recebidas pelo sistema, para se evitar o chamado *atraso de sincronização,* procurando compensar ao máximo o atraso que a mensagem sofre durante a sua transmissão pela rede. Um exemplo de atraso de sincronização na recepção de mensagens é dado a seguir: se o momento de chegada de uma mensagem é imediatamente antes do início do estágio de processamento das mensagens, esta mensagem é tratada logo que recebida e, portanto, tem atraso de sincronização igual a zero (considerado o melhor caso). Entretanto, se a chegada da mensagem é imediatamente após a inicialização do estágio de processamento

[5] mensagens enviadas pelo servidor de ambiente para notificar a presença de novos ODU ou ODS na área de atuação de um cliente específico.

das mensagens, os dados recebidos vão ter que esperar até o próximo estágio de processamento (considerado o pior caso). Existe ainda o caso médio, onde a mensagem chega no decorrer do estágio de processamento das mensagens.

Mathias Wloka [WLOKA95] descreve os atrasos de sincronização em um sistema de RV, bem como os outros atrasos decorrentes do processamento do sistema. Araujo e Kirner [ARAU95b] descrevem os atrasos decorrentes da transmissão de mensagens em um sistema de RV distribuído.

3.1.4 Reposicionamento de objetos ODS

Quando a área de atuação do usuário é carregada no cliente, são carregados também o código dos objetos simulados - ODS que "habitam" ou estão de passagem por esta área de atuação. O cliente passa então a processar a animação destes objetos simulados. Esta animação entretanto pode entrar em dissincronização com os mesmos objetos ODS replicados em outros clientes que compartilham células coincidentes da área de atuação do usuário local. Desta forma, o servidor de ambiente é responsável por enviar, de tempos em tempos, uma *mensagem de sincronização* a todos os clientes contendo objetos ODS em células coincidentes, esperando assim que estes objetos, apesar de processados independentemente por cada cliente, se mantenham sincronizados.

3.1.5 Detecção e Resolução de Colisões

Roberts e colegas [ROBE93] categorizam as colisões como sendo de Usuário para Objeto, de Usuário para Usuário e de Objeto para Objeto. Aqui, as colisões são classificadas nas seguintes categorias: 1) entre um objeto ODU e um ou mais objetos ODS; 2) entre objetos ODU; 3) entre objetos ODS e 4) colisão entre um ODU ou ODS e um ou mais objetos estáticos. Na especificação do sistema aqui descrito, toda a detecção e resolução da colisão entre objetos ODS é feita pelo servidor de ambiente (categoria 3). As outras formas de colisão que envolvam um ODU local com outros objetos é feita no cliente já que o cliente mantém a posição mais atualizada de seu ODU e portanto, pode ser capaz de detectar, de forma mais eficiente, as categorias 1, 2 e 4 descritas acima.

3.1.6 Sintetização de Imagens

A sintetização de imagens ou *rendering* é o processo final da visualização das imagens geradas no ambiente virtual. A taxa de geração de quadros por segundo é um parâmetro importante para assegurar o sentimento de "presença" de um usuário no ambiente virtual.

Considera-se aqui que a sintetização de imagens, ou *rendering,* é efetuado apenas nos clientes. O servidor de ambiente não trata da entrada e saída para um usuário ligado localmente. Entretanto, nada impede que a máquina servidora de ambiente sirva também como uma máquina cliente, seguindo os mesmo princípios de gerenciamento de área de atuação que qualquer outro cliente. Naturalmente, o desempenho deste cliente específico pode ser diferente do desempenho dos clientes remotos já que a troca de informação entre cliente e servidor vai ser local. A especificação e simulação desta situação não está sendo considerada neste artigo.

Apesar de considerada aqui apenas a sintetização de imagens, um trabalho futuro a ser investigado é o impacto do atraso no sistema causado por sintetizadores de áudio que reproduzem, no ambiente virtual, sons que coincidem com a posição dos objetos gráficos, além do impacto de "sintetizadores de reações" (atuadores), que são engatilhados como resultado de ações dos usuários sobre o ambiente virtual etc.

3.2 Especificação das Funções do Servidor de Ambiente

O servidor de ambiente mantém uma base de dados geométrica dos objetos estáticos, bem como dos objetos ODS e ODU da sub-área sob sua responsabilidade. A posição e orientação de todos os objetos ativos do sistema são gerenciados pelo servidor de ambientes para que estas informações, juntamente com o código dos objetos dinâmicos, sejam carregados nos clientes de acordo com a área de atuação selecionada pelos respectivos usuários. Estas e outras funções do servidor de ambiente são descritas logo abaixo no texto. A figura 3 mostra as funções do servidor de ambiente como um ciclo de tarefas.

