Gestão de Elevadores

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Agentes e Inteligência Artificial Distribuída
4º Ano 1º Semestre

Grupo T01_3:
Luís Barbosa - 201405729 - up201405729@fe.up.pt
Paulo Santos - 201403745 - up201403745@fe.up.pt
Sérgio Almeida - 201403074 - up201403074@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

5 de novembro de 2017

Conteúdo

1 Enunciado

1.1 Descrição do cenário

O cenário do presente trabalho consiste num edifício de vários andares com diversos elevadores. Cada elevador possui uma carga máxima, que corresponde ao número máximo de pessoas que pode transportar.

Os elevadores comunicam entre si informação relevante. Quando há um novo pedido numa determinada zona do edifício, este deve ser alocado a um dos elevadores existentes naquela zona. Os elevadores possuem uma lista de pisos em que têm que parar, podendo a mesma ser alterada dinamicamente para se incluirem novos pedidos.

Periodicamente, os elevadores podem partilhar informação sobre os seus estados, no sentido de poderem atribuir tarefas a outros elevadores.

O programa deve permitir que o utilizador configure o sistema, como o número de pisos do edifício, número de elevadores e a carga máxima de cada elevador.

As necessidades dos utentes, como por exemplo a frequência de chamada de elevador em cada piso do edifício (devendo ser superior para o piso 0) e piso de destino, são geradas aleatoriamente.

Um elevador demora um tempo pré-definido a ir de um piso a outro, e o intervalo de tempo correspondente à paragem do elevador para entrada/saída de utentes também é contabilizado.

1.2 Objetivos do trabalho

O trabalho tem como principal objetivo o desenvolvimento de um sistema multi-agente para a gestão eficiente dos elevadores de um determinado edifício, tendo em conta o cenário descrito acima. Para tal, será necessária a implementação de um algoritmo que permita gerir as chamadas aos elevadores de forma a que o transporte seja feito o mais rápido possível.

1.3 Resultados esperados e forma de avaliação

O programa irá apresentar estatísticas relativas ao desempenho dos vários elevadores, relativas ao tempo de espera máximo e mínimo, taxa de ocupação do elevador e tempo de uso/não uso do elevador.

Como forma de avaliação, comparar-se-á o desempenho deste sistema multi-agente com outras estratégias de cooperação, como por exemplo um sistema tradicional em que cada elevador possui uma estratégia fixa e individual: atende o pedido o elevador que se encontre num piso mais próximo de onde foi feita a chamada. Também serão usados como forma de comparação sistemas em que existam dois botões de chamada vs teclado para indicação de piso de destino.

2 Plataforma/Ferramenta

A ferramenta escolhida para o desenvolvimento deste trabalho foi JADE.

2.1 Para que serve

JADE é uma *framework* que permite o desenvolvimento de sistemas multi-agente.

2.2 Descrição das caraterísticas principais

O JADE é compatível com FIPA, isto é, segue os padrões de software estabelecidos para agentes heterogéneos e interativos e para sistemas baseados em agentes.

Possui um *Agent Management System* que supervisiona o acesso e uso da plataforma, mantém um diretório com os identificadores dos agentes e do seu estado e faz com que cada agente esteja registado neste sistema. Além deste, possui um *Directory Facilitator*, que funciona como um serviço de páginas amarelas.

A plataforma pode ter vários *containers*, onde se encontram os agentes. Estes podem estar em máquinas diferentes. O *main container* inclui os serviços descritos anteriormente. Os agentes podem migrar entre *containers*.

A comunicação pode ser feita através de notificações, RMI ou IIOP.

O JADE possui várias ferramentas que simplificam a administração da plataforma e o desenvolvimento:

- RMA (Remote Monitoring Agent): Consola gráfica que permite visualizar a atividade da plataforma, bem como visualizar os *containers* e o estado dos agentes.
- Dummy Agent: Permite interação com outros agentes, através do envio, receção e visualização de mensagens ACL.
- Sniffer Agent: Visualizar mensagens enviadas por/para o agente num diagrama UML. Este é informado pelo AMS quando um agente ou *container* é criado/destruído.
- Directory Facilitator GUI: Permite que os agentes registem o seu serviço ou procurem neste.
- Introspector Agent: Monitoriza o ciclo de vida de um agente.

