Modelos de mobilidade

EP2 – Entrega: 23/11/2008

Computação Móvel – MAC5743 / MAC0463

Segundo semestre de 2008

1. Introdução

Um modelo de mobilidade é uma forma de gerar informações de movimento para nós em uma rede. Um modelo é capaz de gerar diversas movimentações, ou seja, ele faz uso de dados aleatórios para não gerar sempre a mesma coisa. Por exemplo, no caso de um modelo que gere movimentos circulares, para cada nó deve ser gerado o raio e a velocidade angular. Logo, apesar de um modelo descrever de forma clara o comportamento, diferentes execuções apresentam resultados distintos.

As informações de movimento para nós em uma rede é usada para simular redes *ad-hoc* com nós móveis e assim testar diferentes protocolos para tais redes. Os resultados obtidos com essas simulações dependem fortemente do modelo de mobilidade utilizado, portanto este deve ser escolhido com cuidado.

Entrega

Recomendo **fortemente** que este trabalho seja feito **em dupla**!! Quem quiser sofrer pode fazer sozinho, mas nunca em trio!

A entrega deve ser feita pelo Paca, através de um arquivo **ZIP** ou **TAR.GZ** com o nome da dupla. Exemplos:

- **`MarianaAlfredo.zip`**
- **`MarianaBravoAlfredoGoldman.tar.gz`**

Basta que um integrante da dupla entregue o arquivo, e nada de iniciais por favor!

Esse arquivo deve conter o programa desenvolvido e mais um relatório (veja na seção 5 o que esse relatório deve conter).

2. O Exercício

O trabalho de vocês neste exercício será implementar três modelos de mobilidade para uma ferramenta livre chamada BonnMotion[1]. Essa ferramenta já possui alguns modelos, além de facilidades para salvar as informações de movimento geradas em diferentes formatos. Vocês mesmos devem escolher quais modelos implementar, sendo que podem procurar definições na Internet ou inventar seu próprio modelo. A única restrição é não implementar algo que o BonnMotion já tenha.

Os modelos que já estão presentes no BonnMotion são os seguintes:

- GaussMarkov – modelo de Gauss-Markov
- OriginalGaussMarkov – modelo de Gauss-Markov original

1 http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/

- ManhattanGrid – modelo Manhattan Grid
- RandomWaypoint – modelo Random Waypoint
- RPGM – modelo Reference Point Group Mobility
- Static – modelo estático (nenhum movimento)
- ChainScenario – cenário para ligar diferentes modelos

Recomendo a leitura do artigo "*A Survey of Mobility Models for Ad Hoc Network Research*" disponível no Paca para explicação de alguns do modelos acima e principalmente para entender melhor o que são os modelos de mobilidade.

Para os que quiserem inventar seu próprio modelo, usem a criatividade à vontade e não se preocupem se o modelo é realístico ou não. Entretanto é necessário que o modelo seja claro.

3. O BonnMotion

O BonnMotion é um projeto de pesquisa disponível como software livre, e talvez por conseqüência disso o código e a documentação *não são lá as mil maravilhas*. Não esperem ter um EP fácil, mas procurem postar perguntas e respostas no fórum pois isso deve amenizar a situação.

Aliás, o código é em Java. Eu separei o software em dois projetos (são para o Eclipse, mas podem usar fora também, eu só não vou explicar como). Eles estão disponíveis no Paca. A seguir explico o que é cada projeto e como montar o seu ambiente de programação, segundo minha sugestão.

Projeto BonnMotion

Esse projeto contém todas as classes base para o BonnMotion funcionar. Algumas classes são de interesse especial pois será preciso conhecê-las para implementar novos modelos. Elas são explicadas a seguir:

- **`MobileNode`**
  Representa um nó móvel em qualquer cenário de simulação. Usa-se o método **`add(double, Position)`** para acrescentar uma posição em um determinado tempo para esse nó. Alguns outros métodos também podem ser interessantes, como **`add(MobileNode)`**, **`changeTimes()`**, **`positionAt(double)`** e **`shiftPos(double, double)`**.
- **`Scenario`**
  É essa classe (ou alguma de suas subclasses) que os modelos devem estender para realizar as simulações. Ela guarda um **`MobileNode[]`** com os nós móveis da simulação, os valores **`x`** e **`y`** das dimensões do cenário, a duração do mesmo, entre outros. Para o programa funcionar, todos os modelos devem ter um construtor que recebe um **`String[]`** com os parâmetros de simulação e, no próprio construtor, devem preencher o **`MobileNode[]`** com a simulação. Existem métodos **`parseArg(char, String)`** e **`parseArg(String, String)`** para ajudar na leitura desses parâmetros. Recomenda-se também chamar os métodos **`preGeneration()`** e **`postGeneration()`** para cuidar automaticamente do corte do início da simulação[2].