3.2.1 Inicialização do Servidor de Ambiente

O servidor é inicializado uma vez, a menos que novas versões da sub-área sob sua responsabilidade sejam implementadas. A inicialização de uma sub-área compreende basicamente o carregamento da base de dados geométrica, a inicialização da animação dos objetos ODS ali existentes, além de tarefas de gerenciamento como a atualização de um diretório de informações sobre os demais servidores de ambiente do sistema de realidade virtual, por exemplo o X500, etc. O servidor então se prepara para receber associações de clientes, para quem são enviadas as áreas de atuação desejadas e respectivos objetos ODS. O servidor prepara-se também para receber mensagens de outros servidores de ambiente comunicando, entre outras coisas, a movimentação de objetos entre sub-áreas.

3.2.2 Reposicionamento de Objetos ODS

A animação dos objetos ODS é feita, no sistema aqui especificado, em dois níveis de complexidade: um nível mais simples, apenas com a atualização da posição do objeto e um nível mais complexo, com a animação detalhada do objeto.

O servidor de ambiente é responsável pelo nível mais simples do processamento da animação de todos os objetos ODS em sua sub-área, além da manutenção de sincronização das réplicas destes objetos nas máquinas clientes. O servidor de ambientes é responsável também pelo monitoramento da movimentação destes objetos entre células e até entre sub-áreas. Isto vai implicar no envio de mensagens de recolocação de ODS para eventuais clientes, quando o movimento for entre células, ou envio de mensagens para outros servidores de ambiente quando o movimento for entre sub-áreas.

3.2.3 Reposicionamento dos Objetos Dirigidos pelo Usuário

O servidor de ambiente e seus clientes utilizam a técnica de *dead-reckoning* para diminuir a comunicação entre si na atualização dos objetos ODU. Um cliente só envia uma mensagem de reposicionamento de seu ODU local, quando a posição deste, calculada localmente, variar de

um valor acima do previsto. Caso contrário, o próprio servidor de ambiente, e os clientes remotos ao cliente acima, reposicionam o objeto em função dos cálculos efetuados em suas respectivas máquinas.

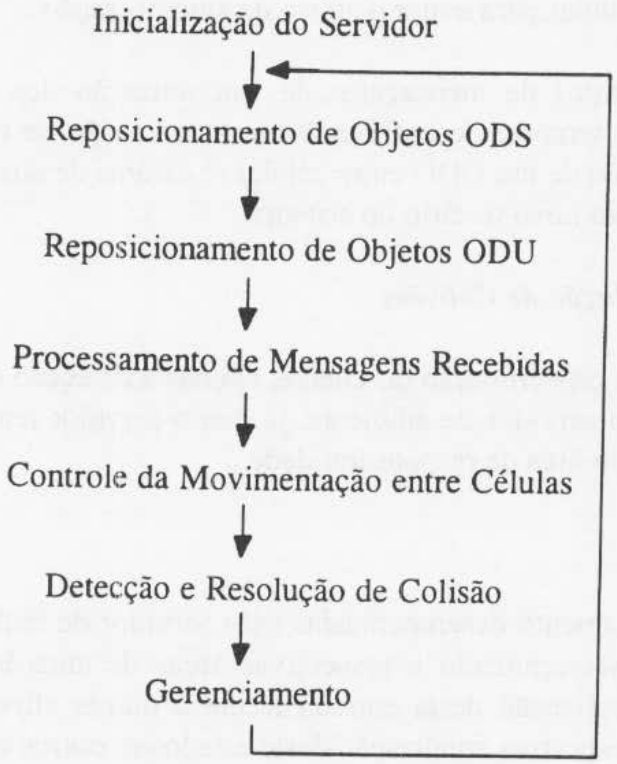


Fig.3 Ciclo de Tarefas do Servidor de Ambiente

3.2.4 Controle da Movimentação de Objetos ODU entre Células

Quando um objeto ODU dá indicação de que vai mudar de uma célula para outra, o servidor de ambiente detecta esta intenção, através do *dead-reckoning,* e envia, para o respectivo cliente, uma *mensagem de notificação de antecipação de células* contendo as eventuais futuras células a serem visualizadas pelo usuário ligado àquela máquina cliente. A taxa de envio destas mensagens de notificação é determinada pela velocidade dos objetos ODU e pela extensão de cada célula. Se o ODU representando o usuário for um pedestre, a velocidade vai variar de poucos kilômetros por hora, significando uma baixa taxa de envio de mensagem do servidor de ambiente para o cliente. Esta taxa pode aumentar consideravelmente quando o ODU for um automóvel ou avião.

