2
Trabalhos Relacionados

Neste capítulo serão apresentados alguns simuladores de redes que estão relacionados com o tema deste trabalho.

Alguns dos simuladores de redes existentes foram estendidos para suportar ambientes de computação móvel, enquanto que outros foram desenvolvidos especificamente para esta finalidade. Alguns dos simuladores mais conhecidos para redes móveis ad hoc são o ns e o GloMoSim.

2.1.
ns

O ns (ns-2, 2002a, b) é um simulador flexível de protocolos de rede que suporta simulações de protocolos em larga escala, organizados em camadas. O ns também pode ser usado como uma emulador de rede, o qual é capaz de interagir com uma rede real. O ns inclui um gerador de cenários, o qual permite a geração automática de diferentes topologias de redes, padrões de tráfego e de falhas.

O ns é implementado num modelo de programação em dois níveis: o núcleo do simulador é implementado em C++ que interage com módulos programados em OTcl, uma extensão orientada a objeto do Tcl. Enquanto que o núcleo implementa os protocolos e as primitivas de simulação, o OTcl é usado para desenvolver os scripts de simulação descrevendo as funcionalidades dos elementos de rede simulados em uma linguagem de programação de alto nível. O OTcl também pode ser usado para implementar protocolos. Devido este modelo de programação, o ns é um simulador de protocolo de propósito geral, flexível, e extensível.

O ns, que atualmente está na versão 2 (ns-2), foi desenvolvido na Universidade da Califórnia em Berkeley como parte do projeto VINT, do qual ainda participam pesquisadores do LBL, USC/ISI e Xerox PARC. O grupo Monarch, da *Carnegie Mellon University* (CMU), foi quem estendeu o ns para redes móveis, suportando WLANs e MANETs.

Para suportar as redes móveis ad hoc o ns implementa quatro protocolos de roteamento: DSDV, DSR, AODV e TORA. Para a subcamada MAC existem duas opções: IEEE 802.11 ou TDMA. Enquanto que na camada de transporte estão definidos os protocolos TCP e UDP. É possível definir para um nó móvel o tipo de antena (p. ex., omnidirecional), e o esquema de energia, assim como o alcance do seu sinal de rádio, para delimitar os seus vizinhos. O endereçamento pode ser de três tipos: flat, hierárquico ou expandido. Os nós móveis são dispostos, através de coordenadas cartesianas, em um espaço tridimensional delimitado e gradeado, sendo que a resolução da grade pode ser ajustada. Para movimentar o nó móvel neste espaço é informado as coordenadas de destino e a velocidade com que o nó móvel deve se mover até este destino.

Foram feitas extensões ao modelo da CMU, para o uso do modelo sem fio em simulações com a rede fixa, esta característica foi chamada de *wired-cum-wireless*. O protocolo *Mobile IP* (implementado para nós da rede fixa) também foi integrado ao modelo sem fio, permitindo que o *Mobile IP* rodasse sobre os nós móveis.

2.2. GloMoSim

O GloMoSim (*Global Mobile Information Systems Simulation Library*) (Zeng et al, 1998; Glomosim, 2002) é um simulador de redes sem fio baseado em bibliotecas, desenvolvido no Laboratório de Computação Paralela da Universidade da Califórnia em Los Angeles (UCLA). O GloMoSim é composto por uma coleção de módulos de biblioteca implementados em PARSEC (*PARallel Simulation Environment for Complex systems*) (Bagrodia et al, 1998), uma linguagem baseada em C para descrição de simulações seqüenciais e paralelas.

O GloMoSim implementa um conjunto de protocolos de rede para comunicação sem fio, os quais estão organizados em uma arquitetura em camadas, mostrada na Tabela 1. Novos protocolos e módulos implementados em PARSEC podem ser facilmente adicionados à biblioteca do GloMoSim para compor diferentes simulações. O núcleo do PARSEC implementa uma máquina de simulação de alto desempenho, a qual permite o GloMoSim suportar a simulação de redes de computação móvel de larga escala, com total transparência de

simulação, tanto para o programador do protocolo quanto para o usuário do simulador. No modelo de programação do PARSEC, os protocolos são implementados em módulos monolíticos chamados *entities* (entidades), e permite somente a composição vertical de protocolos (protocolos em camadas). As primitivas do PARSEC são insuficientes para a definição de abstrações, tais como padrões de mobilidade, posto que estas devem ser programadas explicitamente pelo usuário do GloMoSim.

