Agentes e Inteligência Artificial Distribuídaaaaaaa

up201604503 Tomás Nuno Fernandes Novo
up201604828 João Pedro Viveiros Franco
up201604735 Christopher Fernandes de Abreu

1ª Parte

Apresentação

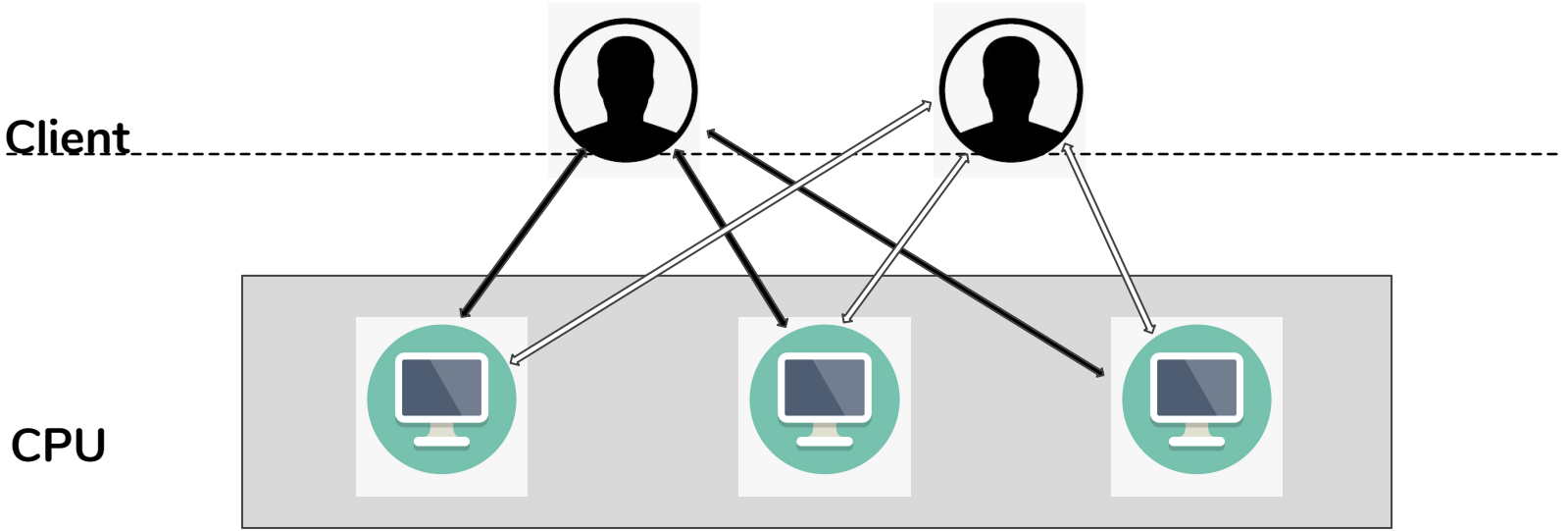
Descrição do Problema

- No âmbito desta unidade curricular, foi-nos proposta como 1ª parte do trabalho prático a realização de um projeto cuja meta residia em implementar um sistema multiagente.

- Descrevendo um pouco as idiossincrasias do problema implementado, no nosso sistema de múltiplos agentes existem **Clientes** que possuem um **projeto por compilar** e necessitam desse projeto compilado até um determinado **deadline**. Por sua vez, existem **CPU's** que têm o fim de receber um projeto de um determinado Cliente, compilar os ficheiros que lhe competem e enviar de volta para o cliente os ficheiros compilados. Estes agentes atuam em cooperação com o objetivo de **diminuir o tempo total de compilação** do projeto do Cliente. É estabelecida negociação entre os agentes caso os CPU's se apercebam que o deadline não vai conseguir ser comprido. Cada agente cliente tem tolerância relativamente às negociações estabelecidas, podendo cancelá-las ou aceitá-las.