Um outro fator a ser considerado aqui, é o nível desejado de antecipação de células. Se no ambiente virtual sendo considerado pelo usuário, existirem objetos que podem ser visualizados a grande distância, por exemplo prédios muito altos numa região plana, ou montanhas que circundam uma cidade, pode não bastar apenas o envio do nível básico de células antecipadas (consideradas aqui como antecipação de nível 1, contendo três células adjacentes à borda para onde o ODU se dirige). Pode ser preciso o envio de vários níveis de antecipação de células (níveis 2, 3 e até mais dependendo da extensão de cada célula).

3.2.5 Processamento de Mensagens Recebidas e Geração de Mensagens a Serem Enviadas

No sistema aqui considerado, o servidor de ambiente recebe apenas mensagens de reposicionamento de ODUs de seus clientes. Assim como no cliente, estas mensagens são tratadas assim que recebidas para evitar o atraso de sincronização.

O servidor gera três tipos de mensagens: de sincronização dos objetos ODS replicados, enviada de tempos em tempos, de notificação de antecipação de células, em resposta a um indício de movimentação de um ODU entre células, e de área de atuação, enviada a um cliente quando da entrada de um novo usuário no sistema.

3.2.6 Detecção e Resolução de Colisões

Conforme discutido na especificação do cliente, apenas a detecção e resolução de colisão dos objetos ODS é feita no servidor de ambiente, já que o servidor tem um controle de todos os objetos ODS em sua sub-área de responsabilidade.

3.2.7 Gerenciamento

As funções de gerenciamento desempenhadas pelo servidor de ambiente incluem: o controle dos clientes, como endereçamento e respectivas áreas de atuação; controle de entrada de novos clientes e a atualização desta entrada frente a outros clientes; controle da saída de clientes do sistema e respectiva atualização deste estado em outros clientes etc.

4. Simulação do Sistema

Um modelo de simulação de evento discreto está sendo utilizado para avaliar o sistema distribuído de realidade virtual especificado.

Como o sistema é bastante complexo, cliente e servidor de ambiente são modelados separadamente, considerando-se apenas as variáveis e as tarefas que causam maior impacto no desempenho do sistema. Desta forma, algumas das tarefas do cliente e servidor de ambiente não foram modeladas na simulação e serão objetos de análise futura. Estas tarefas incluem: a inicialização do servidor de ambiente, a movimentação de objetos ODS entre células e entre servidores de ambiente, a movimentação de objetos ODU entre servidores de ambiente, e o tratamento da colisão na máquina cliente. Estas tarefas entretanto, não devem causar maior impacto no desempenho do sistema uma vez que, no caso da inicialização do SA, não há ainda interatividade entre clientes e SA. O cruzamento de um objeto ODS entre sub-áreas, implica no envio, pelo SA atual, de uma mensagem contendo a base de dados deste objeto ao SA destino, que vai então tratar de disseminar este novo objeto aos seus clientes correspondentes. A movimentação de um objeto ODU para outro SA não deve ocorrer com frequência uma vez que, um usuário, ao selecionar sua área de atuação, tende a permanecer nela - a menos que o objeto ODU seja do tipo avião que pode cruzar entre sub-áreas com maior frequência - neste caso, as novas células têm que ser carregadas no cliente correspondente e o estado deste objeto ODU disseminado para outros clientes que contenham células coincidentes. Esta ação não difere muito da ação tomada quando do cruzamento de

objetos ODU entre células, onde são enviadas, pelo SA, mensagens de notificação de antecipação de células - o envio deste tipo de mensagem é modelado na simulação. Os dados e as expressões utilizadas na simulação do cliente e servidor de ambiente são mostrados nos apêndices A e B respectivamente.

4.1 O Modelo do Cliente

No modelo sendo considerado, o processo de *rendering* é executado por um *hardware* dedicado, não ocupando portanto tempo da CPU principal. A taxa analisada é de 10 quadros por segundo (um quadro a cada 10 ms), mínimo suficiente para manter a interatividade num nível de atraso aceitável. Desta forma, o processo de *rendering* é ativado, no cliente, a cada 10 ms.

Considerando que os sensores são lidos através de *hardware* próprio de interface, esta leitura pode ser feita em paralelo ao processamento destes dados pela CPU principal. Para que a CPU não fique incessantemente processando os dados lidos por esta interface de entrada, o processamento dos dados é ativado a cada 10 ms. Aqui, assim como no processamento de mensagens recebidas discutido na seção 3, pode ocorrer um atraso de sincronização no processamento dos dados lidos dos sensores. Desta forma, no melhor caso, os dados dos sensores são processados imediatamente após a última leitura dos sensores (com atrazo de zero ms) e, no pior caso, imediatamente antes da última leitura dos sensores (o que causaria um atraso de no máximo 10 ms). Se considerarmos que estes dados só serão refletidos na visualização final depois de processados pelo *rendering*, e que o *rendering* é ativado a cada 10 ms, podemos ter aqui um atraso total de até 20 ms (10 ms decorrentes do *atraso de sincronização do processamento dos dados dos sensores*, somados aos até 10 segundos do processamento do *rendering*, se este for processado imediatamente antes do término do processamento dos dados mais atuais dos sensores).

