Projeto do Framework de Simulação Simmcast: uma arquitetura em camadas com ênfase na extensibilidade*

Marinho P. Barcellos **Hisham H. Muhammad**
Luís Felipe Cintra **Giovani Facchini**

[1]PIPCA - Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Unisinos - Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 - São Leopoldo, RS - 93022-000

{marinho,hisham,cintra,giovani}@exatas.unisinos.br

Resumo. *O Simmcast é um framework de simulação orientado a objetos, para pesquisa de protocolos de rede, que emprega um modelo de eventos discretos baseado em processos, no qual blocos básicos são combinados e estendidos a fim de criar novos ambientes de simulação. Trabalhos anteriores sobre o Simmcast descrevem seus princípios básicos e abordam aspectos específicos da ferramenta. Neste artigo, é apresentada uma visão geral da arquitetura do Simmcast em função de suas camadas, sendo as mesmas também usadas na análise do histórico evolutivo do projeto. Discutem-se aqui desde as tecnologias sobre as quais o Simmcast é baseado, até como se dá a interação de um protocolo desenvolvido pelo usuário com a infra-estrutura disponibilizada pela ferramenta.*

Abstract. *Simmcast is an object-oriented simulation framework for network protocol research that employs a discrete event model based on processes in which building blocks are combined and extended to create new simulation environments. Previous papers about Simmcast have described its basic principles and addressed specific aspects of the tool. This paper presents a general view of the architecture of Simmcast in function of its layers, and these are used to analyse the evolving history of the project. It includes the discussion of technologies in which Simmcast is based as well as the interaction of a user-developed protocol with the infra-structure made available by the tool.*

1. Introdução

Protocolos regem a interação entre dois ou mais elementos que, tipicamente, residem em computadores diferentes e que, pelo menos do ponto de vista do hardware, operam de forma autônoma. A natureza inerentemente concorrente desse tipo de software e as incertezas decorrentes de uma rede assíncrona (variações na velocidade de comunicação e dos processadores, ocorrência de falhas, etc.) representam uma dificuldade significativa no desenvolvimento desse tipo de software. Simulação discreta é uma técnica bastante útil na investigação (no senso mais abrangente da palavra) de protocolos existentes, e tem sido usada como ferramenta auxiliar na construção de novos protocolos.

Tipicamente, um projetista possui duas opções ao simular um novo protocolo: construir sua simulação com base em uma linguagem específica (como Simula [5]), ou então estender um simulador de redes (como o NS-2 [13]) com seu novo protocolo. Um *framework* de

*XXII Simpósio Brasileiro de Redes de Computadores - SBRC2004 (Salão de Ferramentas) - Vol.2, pp.951-958 - ISBN 85-88442-79-5

simulação potencialmente combina vantagens destas duas abordagens, pois simulações são desenvolvidas especializando os componentes disponibilizados com a funcionalidade específica para o problema a ser tratado, ao mesmo tempo em que mantém a modularidade entre a base de código oferecida e a desenvolvida pelo usuário. O Simmcast é um *framework* de simulação orientado a objetos para pesquisa de protocolos de rede, que emprega um modelo de eventos discretos baseado em processos, no qual blocos básicos (*building blocks*) são combinados e estendidos a fim de criar novos ambientes de simulação.

Trabalhos anteriores sobre o Simmcast ([3, 11]) descrevem seus princípios básicos, e justificam o emprego de um *framework* de simulação. Em outros trabalhos ([12, 10]), foram abordados aspectos específicos da ferramenta, como mecanismos de roteamento multicast. Neste artigo, é apresentada a arquitetura geral da ferramenta em função de suas camadas, sendo as mesmas também usadas na análise do histórico evolutivo do projeto. O restante do trabalho está organizado da seguinte maneira: a Seção 2 apresenta a arquitetura em camadas; a Seção 3 apresenta a evolução histórica do Simmcast; a Seção 4 revisa o Testbed, uma evolução recente do Simmcast que "aproxima" a simulação de protocolos com sua avaliação experimental. A Seção 5 descreve trabalhos em andamento, enquanto comentários finais são apresentados na Seção 6.

