6
Referências Bibliográficas

1- FRODIGH, M.; JOHANSSON, P.; LARSSON, P. **Wireless ad hoc networking—The art of networking without a network**. Ericsson Review. n°4, 2000.

2- BLAZEVIC, L. et al. **Self-Organization in Mobile Ad Hoc Networks: The Approach of Terminodes**, IEEE Communications Magazine, 2001.

3- **Projeto Sistema Brasileiro de Televisão Digital**. Disponível em: <http://tvd.ic.uff.br/detalhada.html> Acesso em: 21 de junho de 2007.

4- BLAZEVIC, L.; LE BOUDEC, J.; GIORDANO, S. **A Location-Based Routing Method for Mobile Ad Hoc Networks,** IEEE Transactions on Mobile Computing. v 3, n°. 4, 2004.

5- HUBAUX, P.; GROSS, T.; LE BOUDEC, J.-Y.; AND VETTERLI, M. **Toward Self-Organized Mobile Ad Hoc Networks: The Terminodes Project**, IEEE Communications. Magazine, 2001.

6- BLAZEVIC, L.; GIORDANO, S.; LE BOUDEC, J.-Y. **Self Organized Terminode Routing**, Cluster Computing Journal, 2002.

7- REMONDO, D. **Tutorial on wireless ad hoc networks**. In: International Working Conference in Performance Modelling and Evaluation of Heterogeneous Networks (HET-NET), 2004.

8- **Mobile Ad-hoc Networks (manet)**. Disponível em: <http://www.ietf.org/html.charters/manet-charter.html> Acesso em: 21 de julho de 2007.

9- DOYLE, J. **Dynamic Routing Protocols**, Cisco Press, 2001. Disponível em: <http://www.ciscopress.com/articles/article.asp?p=24090> Acesso em: 21 de julho de 2007.

10- JOHNSON, D. et al. **The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4**, http://www.ietf.org/rfc/rfc4728.txt, 2007.

11- HUBAUX, J.P. et al. **Towards mobile ad-hoc WANs: Terminodes**. In: Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2000.

12- BASAGN, S. et al. **A Distance Routing Effect Algorithm for Mobility (DREAM)**, Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom) '98, 1998, pp. 76–84.

13- LI, J. et al. **A Scalable Location Service for Geographic Ad Hoc Routing**, Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), 2000, pp. 120–130.

14- STOJMENO, I.. **Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks**, Technical Report TR-99–10, Computer Science, SITE, University of Ottawa, Canada, 1999.

15- GIORDANO, S.; HAMDI, M. **Mobility Management: The Virtual Home Region**, Technical Report SSC/1999/037, EPFL-ICA, 1999.

16- CAPKUN, S.; HAMDI, M.; HUBAUX, J.-P. **GPS-free positioning in mobile ad-hoc network**, IEEE Hawaii International Conference on System Sciences (HICSS-34), 2001.

17- BUTTYAN L.; HUBAUX J.-P. **Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks**. Technical Report DSC/2001/001, Swiss Federal Institute of Technology - Lausanne, 2001.

18- KARP, B.; KUNG, H.T. **GPSR: Greedy Perimeter Stateless Routing for Wireless Networks**, Proceedings ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00), 2000. P. 243-254.

19- Macedo, D.F. et al. **Um Protocolo de Roteamento para Redes Ad Hoc com QoS Baseado no Controle da Potência de Transmissão**, Article published in 24th Simpósio Brasileiro de Redes de Computadores (SBRC), 2006. P. 605-620.

20- WANG, X.F.; CHEN, G. **Complex networks: Small-world, scale-free, and beyond**, IEEE Circuits and Systems Magazine, 2003.

21- **GloMoSim**. Disponível em: <http://pcl.cs.ucla.edu/projects/glomosim/> Acesso em: 21 de julho de 2007.

22- **Parsec**. Disponível em: <http://pcl.cs.ucla.edu/projects/parsec/> Acesso em: 21 de julho de 2007.

23- C. E. Perkins; E. M. Royer. **Ad-Hoc On-Demand Distance Vector Routing**, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999, P. 90–100.

24- Y.-B. Ko; N. H. Vaidya. **Location-Aided Routing (LAR) in Mobile Ad Hoc Networks**, Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking(MobiCom), 1998. P. 66–75.

25- Nuevo J. **A Comprehensible GloMoSim Tutorial**, 2004. Disponível em: <www.icis.ntu.edu.sg/wagio/campus/research/glomosim/glomoman.pdf/> Acesso em: 21 de julho de 2007.