Taxa de Chegada e de Geração de Mensagens

A taxa de chegada das *mensagens de reposicionamento* é regida por uma distribuição exponencial com média de 5 segundos[6]. Desta forma, N-1 mensagens de clientes remotos são recebidas por um cliente, em média a cada 5 segundos (considerando um ambiente virtual sendo compartilhado por N usuários). O cliente, por sua vez, gera uma *mensagem de reposicionamento* do ODU de sua responsabilidade de acordo com uma distribuição exponencial com média de 5 segundos.

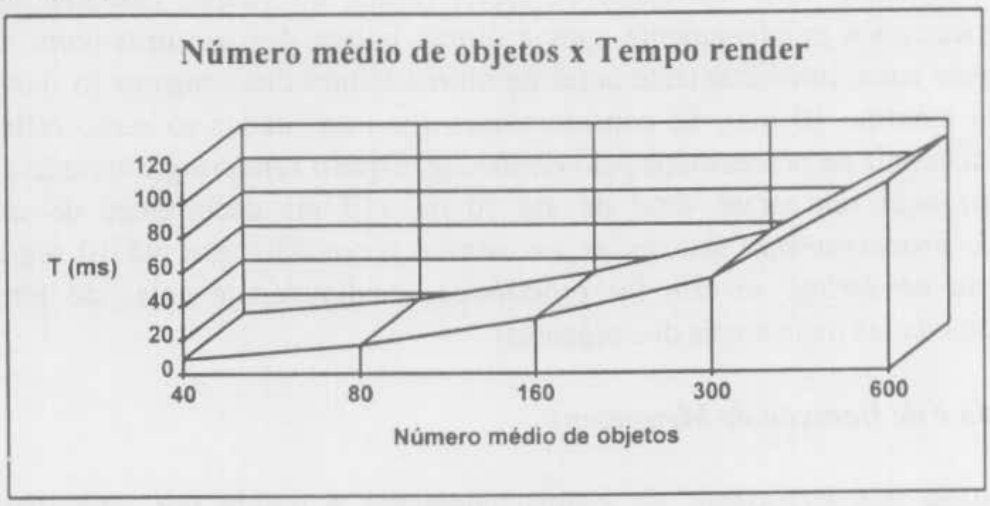
As *mensagens de sincronização*, enviadas pelo servidor de ambiente, são recebidas a cada 5 segundos. Já as *mensagens de notificação de antecipação de células* são recebidas a uma taxa dependente do tamanho da célula e da velocidade do ODU. Para o cenário de aplicações sendo considerado aqui, uma *mensagem de notificação* é recebida, em média (distribuição exponencial), a cada 72.5 segundos, tempo que um automóvel, a 50 Km por hora, leva para atravessar uma célula de 1 Km de diâmetro.

[6] Apesar dos movimentos de um ser humano serem contínuos, estamos considerando, de acordo com o cenário de aplicações descrito na seção 2, que o objeto ODU é um automóvel e que, portanto, um movimento inesperado a cada 5 segundos parece razoável.

4.2 Resultados que Impactam no Desempenho do Cliente

No sistema simulado, a maior fonte de atraso no cliente é o processo de *rendering*, que é altamente influenciado pelo número de objetos do sistema (mais pelo número de objetos fixos do que pelo de objetos dinâmicos, que são em menor número na área de visão de um usuário), bem como pelo número de polígonos de cada objeto fixo e dinâmico, como mostram os gráficos 1 e 2. No gráfico 1, a uma taxa de 0.1 quadros por segundo, perto de 600 objetos, com média de 50 polígonos cada, podem ser sintetizados em tempo. Já para taxas de 30 e 60 quadros por segundo, ou seja, um quadro a cada 0.033 e 0.017 segundos respectivamente, o número de objetos suportado é de 160 e 80 objetos respectivamente. Isto se traduz em um ambiente virtual distribuído com aparência de desenho animado bastante básico, longe do realismo. Estes valores vêm confirmar que o excitamento promovido pela mídia em torno de aplicações inusitadas de realidade virtual ainda está longe de acontecer, apesar da incessante pesquisa nas áreas relacionadas.