2. Arquitetura em Camadas

A divisão de um sistema em camadas (*layering*) tem sido usada por muitos anos como estratagema de projeto para redução da complexidade: identificação de camadas, separação dos componentes nas mesmas e especificação das interfaces de interação entre cada um dos componentes. O Simmcast é um *framework* de simulação cuja construção se baseia conceitualmente em uma série de camadas. Cada camada é desenvolvida de forma independente das superiores, fazendo o uso de serviços oferecidos pelas inferiores. O conjunto de camadas é descrito a seguir.

Camada 0: Java. O suporte a *threads* proporcionado pela API de Java serve como base para o modelo de simulação de processos implementado no Simmcast. Um dos princípios de projeto do Simmcast é proporcionar, nos níveis superiores, onde ocorre o *input* do usuário da ferramenta, modelos de programação de comunicação e *threads* realísticos. Basear o simulador em uma API de *threads* real (os processos instanciados pelo usuário são subclasses de `java.lang.Thread`) mantém o estilo de código desenvolvido pelo usuário realístico. Isto faz com que seja possível, caso desejado, uma correspondência próxima entre o código escrito para execução no simulador e aquele escrito para execução real sobre uma rede. De fato, um módulo do Simmcast foi desenvolvido para automatizar esta transição (vide Seção 4).

Camada 1: Simmcast Engine. A *Simmcast Engine* é uma máquina de eventos discretos baseada em processos. Esta máquina é totalmente genérica, baseada unicamente no conceito de entidades fundamentais, denominadas **processos**, que se sincronizam através de comandos do tipo "sleep" e "wake up". Não há na *Simmcast Engine* qualquer referência específica à simulação de redes. Todo o suporte para modelagem de dispositivos se comunicando via rede ocorre nas camadas superiores do simulador. Assim, a *Simmcast Engine* pode ser reaproveitada no desenvolvimento de simuladores para outras finalidades, como por exemplo processamento paralelo.

Cada processo na *Simmcast Engine* é mapeado em uma *thread* Java. O comportamento das *threads*, entretanto, é estritamente controlado pela *engine*, de modo a liberar execução

de apenas uma *thread* por vez, de forma que o escalonamento seja não-preemptivo. Isto é necessário para que o modelo de execução seja determinístico: duas execuções de uma simulação, tendo o mesmo conjunto de entradas, deve levar ao mesmo conjunto de saídas (resultados reproduzíveis).

Camada 2: *Kernel* do simulador. Na camada 2 são definidas as entidades fundamentais da simulação. Esta é a primeira camada visível ao usuário do Simmcast ao desenvolver um experimento utilizando a ferramenta. A entidade fundamental da simulação, o **nodo** (Node), é definida nesta camada. Um nodo é uma unidade capaz de enviar e receber **pacotes** (Packets), composta de uma ou mais *threads* de execução[1]. Cada nodo possui ainda uma *thread* interna de escalonamento, responsável pelo controle das filas de pacotes e por eventos assíncronos. Os nodos são conectados entre si através de uma entidade de alto nível denominada **caminho** (Path), que possui como atributos largura de banda, latência de propagação e taxa de descarte; estes atributos podem ser modelados probabilisticamente, de forma a simular o comportamento de um enlace ou de um conjunto de enlaces.

Nesta camada é oferecido ainda suporte à comunicação 1-para-*N* de forma abstrata, utilizando um objeto **grupo** (Group). Combinando nodos e caminhos, é possível descrever uma "topologia lógica" através de um grafo conectado, onde cada nodo representa um processo participante do sistema, protocolo ou aplicação do usuário. Neste caso, os caminhos irão determinar as características de comunicação entre dois nodos, sendo a rede modelada como uma camada única ou "nuvem". Por exemplo, um caminho com perda 0 e atraso variável pode ser usado para emular uma conexão TCP entre dois nodos, permitindo descrever um sistema distribuído onde nodos encontram-se conectados via TCP.

