

Departamento de Engenharia Informática

Licenciatura em Engenharia Informática

Mobilidade em Redes de Comunicação 2011/2012

**Mobilidade IP**



Gonçalo Silva Pereira 2009111643

Igor Nelson Garrido da Cruz 2009111924

**Índice**

1. Introdução…………………………..………………………………………………3
2. Descrição geral do emulador……..………………………………………………4
3. Descrição das funcionalidades base e das funcionalidades opcionais……….5
4. Uma pequena explicação da GUI………………………………………………...7
   1. *Mobility Node* GUI………………………………………………………….7
   2. *Correspondent Node* GUI…………………………………………………8
   3. *Home Agent* GUI…………………………………………………………...9
   4. *Foreign Agent* GUI………………………………………………………..10
5. Descrição do código……………………………………………………………...11
6. Descrição dos principais problemas e de como os mesmos foram resolvidos……………………………………………………………………….…13
7. Conclusão…………………………………………………………………………14
8. **Introdução**

Este projecto tem por base a conceção de um emulador para fazer a simulação de mobilidade IP, aplicando e desenvolvendo os conhecimentos adquiridos nas aulas de Mobilidade em Redes de Comunicação e compreendendo os conceitos relacionados com as principais funções e limitações relativamente ao conceito de mobilidade compreendido pela IETF.

A arquitetura da aplicação baseia-se essencialmente em agentes/nós que são simulados através de programas independentes que comunicam entre si com *sockets*, para além disso têm também *Threads* responsáveis pelas diversas tarefas, como por exemplo com a GUI e a receção de pacotes.

Para implementar as diversas funcionalidades propostas, tivemos que recorrer aos conhecimentos adquiridos nas aulas desta cadeira e de outras cadeiras de Licenciatura, como por exemplo Introdução as Redes de Comunicação, Protocolos de Comunicação e Sistemas Distribuídos, tais como a utilização de *sockets* TCP/IP, *Threads*.

1. **Descrição geral do emulador**

Este emulador baseia-se em nós/Agentes, um cenário emulado é um sistema com dois routers, a simular o *Foreign* *Agent* (FA) e o *Home* *Agent* (HA) e duas pessoas, uma a simular o *Mobility* Node (MN) e outra a simular o *Correspondent* Node (CN), como se pode observar na figura seguinte:

De forma a guardar as informações relativamente a cada um dos nós moveis existentes no sistema são utilizas duas tabelas, uma no *Home* *Agent*, *Mobility* *Binding* *Table*, e outra no Foreign *Agent*, *Visitor* *Table*. Assim que um nó móvel muda de rede para a *Foreign* *Agent* é criada uma entrada na *Mobility* *Binding* *Table* (MBT) com o nome do nó que se moveu, com o *home address* e o *care of address* do mesmo, assim como o *time-to-live*.

Quando o *Corresponding Node* quer enviar um pacote para o *Mobility* Node, o *Home Agent* vai consultar a tabela MBT e verificar se já existe alguma entrada lá, se existir manda o pacote para o FA respetivo, caso não exista, envia-o logo para o Nó, uma vez que o mesmo se encontra dentro da sua rede.

Após a mudança de rede do *Mobility* node, a comunicação é feita através de um túnel entre o *Home* *Agent* e o *Foreign* *Agent*, uma vez que o *Home* *Agent* consulta a tabela e envia o pacote para o *Foreign* *Agent* da *Foreign* *Network* onde o *Mobility Node* está ligado.

1. **Descrição das funcionalidades base e das funcionalidades opcionais**

As principais funcionalidades deste emulador são:

Descoberta de agentes:

* Os agentes móveis conhecem a rede à qual estão ligados.
* Os agentes móveis podem receber um *broadcast* de outro agente, ficando assim a saber que o agente ainda está “vivo” dentro da rede, ficando também a saber se o mesmo veio de um *Home* *Agent* ou de outro agente qualquer.

Registo de utilizadores:

* Quando um nó móvel determina que chegou a outra rede, ele se registra no FA, enviando-lhe um pedido de registo com o endereço do seu *Home* *Agent*.
* Por sua vez o *Foreign* *Agent* envia um pedido para o *Home* *Agent*, com o *Home-of-Address* e o *Care-of-Address*.
* Quando o *Home* *Agent* recebe a mensagem de pedido de registo verifica as credenciais do *Mobility* Node, atualiza a MBT e envia uma mensagem para o FA, a dizer que aceita ou rejeita o pedido.
* Depois de receber o resultado do HA, o FA atualiza a *Visitor* *List* e envia o resultado para o MN que efetuou o pedido.
* O registo no FA tem um limite temporal, e se ele expirar, o FA remove automaticamente a entrada correspondente ao nó da *Visitor* *List*. Para evitar isto o nó móvel deve renovar o registo antes do mesmo expirar.