Gráfico 1



O resultado da variação do número de usuários remotos (representados por objetos ODU) suportados por um cliente, é mostrado no gráfico 3. Observa-se que o número de clientes pouco influencia no tempo de processamento do cliente, bem como na sintetização de imagens. Isto porque em termos de sintetização de imagem, o número de objetos ODU na área de visão do usuário é normalmente pequeno. O impacto maior do aumento do número de clientes é no processamento do *dead-reckoning* para cada objeto ODU remoto e suas atualizações na base de dados e no processamento de mensagens de reposicionamento de objetos ODU remotos recebidas pelo cliente. O gráfico mostra os valores para o melhor caso, caso médio e pior caso quando os dados se tornaram disponíveis e os processos de tratamento dos dados lidos dos sensores e sintetização tinham, respectivamente, sido iniciados naquele momento, já estavam na metade do processamento e quando já tinham iniciado imediatamente antes dos dados estarem disponíveis para serem processados.

Gráfico 2

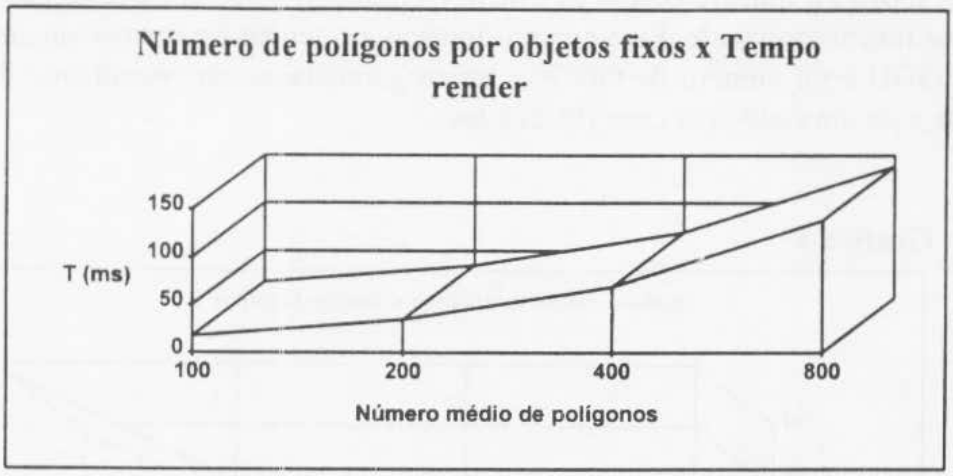
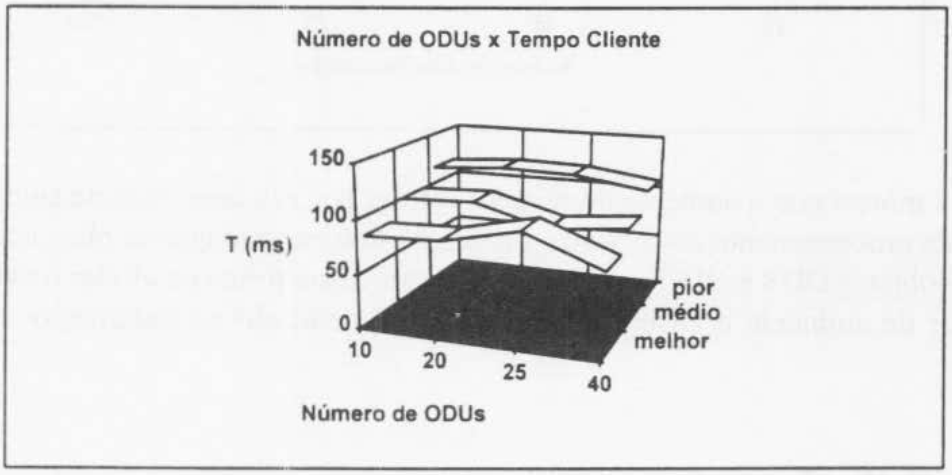


Gráfico 3



4.3 O Modelo do Servidor de Ambiente

No modelo sendo considerado, o servidor de ambiente recebe mensagens de reposicionamento de ODUs de todos clientes associadas a ele. Como cada cliente gera uma mensagem de reposicionamento, em média, a cada 5 segundos, considera-se que o servidor recebe N destas mensagens em 5 segundos, de acordo com uma distribuição exponencial (considerando-se um ambiente virtual compartilhado por N clientes).