Nesta *framework*, os agentes possuem *behaviours*, que implementam as tarefas dos agentes.

2.3 Realce das funcionalidades relevantes para o trabalho

De momento, e com base no código que já temos desenvolvido, é relevante o *Directory Facilitator*, pois permite que os agentes se registem num serviço já existente.

O facto de os agentes terem *behaviours* permite-nos implementar a lógica necessária para cumprir os objetivos do trabalho.

Como os agentes necessitam de partilhar informação sobre os seus estados, a comunicação através de mensagens ACL também será importante.

Para no fim conseguirmos avaliar os resultados, as ferramentas enumeradas anteriormente também serão importantes para comparar diferentes implementações do sistema multi-agente.

3 Especificação

3.1 Identificação e caracterização dos agentes

No nosso projeto, distinguem-se dois tipos de agentes: *Elevator* e *Building*.

O *Building* é um tipo de agente que fornece um serviço. Este possui um número de pisos, entre os quais um certo número de elevadores funciona. Este é responsável pela criação de outros agentes, os *Elevator*. Este agente pretende simular, através do seu comportamento, os pedidos existentes num edifício por parte dos utilizadores. Desta forma, cria pedidos num certo intervalo de tempo, enviando uma mensagem ACL do tipo *INFORM* para os elevadores.

O *Elevator* é o agente que se regista no serviço fornecido pelo *Building*. Possui um peso máximo, um tempo de movimento entre pisos e o seu piso atual. Cada *Elevator* tem uma lista interna de pedidos a realizar, que é dinâmica, uma vez que pode ser alterada caso outros elevadores lhe atribuam um novo pedido, ou caso seja feito um novo pedido no *Building*. O seu comportamento consiste em satisfazer o próximo pedido da lista, simulando o tempo de movimento até esse piso.

3.2 Protocolos de interação

Como protocolo de interação, pretendemos utilizar objetos da classe *ACLMessage* com *performative* relacionado com o tipo de informação enviada (proposta, informação, aceitação de proposta...) e conteúdo correspondente a uma *string* com o piso onde foi feito o pedido e o destino previsto, que pode ser desconhecido, uma direção ou um piso de destino.

Exemplos do conteúdo possível de uma mensagem: "5 UNKNOWN"(pedido no piso 5 com destino desconhecido), "5 UP"(pedido no piso 5 com destino previsto para cima), "5 DOWN"(pedido no piso 5 com destino previsto para baixo), "5 4"(pedido no piso 5 com piso 4 como destino).

3.3 Faseamento do projeto

O trabalho será desenvolvido a partir dos seguintes pontos (os três primeiros já se encontram implementados):

1. O agente "Building"deve alocar pedidos a elevadores sem possibilidade de recusa por parte do elevador.
2. Configuração do sistema por parte do utilizador: número de pisos do edifício, número de elevadores e carga máxima de cada elevador.
3. Elevador deve demorar um tempo pré-definido a ir de um piso a outro.
4. Limitar o peso que o elevador pode transportar.
5. "Building"alocar pedidos a elevadores com possibilidade de recusa por parte do elevador.
6. Elevadores conseguirem comunicar entre si os seus estados para atribuir tarefas a outros elevadores.
7. Melhorar a ordenação interna dos pedidos nos elevadores de forma a tornar a resposta a estes mais eficiente.
8. Simular a saída das pessoas do elevador (demorando algum tempo) e receber piso de destino (até agora só recebe piso de entrada).
9. Contabilizar o tempo correspondente à paragem do elevador para entrada/saída de utentes.

10. Apresentar estatísticas relativas ao desempenho dos elevadores (tempos de espera máximo e mínimo, taxa de ocupação do elevador, tempo de uso e não uso do elevador).

11. Permitir dois botões de chamada (subir/descer) ou teclado para indicação do piso destino.

12. Comparar o desempenho com diferentes estratégias de cooperação.

4 Recursos

4.1 Bibliografia

Slides do Moodle sobre JADE
Documentação do JADE
Código fonte do JADE

4.2 Software

JADE (http://jade.tilab.com/)
Eclipse IDE for Java
IntelliJ IDEA Community Edition