Camada 3: Roteamento. Simulações com maior grau de detalhamento necessitam redes onde o grafo da topologia representa uma organização física, ou seja, onde elementos de rede (como roteadores e *switches*) são incluídos no modelo e há uma distinção entre "elementos fim" (estações) e elementos internos da rede (roteadores). Com a presença de uma estrutura de rede (conjunto de roteadores interconectados), é necessário um suporte subjacente que execute o roteamento. Utilizando unicamente as entidades definidas na camada 2, implementou-se no Simmcast um subsistema de roteamento que pode ser integrado a uma simulação de forma independente ao código desenvolvido pelo usuário nas camadas superiores. Este subsistema permite que um protocolo da camada 4 ou superior seja testado com diferentes algoritmos de roteamento, incluindo o impacto de mudanças dinâmicas de rotas e sobrecarga de mensagens de roteamento. Além disso, o subsistema inclui um roteamento de menor caminho, onde cada nodo conhece a rede como um todo e aprende instantaneamente sobre alterações na mesma. A arquitetura para simulação de roteamento é discutida em maior detalhe em [12].

Conforme acima, nesta camada os nodos são distintos entre **roteadores** (RouterNodes) e **estações** (HostNodes), ambos subclasses de Node. As classes de estações são meramente invólucros para conectar o código das camadas superiores ao subsistema de roteamento. Os roteadores possuem um modelo de *threads* pré-definido, DefaultRouterNode, embora outros possam ser desenvolvidos pelo usuário. Um novo algoritmo de roteamento é implementado como uma nova subclasse de RoutingAlgorithmStrategy, contendo apenas a lógica do protocolo. Portanto, não há necessidade de incluir o controle de encaminhamento de pacotes, pois este é implementado nas *threads* de DefaultRouterNode.

Camada 4: Protocolo. De acordo com o projeto de um *framework*, o Simmcast é

[1]Cada *thread* é implementada internamente como um processo da camada 1.

distribuído como um conjunto de classes que implementa a funcionalidade básica necessária para a sua operação. Estas classes oferecem pontos de extensibilidade bem definidos, para que o usuário adicione o seu código e gere assim o produto final, isto é, um experimento de simulação. É nesta camada conceitual que o usuário adiciona a lógica do protocolo (ou protocolos) que irá(ão) ser executado(s) no experimento. Assim como numa rede real, protocolos são implementados como processos que executam nas estações da rede; no simulador, protocolos são implementados como *threads* que executam nos nodos. Um protocolo de nível de transporte é desenvolvido pelo usuário então como uma ou mais subclasses de NodeThread, que implementam a lógica do protocolo em questão e que são inseridas em um Node (ou em um HostNode, quando empregando o subsistema da camada 3).

É importante ressaltar a independência entre as camadas 2, 3 e 4, e que a interação entre elas deve se dar unicamente por mecanismos de herança e composição de classes. Assim, garante-se que diferentes implementações desenvolvidas em cada camada se mantenham intercambiáveis sem a necessidade de re-editar código. Na última camada (6), é oferecida uma maneira de montar uma simulação com diferentes componentes sem a necessidade de recompilação.

Camada 5: Aplicação. Em termos de facilidades oferecidas pelas camadas inferiores do Simmcast, esta camada não difere em muito da anterior: uma aplicação é desenvolvida igualmente como um conjunto de *threads* inserido nos nodos, gerando envios e recebimentos de pacotes, que podem passar ou não por um subsistema de roteamento. Entretanto, é interessante encarar conceitualmente esta camada de forma isolada da camada 4, para que os protocolos de transporte desenvolvidos naquela camada possam ser utilizados nas aplicações de forma *black-box*, como uma biblioteca de protocolos. Na Seção 3, são discutidos protocolos da camada 4 em desenvolvimento a serem distribuídos juntamente com o *kernel* do Simmcast.

Camada 6: Configuração da simulação. Finalmente, o Simmcast oferece um mecanismo que permite que, através do suporte à carga de classes oferecido pela linguagem Java, um experimento seja instanciado de forma dinâmica. Com isso, a parametrização das entidades e a composição da rede se dá através de um arquivo-texto de configuração, dispensando a necessidade de recompilar o código. Neste arquivo, objetos das camadas 3, 4 e 5 podem ser instanciados e manipulados, bem como parâmetros nestes objetos herdados da camada 2.