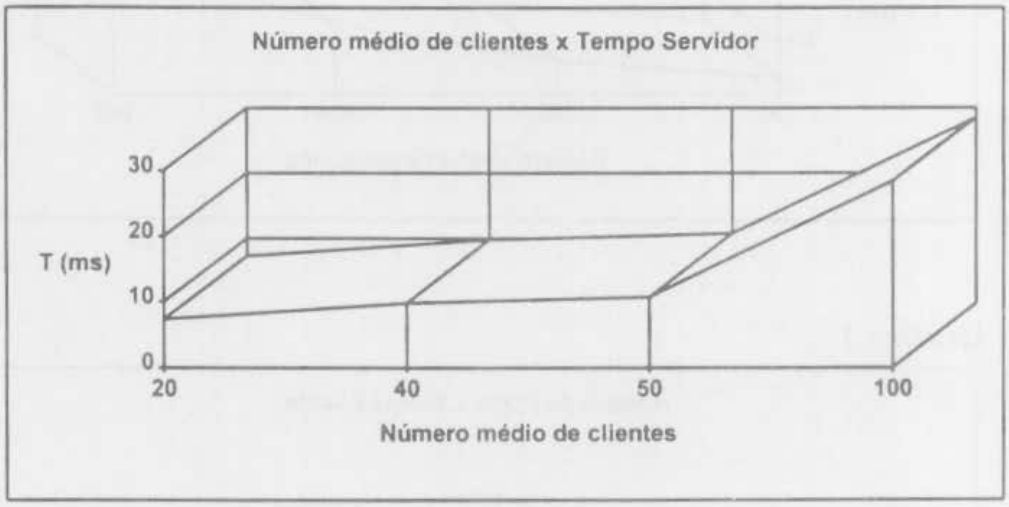
Por sua vez, o servidor gera três tipos de mensagens: de sincronização (uma a cada 5 segundos), M mensagens de notificação de antecipação de células (uma mensagem para um número médio de M clientes) a cada 72.5 segundos (vide sub seção 4.1) e uma mensagem de envio de área de atuação a cada 600 segundos (considerando-se que um novo usuário entra no sistema a acada 5 minutos).

Todo o processamento das tarefas do servidor é feito por apenas uma CPU principal.

4.4 Variáveis que Impactam no Desempenho do Servidor de Ambiente

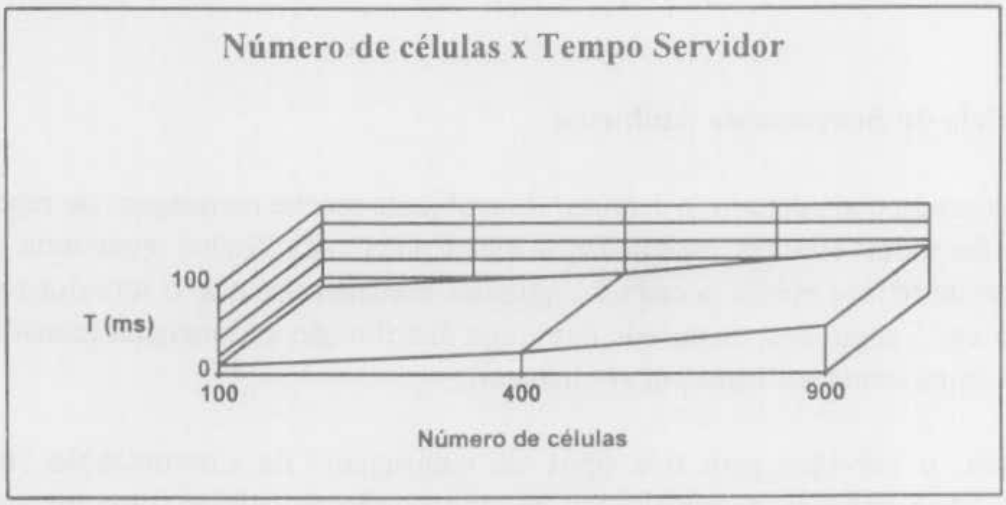
O gráfico 4 mostra o número médio de clientes suportado pelo servidor de ambiente na sub-área sob sua responsabilidade. Este número impacta no tempo de reposicionamento de todos os objetos ODU e no número de ODUs a serem gerenciados no cruzamento de células (foi considerada aqui uma sub-área com 100 células).

Gráfico 4



O gráfico 5 mostra que o aumento do número células por sub-área impacta significativamente no tempo de processamento do servidor, o que é de se esperar já que, implica num aumento do número de objetos ODS e ODU a serem gerenciados. Uma forma de aliviar o eventual gargalo no servidor de ambiente é atribuir um processador dedicado ao tratamento das mensagens recebidas.