Esquema Global

Na realização do projeto, foram considerados 2 tipos de agentes: **Cliente** e **CPU,** que interagem da seguinte maneira:

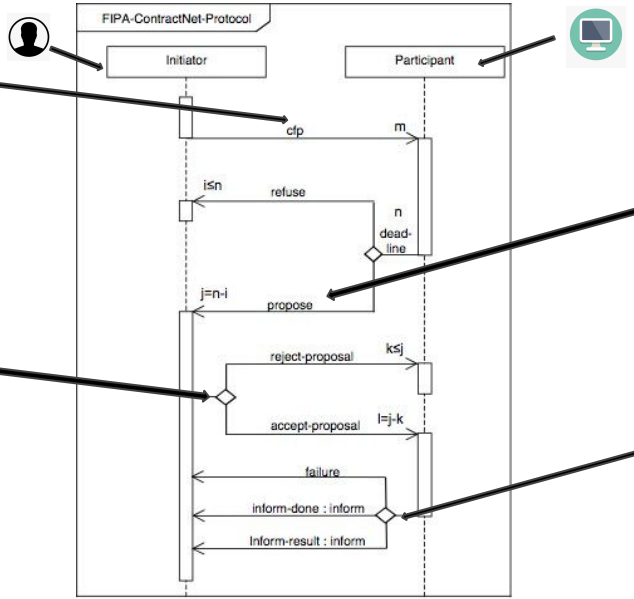


Interação e Protocolos

As interações entre os agentes, quer seja do Cliente para o CPU ou o contrário, são baseadas na troca de **ACL Messages**. O envio e receção destas mensagens é realizado recorrendo ao protocolo **Contract Net**.

Mensagem que contém a descrição do projeto

Mensagem que comunica a decisão quanto à aceitação do novo deadline



Mensagem que informa se o deadline foi aceite OU Mensagem com proposta novo deadline

Envio dos ficheiros compilados

Arquiteturas e Estratégias Utilizadas

Cliente:

Aceitação do deadline proposto- Na negociação, o agente Cliente possui um atributo tolerância(%) que é gerado de forma aleatória na criação de um Agente deste tipo. Aquando da recepção da negociação com o CPU, o agente aceitará a negociação caso se verifique a condição:

DeadlinePropostaCPU <= DeadlineInicial+ Tolerância * Deadline Inicial

CPU:

Verificação se o deadline é exequível - Para verificar se o deadline recebido do cliente é exequível, o CPU começa por calcular o seu tempo médio de compilação por tamanho de ficheiro compilado, recorrendo a anteriores compilações, e, posteriormente verifica a condição **CompilationTimePredicted <= Deadline.** Caso esta condição se verifique, ou caso não haja compilações anteriores, o deadline é aceite e os ficheiros prontos para compilar.

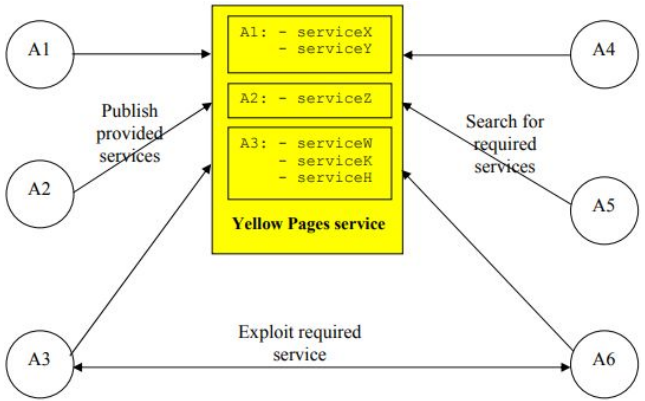
Compilação dos ficheiros - na primeira interação Cliente->CPU, o Cliente cria um objeto da classe ProjectInfo que contém todas as informações relativas ao seu projeto e converte os ficheiros .cpp ou .h recebidos para objetos CompilationFile. Cada CPU recebe um ProjectInfo com ficheiros para compilar diferentes. Por sua vez, os ficheiros são compilados recorrendo à classe Process que faz uma chamada ao comando g++ pelo sistema operativo.