Tabela 1 – Camadas de Protocolos do GloMoSim

Camadas	Protocolos
Mobilidade	*Random waypoint, Random drunken, Trace based*
Propagação de Rádio	*Two ray and Free space*
Modelo de Rádio	*Noise Accumulating*
Modelos de Recepção de Pacote	Restringido pelo SNR, baseado em BER com modulação BPSK/QPSK
Enlace de Dados (MAC)	CSMA, IEEE 802.11 e MACA
Rede (Roteamento)	IP com AODV, *Bellman-Ford*, DSR, *Fisheye*, LAR *scheme* 1, ODMRP, WRP
Transporte	TCP e UDP
Aplicação	CBR, FTP, HTTP e Telnet

Atualmente o GloMoSim suporta protocolos apenas para redes sem fio. No futuro, irão adicionar funcionalidades para simular redes fixas, assim como redes híbridas (com e sem fio).

2.3. MobiCS

O MobiCS[3] (*Mobile Computing Simulator*) (Rocha & Endler, 2000; Rocha, 2001; MobiCS, 2002) é um simulador de protocolos distribuídos para computação móvel. É uma ferramenta única para prototipagem, teste, validação e avaliação de protocolos distribuídos, baseada em abstrações de programação de alto nível e na transparência da simulação.

[3] Pronuncia-se: mobix.

A versão atual do MobiCS contempla apenas as redes sem fio com infra-estrutura, e não suporta as redes móveis ad hoc. Para suporte as redes sem fio com infra-estrutura o MobiCS implementa abstrações como: host móvel (MH – *Mobile Host*), estação de suporte à mobilidade (MSS – *Mobile Support Station*), host fixo (FH – *Fixed Host*), célula, enlaces com fio e sem fio, componente *hand-off*[4], entre outras.

Uma das principais contribuições do MobiCS é implementar uma arquitetura de software para simuladores que suporta a troca do modo de simulação sem afetar o protocolo que está sendo prototipado, ou seja, o mesmo protocolo pode ser testado em diferentes modos de simulação sem a necessidade de alteração do seu código. O MobiCS implementa dois modos de simulação: determinístico e estocástico. No modo determinístico o usuário define um script de simulação descrevendo um cenário particular, que é executado passo a passo, permitindo o usuário observar o estado do protocolo ou do objeto de rede. Já no modo estocástico o usuário atribui padrões de comportamento probabilístico para os elementos simulados da rede. O modo determinístico é utilizado para teste e avaliação da correção do protocolo, já o modo estocástico é para testar o seu desempenho.

O MobiCS é um framework que foi desenvolvido em Java, e que oferece um ambiente extensível, flexível, com capacidade de criar abstrações que facilitem a criação de protótipos de protocolos distribuídos.

O usuário do MobiCS utiliza apenas uma linguagem de programação orientada a objetos (Java) tanto para desenvolver o protótipo do protocolo quanto para criar as simulações, o que facilita a portabilidade e extensão do simulador.

A maioria dos simuladores suporta alguma forma de definição de abstração e estruturação de protocolos, que geralmente estão estruturados em camadas. Já o MobiCS provê ao usuário um modelo de programação baseado em micro-protocolos (Bhatti et al, 1998), com o qual é possível descrever protocolos altamente modularizados e organizados em componentes funcionais, que

[4] Processo de transferência da responsabilidade sobre um MH entre dois MSSs, quando este migra de uma célula para outra. Este processo engloba todas as trocas de mensagens, alterações de estados de conexões e/ou elementos da rede envolvidos na migração do MH.

interagem através de eventos. O MobiCS induz a organização de um protocolo em três componentes funcionais:

- *Wired*: responsável pela troca de mensagens pela rede fixa.
- *Wireless*: responsável pela troca de mensagens pelos canais de comunicação sem fio.
- *Hand-off*: responsável pela comunicação entre os elementos de rede durante o processo de *hand-off*, e que pode incluir o envio de mensagens pela rede fixa e pelo canal sem fio.

Para criar uma simulação, o usuário basicamente implementa classes que definem os protocolos a serem simulados, o ambiente e o modelo de simulação desejado. Estas classes estendem as classes básicas do MobiCS, que implementam a funcionalidade das várias camadas de software do simulador. Do ponto de vista do usuário MobiCS, há dois pacotes Java principais que são utilizados para criar uma simulação. Com o pacote `mobics.ppi` o usuário programa protocolos que podem ser simulados, fazendo uso de dois sub-pacotes: `mobics.ppi.protocol`, que declara as classes necessárias à implementação do protocolo propriamente dito, e `mobics.ppi.message`, que é usado para a declaração de mensagens utilizadas pelo protocolo. Com o pacote `mobics.simulation`, o usuário implementa a simulação propriamente dita, em termos de número e tipos de elementos de rede envolvidos na simulação, topologia da rede, tipos de protocolos a serem simulados, modo e modelo de simulação.

A Figura 1 ilustra como o usuário implementa um protocolo e uma simulação a partir das classes básicas implementadas no MobiCS.

Atualmente o MobiCS é utilizado em disciplinas e em pesquisas no Departamento de Informática da PUC-Rio, no Instituto de Matemática e Estatística da USP, e no Departamento de Ciência da Computação da UFMG.