Gráfico 5



Um resultado interessante é o desempenho do servidor em função do tipo de objeto que pode representar o usuário no sistema (ODU), isto é, o atraso frente a ODUs do tipo pedestre (percorre 1Km, isto é, cruza uma célula do ambiente virtual, em 20 minutos), automóvel (a 50

km por hora e portanto, percorre uma célula em 72.5 segundos) e avião (a 1000 km por hora, atravessando uma célula em 3.6 segundos). Este resultado ainda está sendo analizado, juntamente com o atraso causado pela rede.

4.5 Metodologia Utilizada na Simulação

O cliente é considerado um sistema terminante já que, inicia-se com a entrada do usuário no ambiente virtual desejado, e termina com a saída do usuário do ambiente virtual. Já o servidor de ambiente, que reflete um sistema *non-stop*, é tratado como não terminante. Desta forma, foram adotados para a coleta de dados da simulação do cliente e servidor de ambiente, respectivamente, o método das *Replicações*, que consiste numa série de "n" rodadas independentes (cada replicação utilizando uma nova sequência de números aleatórios e independentes - foram consideradas aqui 10 replicações, cada uma com tempo de 2400 segundos de simulação) e o de *Batch Means*, onde uma sequência de dados é dividida em lotes aproximadamente independentes (foi considerada aqui uma sequência de tamanho 24000, dividida em 10 lotes de 2400 segundos) [PEGDEN95].

4.6 Validação da Simulação

O processo de validação de uma simulação é um processo que exige cuidado, especialmente se não houver na literatura sistemas similares. Os atrasos médios para a aplicação encontrados durante o processo de simulação, estão dentro da faixa de atrasos encontrada na literatura [WLOKA95], que sugerem de 10 a 120 ms para a leitura de sensores, 1 a 500 ms para a aplicação, 14 a 100 ms para o render, além de um total de 0 a 633ms para o atraso de sincronização.

5. Trabalhos Relacionados

Mathias Wloka [WLOKA95] faz uma análise do atraso em um sistema de realidade virtual simples, com dados estáticos, o que apesar de dar uma idéia interessante sobre os mecanismos de sincronização dos estágios de processamento e seus impactos no atraso final do sistema, pode não refletir a realidade quando considerados a distribuição do sistema, dispositivos de entrada com desempenho diferente do utilizado, maior complexidade da aplicação utilizada, número de objetos no sistema etc. Berger e colegas [BERG93] fazem uma comparação de arquiteturas de RV em rede, com considerações bastante restritas, o que torna difícil a extrapolação dos resultados para uma aplicação genérica. Em termos de número de usuários suportados, os resultados, apesar de preliminares, mostram superioridade em relação a sistemas de RV distribuídos como o MASSIVE [BENF94], que relata um número máximo de seis usuários compartilhando um mesmo ambiente virtual simultaneamente, contra mais de 40 usuários dentro da faixa de 100 ms aqui relatados, conforme gráfico 3. Uma excessão é feita ao sistema NPSNET-IV [MACE94], cuja escalabilidade é de centenas de usuários - estes números entretanto, foram obtidos com uma versão que mantinha toda a base de dados em cada nó do sistema. Uma análise de desempenho da nova versão do sistema NPSNET [MACE95], que distribui os dados, não é de conhecimento dos autores até o momento da escrita deste artigo.

6. Conclusão

Este artigo descreveu a arquitetura, especificação e análise de um sistema distribuído de realidade virtual. Os resultados, apesar de preliminares, mostram a viabilidade do sistema no suporte a um número relativamente alto de usuários. Uma análise mais detalhada do sistema, um tratamento estatístico mais refinado dos dados, além do incremento da complexidade do modelo e do tratamento do atraso na transmissão de mensagens entre os elementos do sistema pela rede estão sendo desenvolvidos no momento como parte da tese de doutorado do primeiro autor.

7. Bibliografia

[ARAU95a] Araujo, R. B. and Kirner, C. - Arquitetura de um Ambiente Virtual Distribuído para Aplicações de Larga Escala, XIII SBRC, Belo Horizonte, Maio 1995.

[ARAU95b] Araujo, R. B. and Kirner, C. - Network Requirements for Large Scale Virtual Environment, seção do livro Virtual Prototyping - IFIP, Chapman & Hall, 1995.

[BENF94] Benford, S. et al. Managing Mutual Awareness in Collaborative Virtual Environments. Virtual Reality Software & Technology, Proceedings of the VRST'94 Conference, Singapore, 23-26 August, p.223-236, 1994

[CALV93] Calvin, J. et. alii. - The SIMNET Virtual World Architecture, IEEE Virtual Reality Annual International Symposium, September 18-225, Seattle, WA, 1993, pp. 450-455.

[MACE94] Macedonia, M. et al. - NPSNET: A Network Software Architecture for Large Scale Virtual Environments, Presence Magazine, Vol.3 No.4, 1994.