Apêndice A

GloMoSim

O GloMoSim é um ambiente de simulação escalável para redes sem fio. O GloMoSim utiliza o Parsec para fornecer sua capacidade de simulação discreta de eventos paralelos. Ele tem a capacidade de simular redes com até mil nós, que podem ser conectados por links heterogêneos. Sua capacidade inclui o suporte para redes ad hoc sem fio e os protocolos da Internet, e modelos de mobilidade dos nós.

Uma vantagem desse simulador é sua construção em camadas similares as utilizadas no modelo OSI. Para a troca de mensagens entre camadas são utilizadas APIs padronizadas. Isto permite uma rápida integração de modelos desenvolvidos em diferentes camadas. Os protocolos suportados atualmente são mostrados na tabela seguinte.

Tabela A.1 – camadas do simulador Glomosim

Camadas	Modelos
Mobilidade	Random waypoint, Random drunken, Trace based.
Rádio Propagação	Free space, Two-Ray
Modelos de recepção de pacotes	Limitada por SNR, baseado em BER com modulação BPSK/QPSK.
Enlance (MAC)	CSMA, MACA, TSMA, 802.11
Rede (roteamento)	IP com os protocolos de roteamento, AODV, Bellman-Ford, DSR, Fisheye, LAR scheme 1, ODMRP, WRP
Transporte	TCP, UDP
Aplicação	CBR, FTP, HTTP and Telnet

Essa característica do GloMoSim permite adicionar os modelos desenvolvidos para os novos testes no protocolo, fazendo alterações pontuais em algumas camadas do simulador. Os protocolos podem ser escritos basicamente em linguagem C com algumas funções Parsec para controlar o tempo de eventos [21].

Além desses protocolos, na versão do simulador utilizada foi acrescentado o protocolo de roteamento terminodes, utilizando o nome GEODSR. E o modelo de mobilidade *Restricted random waypoint*. Mas esses protocolos ainda não estão na versão oficial do simulador, mas podem ser encontrados na Internet em: http://ica1www.epfl.ch/TNRouting.

A.1
Obtendo e Instalando o GloMoSim

O GloMoSim pode ser obtido gratuitamente na Internet no site: http://pcl.cs.ucla.edu/projects/glomosim. Será obtido um arquivo que deve ser descompactado. Dois diretórios subdiretórios podem ser encontrados: *glomosim* e *parsec*.

O diretório *parsec* contém as bibliotecas pré-compiladas para alguns sistemas operacionais, entre eles o Windows e o Linux Red Hat. A variável de ambiente "PCC_DIRECTORY" deve apontar para subdiretório contendo as bibliotecas do sistema operacional correspondente.

O diretório *glomosim* corresponde ao simulador e contém os seguintes subdiretórios:

- /application – contém os códigos para a camada de aplicação.
- /bin – contém o executável e arquivos de entrada e saída.
- /doc – documentação do simulador.
- /include – contém os arquivos comuns de cabeçalhos (*.h)
- /java_gui – contém a ferramenta de visualização.
- /mac – contém os códigos para a camada mac.

- /main - contém as funções principais do simulador.
- /network – contém os códigos para a rede.
- /radio – contém os códigos para a camada de rádio.
- /scenarios – contém alguns exemplos de cenários de simulação.
- /tcplib – bibliotecas para o protocolo TCP.
- /transport – contém os códigos para a camada de transporte.

A.2
Executando uma simulação.

Para executar o GloMoSim e realizar uma simulação, deve-se ir para subdiretório /bin do simulador, e executar o seguinte comando: ./glomosim *config.in.*

O arquivo *config.in* contém a configuração para os parâmetros de simulação (o nome do arquivo pode ser outro, embora este seja o padrão). Ao final da simulação será gerado o arquivo *glomo.stat* contendo todas as estatísticas geradas.

A.3
O arquivo de configuração do GloMoSim.

A seguir será apresentada uma tabela contendo os principais parâmetros que devem ser configurados.