Serviço comum:

* Quando um *Correspondent* *Node* quer enviar um pacote para um nó móvel, envia um pacote para o *Home* *Address* desse nó móvel, depois o HA consulta a MBT para determinar a localização do nó móvel.
* Se o nó móvel está na sua rede domestica, o HA envia o pacote utilizando a rota padrão.
* Se o nó móvel está numa rede estrangeira, o HA determina o *Care-Of-Address* através da MBT, então o pacote recebido é encapsulado noutro pacote e o endereço de destino passa a ser o *Care-Of-Address* e o de origem o do HÁ.
* Quando o pacote chega ao FA, ele acede ao pacote interno e determina o endereço de destino original e consulta-o na *Visitor* *List*, para determinar o MAC *Address* para o qual o pacote deve ser enviado.
* O nó móvel também pode enviar pacotes para o CN, seguindo o pacote a rota contrária à descrita acima.

Otimização de rotas:

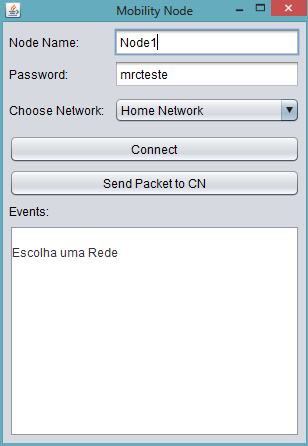
* A otimização da rota baseia-se essencialmente por eliminar a passagem do pacote pelo HÁ, uma vez que o CN faz o seu trabalho e envia o pacote já encapsulado para o FA correspondente.
* Esta otimização de acontece quando o MN envia um *Binding* *Update* com o seu *Care-Of-Address* para o CN, que irá responder com um *Binding* *Acknowledgement*, assim a comunicação entre o CN e o MN pode ser feito diretamente sem a intervenção do HÁ.

Cancelamento de registo de utilizadores

* Acontece quando um nó móvel volta à sua HN, fazendo com que o HA remova a entrado do nó móvel da MBT, e a partir daí o encaminhamento é feito normalmente.
* Normalmente não é necessário o cancelamento de registo no FA, uma vez que o mesmo irá expirar.

1. **Uma pequena explicação da GUI**
   1. ***Mobility Node GUI***

Na figura seguinte podemos visualizar a interface gráfica do *Mobility Node*.

Nesta interface temos os campos para inserir as credenciais do nó, assim como o nome e a palavra-chave.

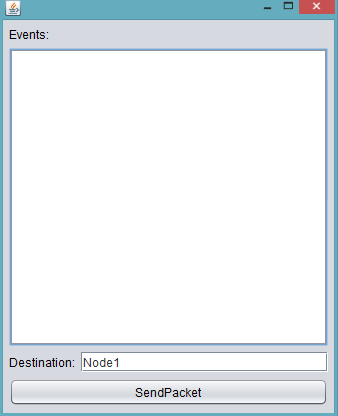
Podemos de seguida escolher a rede à qual queremos conectar, e carregar no botão *Connect* para estabelecer a comunicação.

Temos também a possibilidade de enviar um pacote para o “*Correspondent Node*”.

Temos também uma caixa de texto onde vão ser mostrados os eventos na rede relacionados com o nó em questão.

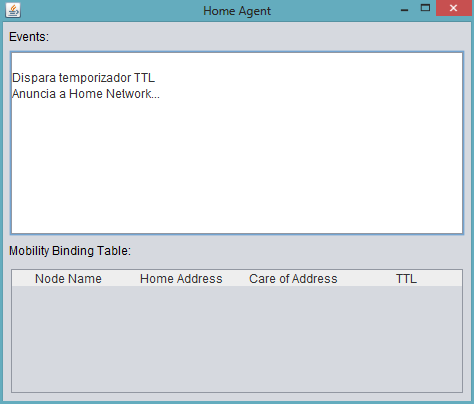
Os eventos que são mostrados são: a conexão a uma rede (*Home Network* ou *Foreign Network*), os *Broadcasts* recebidos/enviados, o envio ou a receção de um pacote, assim como outras mensagens de falhas de comunicação.

* 1. ***Correspondent Node GUI***

Na figura seguinte podemos ver a interface gráfica do correspondente node, onde podemos visualizar uma caixa com os eventos que vão ocorrendo dentro do sistema.

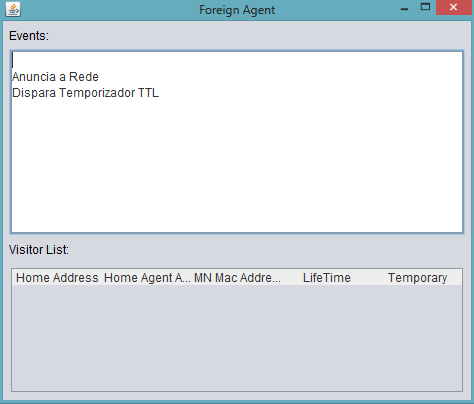
Nesta interface gráfica podemos escolher o nó de destino a enviar um pacote e envia-lo.