[MACE95] Macedonia, M. et al. Exploiting Reality with Multicast Groups. IEEE Computer Graphics and Applications, vol.15, no.9, p.38-45, 1995.

[PEGD95] Pegden, C.D. et ali - Introduction to Simulation Using Siman, 2nd. Edition, McGraw Hill, 1995.

[ROBE93] Roberts, D.J. et al. LOKI-2 - An Architecture for Distributed Virtual Reality, Colloquium on Distributed Virtual Reality - Digest no. 1993/121, London, UK, 1993.

[WLOK95] Wloka, M. - Lag in Multiprocessor VR, Presence Magazine, Vol. 4, No.1, Winter 1995.

APÊNDICE A

Variável	*Valor no Cliente*	*Valor no Servidor Ambiente*
NCellAVisao	4	4
NDispoSensores	3	
NivelAntecip	1	
NODUAVisao	1	-
NODURem	3	-
NumCellsSA	-	100
TDetermCells	-	10-6
TDeadreck	2x10-6	2x10-6
TPrepMsgACK	2x10-5	2x10-5
TPrepMsgNotif	-	18x10-5
TPrepMsgRepos	4x10-5	-
TPrepMsgSync	-	2x10-5
TProcMsgAck	-	2x10-5
TProcMsgRepos	4x10-5	4x10-5
TProcMsgSync	2x10-5	-
TProcMsgNotif	18x10-5	-
TRenderUmPoly	10-6	-
TRxSystemCall	10-6	10-6
TTxSystemCall	10-6	10-6
NMObjFixos	Distr.Triangular (20,40,60)	-
NMODS (por celula)	Distr.Triangular (2,3,7)	Distr.Triangular (2,3,7)
NMODU	-	Distr.Triangular (30,70,80)
NMPolyObjDinam	Distr.Triangular (40,50,60)	-
NMPolyObjFixo	Distr.Triangular (40,50,60)	-
TMAnimODS	Distr.Triangular (10-5,10-4,2x10-4)	Distr.Uniforme (10-6,2x10-6)
TMLeSensor	Distr.Uniforme (10-2,12x10-2)	-
TMProcSensor	Distr.Uniforme (10-6,2x10-6)	-
TMUpdRepos	Distr.Triangular (10-6,4x10-6,2x10-6)	Distr.Triangular (10-6,4x10-6,2x10-6)

Tabela 1. Dados Utilizados na Simulação do Cliente e Servidor de Ambiente

Para os valores da tabela acima, foi considerada a máquina INDIGO Maximum Impact da Silicon Graphics, com estimativa de tempos de instrução de máquina variando de 30 a 100 ns. O tempo médio de leitura de um dispositivo de sensor sendo considerado (TMLeSensor) é o descrito na literatura [WLOKA95]. Considerando as características peculiares ao sistema distribuído especificado, a maioria dos valores foram estimados já que não puderam ser encontrados na literatura.

APÊNDICE B

EXPRESSÕES DO CLIENTE

```
TreatSyncExpr = TRxSystemCall+TProcMsgSync+TMUpdRepos
TreatReposExpr = TRxSystemCall+TProcMsgRepos+TMUpdRepos
TreatNotifExpr = TRxSystemCall+TProcMsgNotif+TProcACK+
                                TTxSystemCall
ProcSensorExpr = NDispoSensores*(TMProcSensor+TDeadreck+
                                TMUpdRepos)
PosicODSExpr = (TMAnimODSC+TMUpdRepos)*NMODS*
                                (NCellAA+NivelAntecip*NCellBorda)
PosicODURemExpr = (TDeadreck+TMUpdRepos)*NMODURem
PrepReposExpr = TPrepMsgRepos+TTxSystemCall
RenderExpr = TrenderUmPoly*(NMPolyObjDinam*(NMODS*
                                NCellAVisao+NODUAVisao) +
                                NMPolyObjFixo * NMObjFixos*
                                NCellAVisao);
```

EXPRESSÕES DO SERVIDOR

```
TreatReposExpr = TProcMsgRepos+TRxSystemCall+TMUpdRepos
PosicODSExpr = (TMAnimODSS+TMUpdRepos)*NMODS*
                                NumCellsSA
PosicODUExpr = (TDeadreck+TMUpdRepos)*NMODU
PrepSyncExpr =  TPrepMsgSync+TTxSystemCall
PrepNotiExpr = (TDetermCells+TPrepMsgNotif*NivelAntecip+
                                TPrepODS*NMODS*3*NivelAntecip+
                                TTxSystemCall+TProcMsgAck+
                                TRxSystemCall)*NMODU*0.3
PrepInitCLExpr = TPrepMsgInit+TTxSystemCall
```