Tabela A.2 – parâmetros iniciais do arquivo de configuração

Parâmetro	Descrição
SIMULATION-TIME	Tempo máximo da simulação
SEED	Semente para o gerador de números aleatórios.
TERRAIN-DIMENSIONS	Área de simulação
NUMBER-OF-NODES	Número de nós sendo simulado
NODE-PLACEMENT	Estratégia de distribuição inicial sobre a área de simulação.
MOBILITY	Seleciona o modelo de mobilidade utilizado

Tabela A.3 – parâmetros de configuração do modelo de propagação e de rádio

Parâmetro	Descrição
PROPAGATION-LIMIT	Sinais abaixo desse valor (em dBm) não são entregues.
PROPAGATION-PATHLOSS	Modelo de perda de propagação.
NOISE-FIGURE	Figura de ruído.
TEMPERATURE	Temperatura ambiente.
RADIO-TYPE	Modelo para transmitir e receber pacotes.
RADIO-FREQUENCY	Freqüência em Hz
RADIO-BANDWIDTH	Banda passante em bits/s
RADIO-RX-TYPE	Modelo de recepção de pacotes.
RADIO-TX-POWER	Potência de transmissão em dBm.
RADIO-ANTENNA-GAIN	Ganho da antena do rádio.
RADIO-RX-SENSITIVITY	Sensibilidade do rádio (em dBm)
RADIO-RX-THRESHOLD	Potência mínima para receber um pacote (em dBm)

Tabela A.4 – configuração dos protocolos utilizados em cada camada

Parâmetro	Descrição
MAC-PROTOCOL	Protocolo da camada MAC
NETWORK-PROTOCOL	Protocolo da camada de rede, somente o protocolo IP está disponível.
ROUTING-PROTOCOL	Protocolo de roteamento
APP-CONFIG-FILE	Indica o arquivo que contém a configuração das aplicações.

Os parâmetros defininem quais camadas ou protocolos devem ter suas estatíscas gravadas no arquivo *glomo.stat*: APPLICATION-STATISTICS, TCP-STATISTICS, UDP-STATISTICS, ROUTING-STATISTICS, NETWORK-LAYER-STATISTICS, MAC-LAYER-STATISTICS, RADIO-LAYER-STATISTICS , CHANNEL-LAYER-STATISTICS, MOBILITY-STATISTICS.

Um arquivo texto será usado para configurar as aplicações, e terá formato para os diferentes tipos de aplicação. Cada linha do arquivo cria uma aplicação para ser simulada. Abaixo é mostrado como configurar um gerador de dados constante CBR, que foi a aplicação usada nesse trabalho.

Linha do arquivo de configuração:

CBR <src> <dest> <items to send> <item size> <interval> <start time> <end time>

Parâmetros:

- <src> - nó cliente CBR, gerador dos dados.
- <dest> - servidor CBR.
- <items to send> - itens que devem ser enviados. Se for igual a 0, envia os dados até a aplicação ser encerrada.
- <item size> - tamanho do pacote enviado.
- <interval> - intervalo de transmissão dos dados.
- <start time> - tempo que a aplicação inicia.
- <end time> - tempo que a aplicação termino. Se for igual a 0, envia a quantidade de dados de <items to send>, ou até final da simulação.

A.4
Estrutura básica dos códigos do GloMoSim

Para poder implementar novos protocolos e realizar alguma mudança, é necessário entender a estrutura básica da codificação do GloMoSim. Como mostrado anteriormente, os códigos são relativos a cada camada estão localizados em diretórios diferentes. Isto facilita o entendimento e organiza o código.

O Glomosim não cria uma entidade para cada nó que fará parte da simulação, para diminuir o custo da simulação, por exemplo, memória. No Glomosim cada entidade representa uma área geográfica da simulação (uma partição), que representará todos os nós contidos dentro dessa área. Uma estrutura de dados separados representa o estado completo de cada nó mantido dentro da entidade. Atualmente, o Glomosim só suporta uma partição.

O diretório *main* contém os arquivos com as funções principais do simulador. O arquivo *driver.pc* Contém a função principal do programa, *main( )* , que faz a leitura do arquivo de configuração e escreve as estatísticas finais no arquivo específico. Neste arquivo é definido o *entity driver*, que é obrigatório em todo programa Parsec. A função *main* chama a função *parsec_main()*, que inicia a *engine* Parsec e cria a *driver entity.*

O *entity driver* tem função semelhante ao *main* no C, ele vai ler alguns dos parâmetros do arquivo de configuração como: área da simulação, tempo de simulação, número de nós, faz a distribuição inicial dos nós sobre a área de simulação, e inicia a simulação. O *entity driver* inicia a simulação enviando a mensagem *StartSim* para a *partitionEntityName*, que é uma instancia do tipo *GLOMOPartitio*. O *partitionEntityName* irá controlar a simulação.