* 1. ***Home Agent GUI***

****Na figura seguinte podemos ver a interface do *Home* *Agent*. Nesta interface está presente uma caixa de eventos, onde há media que vão acontecendo os eventos, os mesmos vão sendo adicionados, para facilitar a compreensão do simulador ao utilizador. Nesta interface é também possível ver as entradas na *Mobility* *Binding* *Table* (MBT), quando um nó móvel se muda para uma *Foreign* *Network* é adicionada uma entrada na tabela para o *Home* *Agent* saber para onde tem de reencaminhar o pacote para o mesmo ser recebido pelo nó móvel.

* 1. ***Foreign Agent GUI***

Na figura seguinte podemos ver a interface do *Foreign Agent*. Nesta interface está presente uma caixa de eventos, onde há media que vão acontecendo os eventos, os mesmos vão sendo adicionados, para facilitar a compreensão do simulador ao utilizador. Nesta interface é também possível ver as entradas na *Visitor* *Table*, quando um nó móvel se muda para uma *Foreign* *Network* é adicionada uma entrada na tabela para o *Foreign* *Agent* saber para onde tem de reencaminhar o pacote para o mesmo ser recebido pelo *Correspondent* *Node*.

****

1. **Descrição do código**

Foi necessário dividir os Agentes em vários projetos e para além de isso definir uma ordem pela qual os mesmos devem ser executados, sendo que o dígito inicial define qual o lugar que o mesmo deve ter aquando a execução do programa geral. Já as letras finais definem os Agentes (MN: *Mobility Node*, HA: *Home Agent*, CN: *Correspondent Node* e FA: *Foreign Agent*). Deste modo criamos 4 projetos com os seguintes nomes:

* 1MRCMN
* 2MRCHA
* 3MRCCN
* 4MRCFA

Existe uma package comum a todos os projetos que se chama “*common*” que tem a classe “Utils”, classe essa onde estão definidos os IPs e as Portas dos vários Agentes.

As principais classes criadas foram:

* BindingUpdate.java -> Classe que representa o pedido de otimização de rotas por parte do MN para o CN.
* Pacote.java -> Classe que representa um pacote no programa. Tem como atributos o “*source*”, o “*destination*” e o “data”.
* PedidoCancelamentoRegisto.java -> Classe que representa, como o próprio nome indica um pedido de cancelamento de registo.
* PedidoRegisto.java -> Classe que representa um pedido de registo numa rede.

Classes específicas do MN:

* ListeningForPackets.java -> Classe responsável pela criação do *socket* e de uma *thread* para fazer a comunicação de pacotes.
* MNGUI.java -> Classe responsável pela criação da interface gráfica do *Modility* Node.
* NetworkMonitoring.java -> Classe que cria uma *Thread* para monitorizar as redes disponíveis.

Classes específicas do HA:

* AnuncioRede.java -> Classe responsável pela propagação da rede.
* HomeAgentGUI.java -> Classe responsável pela criação da interface gráfica do *Home* *Agent*.
* MobilityBindingTableEntry.java -> Classe que representa cada uma das entradas da MBT.
* TemporizadorTTL.java -> Classe responsável pelo *TimeOut* dos elementos da MBT.

Classes específicas do CN:

* CNGUI.java -> Classe responsável pela interface gráfica do *Correspondent* Node.
* BindingEntry.java -> Classe responsável pelo objeto a enviar para otimização de rotas.
* ListeningForPackets.java -> Classe responsável pela criação do *socket* e de uma *thread* para fazer a comunicação de pacotes.

Classes específicas do FA:

* AnuncioDeRede.java -> Classe responsável pela propagação da rede.
* FAGUI.java -> Classe responsável pela interface gráfica do *Foreign* *Agent*.
* TemporizadorTTL.java -> Classe responsável pelo *TimeOut* dos elementos da *Visitor* *Table*.
* VisitorListEntry.java -> Classe que representa cada uma das entradas da MBT.

1. **Descrição dos principais problemas e de como os mesmos foram resolvidos**

Os principais problemas que tivemos na implementação deste projeto estão diretamente ligados à arquitetura a escolher/escolhida, uma vez que não possuíamos conhecimentos relativamente à melhor arquitetura a escolher, mas após a exposição desta dúvida com o professor, ficamos a perceber qual seria a melhor opção a tomar.

Tivemos também alguns problemas relativamente à melhor forma de simular os diversos cenários possíveis, tendo em especial atenção ao *Mobility* Node, mas após algum tempo de raciocínio e diálogo com outros colegas, esclarecemos esta dúvida.

1. **Conclusão**

Com o desenvolvimento deste simulador, podemos aplicar diversos conceitos abordados nas aulas teóricas desta mesma cadeira, assim como de outras cadeiras, como por exemplo: Sistemas Distribuídos. Ficamos a perceber o funcionamento da Mobilidade IP, e de como os pacotes são reencaminhados entre redes distintas, assim como também adquirimos conhecimentos relativamente à otimização de rotas no encaminhamento dos pacotes.