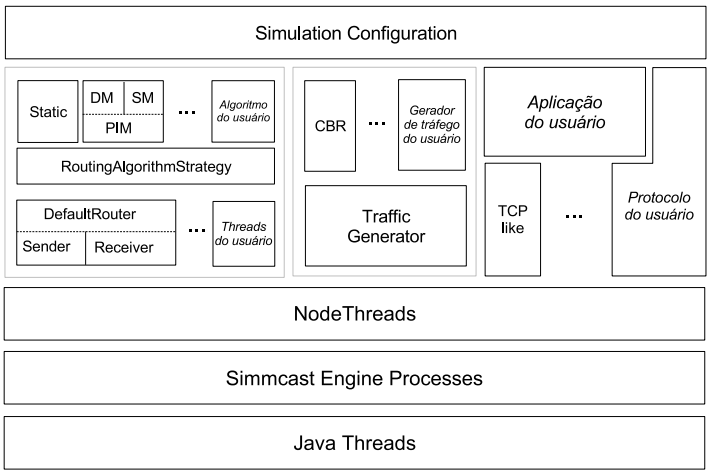

Figura 1: Arquitetura das unidades de execução do Simmcast

Em função do conjunto de camadas acima, a arquitetura do Simmcast pode ser repre-

sentada conforme a Figura 1.

3. Evolução Histórica do Simmcast

Embora formalmente criado em 2002, o Simmcast é resultado de uma década de pesquisa. Em termos históricos, ele é originário de um projeto na Unisinos - Universidade do Vale do Rio dos Sinos, desenvolvido entre 1999 e 2002 com apoio do CNPq e da FAPERGS, e que por sua vez representou a continuação de uma investigação sobre protocolos para multicast confiável 1-*N* iniciada em 1994, em Newcastle upon Tyne ([2]). Os resultados desta última pesquisa, inicialmente reportados em[1], foram obtidos através de simulação. Visando uma avaliação mais detalhada de um protocolo (PRMP), em [2] esta simulação foi redesenhada com base em um modelo *produtor-consumidor.* O modelo adotado tinha como componentes fundamentais de construção *threads* e filas (de mensagens e de eventos); sua hierarquia de classes incluia *hosts* e *routers* (herdeiros de *node*), e *links* com banda fixa e latência e taxa de erro aleatórias. Tal simulação foi construída em C++, com base no pacote C++SIM ([7]). Em 2000, como parte do projeto PRMP na Unisinos, ao avaliar-se mecanismos para protocolos multicast baseados em *polling*, aquele modelo de rede foi reprojetado, levando à criação de um *framework* de simulação. Os resultados iniciais deste esforço foram tão positivos (vide [3, 11]) que levaram à criação de um projeto de pesquisa específico, primeiro na Unisinos, e depois junto ao CNPq.

Em termos arquiteturais, a evolução do Simmcast começou com o desenvolvimento do *kernel* do simulador (camada 2), utilizando como camada 1 a *engine* genérica de simulação JavaSim ([8]). Esta última, um porte para Java do pacote C++SIM, implementa como abstração uma máquina de eventos discretos orientada a processos, e baseada no modelo de *threads* da linguagem Java. Neste estágio, foram definidos, no *kernel*, os blocos básicos fundamentais, como **nodo** e **thread**, a partir dos quais as camadas superiores são baseadas.

As aplicações inicialmente desenvolvidas utilizando o simulador foram simulações de mecanismos para protocolos multicast para transmissão 1-N confiável. Estes experimentos utilizavam modelos de rede simplificados, sobre uma topologia abstrata, aproveitando o suporte à definição de caminhos a partir de modelos probabilísticos. Nestes protocolos, não havia ainda diferenciação que hoje há entre as camadas 4 e 5 do simulador (respectivamente, "protocolo" e "aplicação"), sendo a geração de tráfego integrada diretamente ao código dos protocolos.