Sugiro que vocês explorem um pouco o código de **`Scenario`** e de algumas das

2 Isso é explicado com mais detalhe no artigo, mas trata-se basicamente de eliminar os primeiros segundos de simulação para evitar quadros instáveis da mesma.

subclasses para entender melhor como a simulação funciona. Como eu já disse, o código não é lindo, mas dá para acompanhar.

Projeto BMRunner

Esse projeto contém apenas uma classe, **BM**, que é responsável por fazer o programa rodar. É um pouco modificada da classe original, pois adicionei a possibilidade de executar um modelo que não está no BonnMotion. Notem que este projeto depende do projeto anterior em seu *classpath* para funcionar.

Vamos ver a seguir os principais argumentos a serem usados com a classe **BM** para rodar o programa.

- Para um pouco de ajuda:

  -h ajuda geral

  -hm lista todos os modelos

  -hm <modelo> ajuda específica de um modelo

- Para rodar um modelo do BonnMotion:

  -f <nome> <modelo> <argumentos>, exemplos:

  -f simulacaoRW RandomWaypoint -n 10 -x 400 -y 400
  roda uma simulação do RandomWaypoint com 10 nós em uma área de 400 por 400, salvando os parâmetros e os movimentos em arquivos com nome "simulacaoRW".

  -f simulacaoRW2 -I simulacaoRW.params RandomWaypoint -n 30
  roda uma simulação usando os mesmos parâmetros da anterior, mudando apenas o número de nós para 30.

- Para rodar um modelo e salvar os arquivos em um formato diferente, seguir detalhes da ajuda da aplicação. Por exemplo para salvar no formato do NS2, os parâmetros são:

  NSFile -f simulacaoNS RandomWaypoint -n 10 -x 400 -y 400
  roda a mesma simulação do primeiro exemplo do item anterior, mas salvando o arquivo no formato do NS2.

Seu projeto

O seu projeto precisa depender do BonnMotion para poder estender as classes apropriadas. Além disso, para rodar os seus modelos o BMRunner precisará ter o seu projeto no *classpath*. Para rodar o **BM** com o seu modelo, a única diferença nos parâmetros é que ao invés de passar o nome da classe apenas, vocês precisam passar o nome qualificado dela. Por exemplo, ao invés de:

-f simulacaoRW RandomWaypoint -n 10 -x 400 -y 400

Vocês chamariam:

-f simulacaoRW meu.pacote.MeuModelo -n 10 -x 400 -y 400

Sugiro que vocês implementem seus modelos como subclasses ou de **Scenario**, ou de **RandomSpeedBase** quando for o caso, lembrando de:

- Prover um construtor que recebe **String[]** e preenche o **MobileNode[]** corretamente;
- Implementar um método estático em suas classes chamado **printHelp()**,

que recomenda-se que chame o **`printHelp()`** da superclasse, escrevendo ajuda específica para o seu modelo.

4. Bônus

Os bônus são opcionais e não podem diminuir sua nota, mas podem aumentá-la. Darei bônus por três motivos:

1. Testes automáticos para seus modelos, ou para classes que vocês desenvolveram;
2. Testes automáticos para classes do BonnMotion (vocês podem fazer estes testes para entender melhor como alguma coisa funciona, por exemplo);
3. Refatorações e melhorias no código do BonnMotion (lembrem de apontar isso no relatório).

5. Relatório

Segue abaixo uma "*checklist*" do que eu espero ver no relatório de vocês:

- Lista dos modelos implementados;
- Para cada modelo, citar a fonte onde conseguiram ou se vocês que inventaram;
- Para cada modelo, explicar brevemente e com suas próprias palavras como ele funciona (explicar um pouco melhor se foram vocês que inventaram o modelo);
- Indicar como os modelos foram implementados e eventuais dificuldades enfrentadas com o BonnMotion e na implementação;
- Se fizeram algum bônus, informar isso no relatório, junto com uma lista do que foi feito com relação aos bônus.

Boa diversão!