O tipo *GLOMOPartition* é definido no arquivo *glomo.pc* também no diretório *main*. Este é o tipo usado para definir uma partição no Glomosim, que no caso atual, contém todos os nós presentes na simulação. As seguintes funções serão executadas:

- Recebe as informações de *driver entity.*
- Descobre os nós que pertencem a partição, e para cada nó inicializa as camadas do simulador chamando a função específica de cada camada.
- Fica em loop infinito esperando receber mensagens, quando uma é recebida seleciona a camada e o nó específico.
- Chama as funções para finalizar a execução de todas as camadas e todos os nós, permitindo coletar as estatísticas.

Os dados de uma partição são guardados em um *struct* definido em */main/glomo.h*, entres os principais campos estão: os dados recebidos por *driver entity*, ponteiros para acessar os dados específicos de cada nó, e ponteiros para acessar a lista de eventos agendados. Os dados específicos de cada nó são guardados em um *struct* definido em */include/api.h*, com os principais campos: os

parâmetros especificados para cada camada, o endereço e localização geográfica do nó, ponteiro que permite acessar os dados da partição e ponteiro para acessar dados de outros nós. A figura abaixo mostra de forma básica, como os dados são organizados no Glomosim. Uma estrutura de dados é definida para a partição, uma para os nós e uma para cada camada, formando uma hierarquia. As setas na figura representam ponteiros, que é a forma que as estruturas são acessadas.

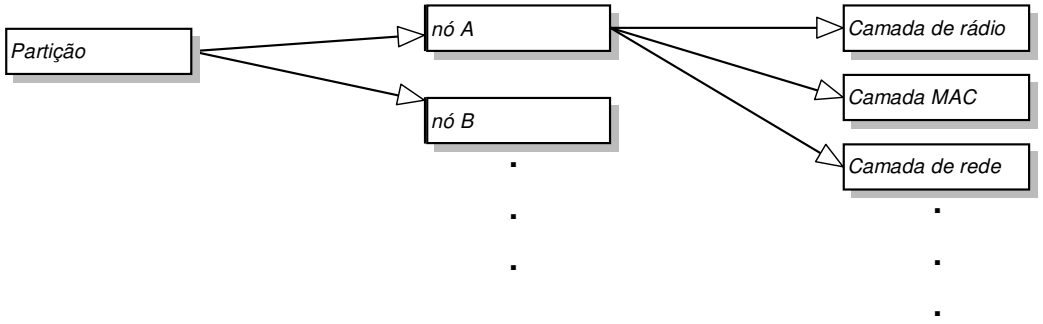

Figura A.1 – organização básica dos dados no GloMoSim

As funções de inicialização são: GLOMO_PropInit(), GLOMO_RadioInit(), GLOMO_MacInit(), GLOMO_NetworkInit(), GLOMO_TransportInit(), GLOMO_AppInit() e GLOMO_MobilityInit(). Ou seja, existe uma função específica para cada camada. Em geral, essas funções inicializam as variáveis comuns de cada e chamam uma função específica (definida dentro do diretório da camada correspondente) de incialização para o modelo escolhido para cada camada. Na figura a seguir é um exemplo de inicialização da camada de aplicação. O exemplo mostra as funções chamadas para inicializar um gerador de dados CBR definido para um nó.

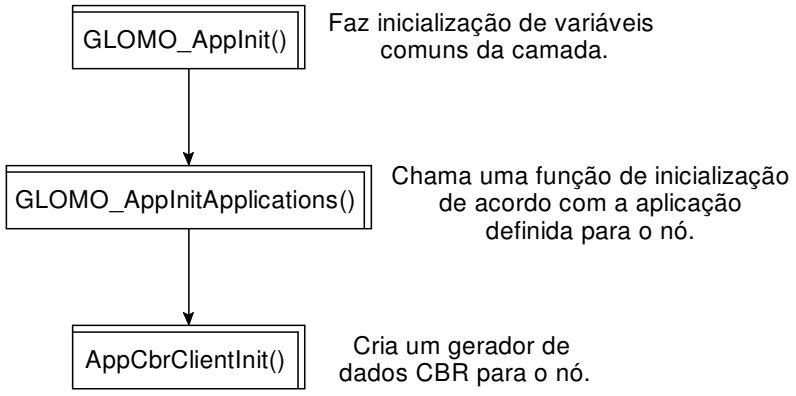

Figura A.2 – funções chamadas para criar um gerador de dados CBR

As mensagens para cada camada são geradas por eventos agendados por alguma função dentro do programa. As funções de inicialização agendam algumas mensagens iniciais para cada camada. E as aplicações definidas para geram as mensagens de dados na simulação.