Como passo seguinte, foi desenvolvido um subsistema de roteamento utilizando os blocos básicos da camada 2 ("kernel"). A arquitetura de roteamento foi desenvolvida visando **extensibilidade**, isolando o modelo de *threads* empregado por um nodo roteador (RouterNode) das classes que implementam os algoritmos de roteamento em si (RoutingAlgorithmStrategy). Desta forma, pôde-se definir uma camada de roteamento independente que pode ser inserida transparentemente entre os protocolos implementados e o *kernel* do simulador. Assim, embora as camadas 3 ("roteamento") e 4 ("protocolo") sejam igualmente desenvolvidas a partir dos componentes da camada 2, em tempo de execução o protocolo da camada 4 executa "sobre" o protocolo da camada 3.

Recentemente, a camada 1 da arquitetura foi substituída, removendo-se o JavaSim em favor de uma implementação própria de uma máquina de simulação, denominada *Simmcast Engine*. Comparativamente, a *Simmcast Engine* apresenta maior desempenho e melhor compatibilidade com versões recentes da Java Virtual Machine[2]. Além disso, é distribuída sob

[2]Atualmente, o requisito para compilação do Simmcast é a JDK 1.4. Versões de desenvolvimento já foram

a licença GNU GPL, permitindo a distribuição de todas as camadas do Simmcast em um pacote único.

Atualmente, o pacote do Simmcast é distribuído contendo as implementações das camadas 1, 2 e 3, ou seja, a *Simmcast Engine*, o *kernel* do simulador e o subsistema de roteamento, incluindo algumas implementações de protocolos desta camada. Novos protocolos de transporte podem ser desenvolvidos utilizando as facilidades oferecidas pela camada 2, produzindo um novo *building block* (bloco básico) independente na camada 4. Esse novo bloco pode ser re-utilizado em diferentes aplicações. De fato, encontra-se em andamento um trabalho que objetiva incluir na distribuição do Simmcast *building blocks* para a camada 4, isto é, implementações de protocolos de nível de transporte.

Para o desenvolvimento de protocolos nesta camada, inicialmente partiu-se de um conjunto de modelos abstratos de protocolos para comunicação confiável, baseados em [6]. A família de protocolos é denominada "RDT", de *Reliable Data Transfer*; cada versão incrementa sobre as anteriores em funcionalidade, e/ou reduz o conjunto de premissas adotadas sobre o canal de comunicação subjacente. Este tipo de "processo evolutivo" no projeto de um protocolo é expresso de forma natural no Simmcast, e é um dos claros benefícios da filosofia de *framework* adotada pelo mesmo.

No momento atual, está sendo implementado o mecanismo de controle de erro GBN (*Go-Back-N*) no qual o protocolo TCP (*Transmission Control Protocol*) é baseado. O GBN é um aprimoramento em relação aos modelos RDT, pois nele já há o prinicípio de *janela deslizante*. Seguindo a filosofia do Simmcast, a implementação contempla um conjunto de *threads*, como segue. A aplicação (geração de tráfego e *sink*) são implementados através de *threads*; seu funcionamento é simples, e consiste em enviar e receber (transmissor e receptor, respectivamente) mensagens de uma *thread* local do protocolo GBN. Este, por sua vez, utiliza duas *threads* no transmissor e uma no receptor: no primeiro caso, uma *thread* remetente envia pacotes e configura *timeouts* caso necessário, enquanto outra recebe *acks*, ajusta a janela e suspende *timeouts*, ou retransmite e reseta *timeouts* quando os mesmos expiram.

4. Simmcast Testbed

Um dos objetivos do modelo de programação desenvolvido na camada 2 do Simmcast foi assemelhar-se à API de programação de redes e à concorrência da linguagem Java. Além de dotar o programador de um ambiente de programação familiar, este princípio de projeto permitiu recentemente o desenvolvimento do Simmcast Testbed. Este último possibilita, através de uma API "intermediária", a execução de experimentos tanto no simulador como sobre a rede real, utilizando uma base de código única.