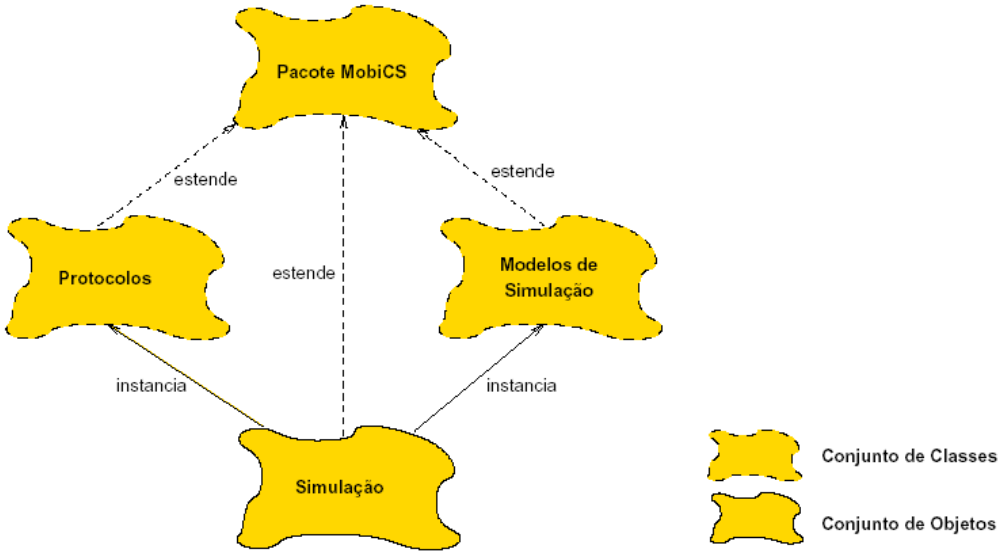

Figura 1 – Implementação de Protocolos e Simulações a partir do MobiCS

A idéia inicial desta dissertação era estender o framework MobiCS para simulação de redes móveis ad hoc. Após um estudo mais aprofundado do MobiCS, esta idéia foi abandonada e foi decidido desenvolver um novo framework especialmente para as redes móveis ad hoc. Este novo framework se chama MobiCS2, e será discutido a partir do próximo capítulo.

2.4. Outros Simuladores

O GloMoSim e o ns são os principais simuladores para MANETs, mas vale comentar o MaRS (*Maryland Routing Simulator*) (Alaettinoglu et al, 1994), um simulador de roteamento, no nível de pacotes, de eventos discretos. O MaRS é uma plataforma flexível desenvolvida especificamente para avaliação e comparação de algoritmos de roteamento para redes fixas. O MaRS foi incrementado para prover mobilidade nos nós (Das et al, 2000), apresentando algumas simplificações no modelo de simulação. Os nós podem se mover numa região retangular de acordo com um dado modelo de mobilidade. Cada nó tem um alcance de rádio fixo e tem um enlace para todos os outros nós do sistema. Se o outro nó não está dentro do alcance, o custo do enlace é infinito. De outra forma, o custo do enlace é modelado pela função *hop-normalized delay*, a mesma métrica (versão revisada) usada na ARPAnet. Não é modelada nenhuma contenção de múltiplo acesso ao meio físico, nem interferência. Essencialmente

cada enlace utiliza toda a largura de banda do canal enquanto transmite os pacotes. No modelo de simulação, um pacote pode ser transmitido via unicast ou broadcast, sendo que as transmissões broadcast são modeladas como uma seqüência de transmissões unicast para todos os nós vizinhos, porém o pacote é contabilizado somente uma vez nas estatísticas da simulação. O tempo de processamento do pacote de dados (*parsing* do cabeçalho, consulta à tabela de roteamento ou ao cache, e inserção do pacote na sua fila de saída) é fixo, já o tempo de processamento dos pacotes de roteamento depende do protocolo de roteamento sendo utilizado.

Outro simulador é o SensorSim (Park et al, 2000), que é um framework para simulação de redes de sensores. O SensorSim é um simulador de redes orientado a eventos, e possui as seguintes características: um modelo de energia nos nós sensores; simulação híbrida que permite a interação de nós reais com simulados; e novos protocolos de comunicação específicos para redes de sensores.

E por último, gostaríamos de mencionar o SSF (*Scalable Simulation Framework*) (SSF, 2002; Cowie, 1999), que é um framework de para simulação de redes fixas (não suporta redes móveis ad hoc). Este framework está disponível em duas versões: Java e C++. A sua arquitetura de protocolos é baseada no x-Kernel (Hutchinson & Peterson, 1991). Um dos seus principais objetivos é suportar simulações escaláveis e de alto desempenho.