Após executar as funções de inicialização para todas as camadas, todos os nós da partição estarão configurados. O programa entra em *loop* infinito verificando a ocorrência de um evento de simulação. Um evento é formado por uma mensagem para um nó e uma camada específica. Para cada avanço do relógio, sempre será verificada a ocorrência de eventos de mobilidades antes de verificar se uma mensagem de dados foi agendada para o mesmo instante. Isso evita que uma mensagem seja enviada para um nó que poderia estar inalcançável.

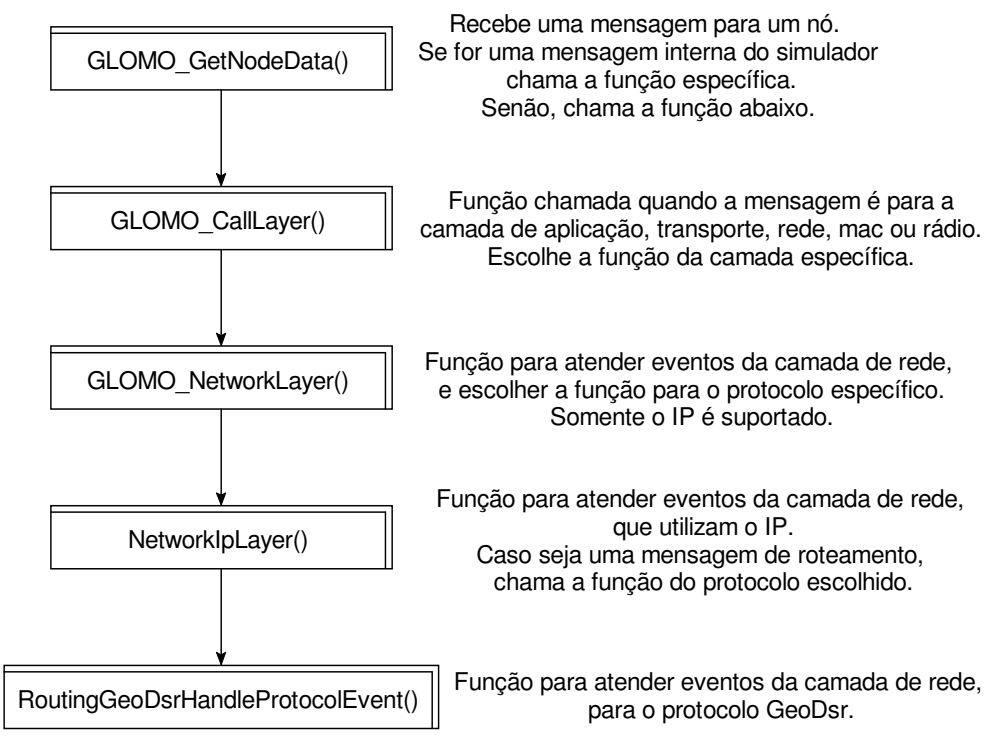

Figura A.3 – atendendo um evento para o protocolo GeoDsr.

Quando um evento acontece o Glomosim executará algumas funções para manipular a mensagem recebida. A figura acima mostra um exemplo em que uma mensagem é recebida pelo protocolo de roteamento GeoDsr. Todas as camadas seguem uma estrutura semelhante a essa mostrada. Será chamada uma função de

uso geral para cada camada, e as funções seguintes serão chamadas de acordo com o protocolo utilizado. E a função específica do protocolo pode chamar diferentes funções para tratar a mensagem recebida.

As funções de finalização de cada camadas, para cada nó, serve basicamente para colocar as estatísticas de interesse que serão gravadas no arquivo *glomo.stat*.

Todas mensagens que podem ser agendadas no simulador são definidas em */include/structmsg.h*. Então, qualquer nova mensagem deve ser definida nesse arquivo. Para agendar um evento no Glomosim, deve ser a escolhida a camada, o protocolo e um dos tipos de mensagens desse arquivo que será enviada.

Para adicionar um protocolo, o mínimo que deve ser definido é uma função e inicialização e outra finalização. Além da função para tratar as mensagens específicas do protocolo. E como mostrado, a opção de escolher as funções desse novo protocolo devem ser inserida na função específica de cada camada. As novas mensagens desse protocolo devem ser definidas em */include/structmsg.h*. A constante que representa o protocolo deve ser declarada no tipo que define os protocolos de cada camada. Isso é feito no arquivo *.h* da camada no diretório */include*.

Outras informações sobre o Glomosim podem ser encontradas no manual do usuário e em [25].