O Simmcast Testbed é uma interface de programação para a linguagem Java, que isola o programador tanto das peculiaridades do ambiente de simulação como dos detalhes das bibliotecas de rede. Esta API é composta por dois *back-ends*: *Testbed-Sim*, baseado nas camadas 1 e 2 do Simmcast; e *Testbed-Exp*, baseado diretamente na API de redes da plataforma Java.

É importante frisar que o *Testbed-Sim* utiliza o Simmcast tão somente como substrato de suporte em sua execução, encapsulando o simulador como um todo. Em outras palavras, ao utilizar o Simmcast Testbed, o programador deve fazer uso apenas das facilidades providas por esta API, não devendo incluir diretamente outras classes do Simmcast. Isto é importante

adaptadas para execução nativa utilizando o GCJ (GNU Compiler for Java).

para que o código desenvolvido mantenha a portabilidade entre as versões *Testbed-Sim* e *Testbed-Exp*, permitindo que um experimento seja migrado do simulador para a rede e vice-versa sem a necessidade de reimplementação.

O fato de que o Simmcast é construído como uma série de módulos com interações bem-definidas foi fundamental para o desenvolvimento de uma ferramenta como o Simmcast Testbed, uma vez que isto permitiu integrar um subconjunto de sua estrutura (camadas 0, 1 e 2) ao *back-end*, e utilizar as demais camadas de forma independente apenas no modo de execução simulado (por exemplo, modelando o roteamento similar ao disponível à rede real utilizando serviços das camadas 3 e 6). Desta forma, a arquitetura do Simmcast Testbed se mostra como um estudo de caso das possibilidades trazidas pelo *design* modular do Simmcast.

5. Trabalhos em Andamento

Uma vez desenvolvidas as camadas fundamentais do simulador, o projeto Simmcast prossegue com uma série de trabalhos, no sentido de desenvolver novas aplicações para o *framework* de simulação e agregar ao conjunto de blocos básicos novos protocolos e recursos, a fim de prover novas facilidades aos usuários e permitir a realização de novas classes de experimentos.

Um dos trabalhos em desenvolvimento atualmente é uma arquitetura de suporte à simulação de falhas em experimentos. O objetivo deste trabalho é permitir a simulação de diversos tipos de falhas, das classes omissivas, assertivas e arbitrárias, servindo como uma ferramenta de auxílio no desenvolvimento de protocolos e sistemas distribuídos tolerantes a falhas. A arquitetura é extendida através da modelagem de falhas nos blocos *nodo*, *caminho* e *roteador*, inserindo-se novos pontos de extensibilidade nas camadas 2 e 3. Estes pontos de extensibilidade relativos ao controle de falhas são transparentes às camadas de protocolo e aplicação, sendo manipulados diretamente na configuração do experimento (camada 6), afetando então os resultados produzidos pelas camadas 4 e 5, devido às falhas geradas. Entre os exemplos de intervenções possibilitados pelo modelo de falhas estão a duplicação de mensagens, o colapso de nodos e a queda de caminhos (enlaces ou conjunto de enlaces).

O Simmcast vem sendo também utilizado como ferramenta para avaliar uma simulação de jogos on-line. Para isto são avaliados quatro modelos, dois centralizados (com servidor) e dois descentralizados [4]. Entre os parâmetros avaliados nos experimentos estão consumo de banda e atraso médio de pacotes e o impacto decorrente na interatividade. Junto com uma avaliação experimental, serão comparados os resultados obtidos perante os aferidos em um experimento equivalente sobre um ambiente de rede real ([9]). Este trabalho vem sendo realizado utilizando o Simmcast Testbed, ilustrando como se dá a dualidade de execução, utilizando o mesmo código sobre a rede real e no ambiente do simulador, com uma camada única de abstração.

6. Conclusão

O Simmcast é um projeto de pesquisa para o desenvolvimento de uma infra-estrutura para simulação de redes utilizando uma arquitetura orientada a objetos. Partindo de uma estrutura de *framework*, foi possível desenvolver o *kernel* do simulador de forma modular, e a partir deste, prosseguir para o desenvolvimento de módulos que especializam o uso do simulador como ferramenta de pesquisa em diversas áreas, de comunicação em grupo a tolerância a falhas, passando por protocolos de roteamento e multicast.

Este artigo apresentou uma visão geral da arquitetura do Simmcast, descrevendo os seus diversos componentes utilizando um modelo conceitual em camadas. Apresentou-se, através desse modelo, as diversas fases do desenvolvimento da ferramenta, bem como o contexto onde se encaixam os diversos trabalhos do projeto como um todo. Além disso, foram enumerados alguns dos avanços recentes do projeto Simmcast neste último ano bem como as perspectivas de trabalhos em andamento.

O Simmcast é uma ferramenta já madura e utilizada ativamente. Ela está disponível no site do projeto, `http://inf.unisinos.br/~simmcast`, acompanhada de documentação completa de suas APIs. É disponibilizada no site também uma série de exemplos de uso, indo desde tutoriais básicos até implementações de algoritmos distribuídos clássicos.

Referências

[1] Barcellos. M. P., Ezhilchelvan, P. "A Reliable Multicast Protocol using Polling for Scaleability", In IEEE INFOCOM'98 - The Conference on Computer Communications, *Proceedings*, IEEE (New York), São Francisco, 29 de março a 2 de abril de 1998.

[2] Barcellos, M. P. *PRMP: A Scaleable Polling-based Reliable Multicast protocol.* Ph.D. Thesis, Department of Computing Science, Newcastle University, Newcastle upon Tyne, October 1998, 200p.

[3] Barcellos, M. P., Muhammad, H. H., Detsch, A. (2001) "Simmcast: a Simulation Tool for Multicast Protocol Evaluation", In XIX Simpósio Brasileiro de Redes de Computadores (SBRC 2001), *Anais,* Maio.

[4] Bedin, G. B., Muhammad, H. H., Detsch, A., Barcellos, M. P. "Generalized Models of Network Architectures for Online Games", In I Workshop em Jogos (WJogos 2002)*, Anais,* SBC, Fortaleza, Outubro 2002.

[5] G. Birtwistle, O-J. Dahl, B. Myhrhaug, and K. Nygaard, "Simula begin", Studentlitteratur, Lund, Sweden, 1973.

[6] Kurose, J. F., Ross, K. W. "Computer Networking: a top-down approach featuring the Internet". Addison-Wesley, 2001.

[7] Little, M. C., D. L. McCue, "Construction and Use of a Simulation Package in C++," *Computing Science Technical Report*, University of Newcastle upon Tyne, Number 437, Julho 1993 (também no *C User's Journal* Vol. 12 Number 3, March 1994).

[8] M. C. Little, "JavaSim Users Guide", Public Release 0.3, Version 1.0, `http://javasim.ncl.ac.uk`, disponível em Janeiro de 2003.

[9] Metropoa-2, Site do Projeto, em `http://www.metropoa.tche.br/`. Acessado em Jan. de 2004.

[10] Muhammad, H. H., Barcellos, M. P. "Protocol Simulation with the Simmcast Framework". In XXI Simpósio Brasileiro de Redes de Computadores (SBRC2003), *Anais, Salão de Ferramentas*, SBC, Natal, Maio 2003, pp.889-896. ISBN 85-88442-48-5

[11] Muhammad, H. H., Barcellos, M. P. "Simulation Group Communication Protocols Through an Object-Oriented Framework", in *Proceedings of the 35th SCS Annual Simulation Symposium*, San Diego, Abril 2003.

[12] Muhammad, H. H., Barcellos, M. P. , Casais. R. "Simulação de Roteamento na Avaliação de Protocolos Multicast e Sistemas Distribuídos em Grupo", Wperformance 2002, *Anais*, SBC, Florianópolis, Julho 2002, pp. 65-76.

[13] *"The VINT ns-2 network simulator"* (2003) The VINT Project, UC Berkeley, LBL, USC/ISI, Xerox PARC, `http://www.isi.edu/nsnam/`, Jan. 2003.