Software Utilizado e Outros Mecanismos

Para a realização deste projeto:

- Foi utilizado o IDE Eclipse

- Usufruímos da framework Jade



- Foi implementada uma estrutura de descoberta de agentes DF (Directory Facilitator), com o objetivo de os Clientes encontrarem os CPUs.

Experiências Realizadas

- Com o fim de testar o nosso projeto, foram realizadas determinadas experiências que contribuíram para que a consistência do nosso projeto aumentasse. A principal experiência consistiu na distribuição dos ficheiros por parte do Cliente, sendo que esta repartição foi testada:

  - Atribuindo os ficheiros a cada CPU **aleatoriamente**.
  - Atribuindo os ficheiros a cada CPU **aleatoriamente** de forma a que o nº de ficheiros a compilar **não varie por mais do que 1** entre 2 CPUs distintos
  - Atribuindo os ficheiros aos CPUs de maneira a que o **tamanho total** dos ficheiros a compilar seja **igualmente repartido.**

Análise dos Resultados

- Tirando ilações das experiências realizadas, concluímos que efetivamente existe uma redução significativa do tempo total de compilação quando os ficheiros são distribuídos por tamanho. Verificou-se também que o objetivo de **diminuir o tempo total de compilação** do projeto foi alcançado.

Por tamanho

```
INFO    | CPU1  | Successfully compiled project "neuro" in .86 seconds
INFO    | CPU1  | Successfully compiled project "aeda" in 1.32 seconds
```

Aleatório Balanced

```
INFO    | CPU1  | Successfully compiled project "neuro" in .93 seconds
INFO    | CPU1  | Successfully compiled project "aeda" in 2.13 seconds
```

Aleatório

```
INFO    | CPU2  | Successfully compiled project "neuro" in 1.34 seconds
INFO    | CPU2  | Successfully compiled project "aeda" in 2.82 seconds
```

Conclusões

Em suma, a realização deste projeto foi benéfico para os elementos constituintes do grupo devido à aquisição de conhecimentos acerca do funcionamento de um sistema composto por múltiplos agentes.

A implementação do projeto encontra-se em concordância com o acordado com o professor e foi efetuada com sucesso, pelo que achamos que a sua concepção foi bastante positiva.

No próximo trabalho objetivamos primeiramente melhorar um pouco o funcionamento do nosso sistema e posteriormente realizar uma análise mais detalhada acerca do funcionamento do nosso sistema através da realização de novas experiências e aumentar o leque de ficheiros que podem ser compilados pelos CPU's, como por exemplo ficheiros de C ou Java.

2ª Parte

Informação Adicional

Exemplos detalhados de Execução

O Cliente inicia e a sua tolerância é gerada de forma aleatória

```
INFO    | Client1    | Tolerance: 0.93
INFO    | Client2    | Tolerance: 0.8
```

O Cliente procura os CPUs na DF e caso encontre este tipo de agentes, envia-lhes uma proposta com ficheiros do seu projeto e deadline respetivo

```
INFO    | CPU1  | Received a proposal from "Client1" for project "aeda"
INFO    | CPU1  | Received a proposal from "Client2" for project "neuro"
```

Exemplos detalhados de Execução

Os CPUs verificam recorrendo ao seu tempo médio de compilação se é possível compilar os ficheiros que lhe competem no deadline pretendido pelo Cliente e comunicam ao Cliente o resultado da verificação, aceitando o deadline ou propondo um novo deadline

```
INFO    | CPU1   | CPU sending PROPOSE: Deadline is acceptable
INFO    | CPU2   | Received deadline: 6701 ms, proposing 9804ms
```

O Cliente pode então ser informado que o deadline é exequível e os ficheiros estão prestes a ser compilados (1) ou pode receber uma proposta de um novo deadline por parte do CPU que será aceite ou rejeitado de acordo com a tolerância do Cliente (2)

1

```
INFO    | Client2    | CPU1 accepted the deadline
INFO    | Client2    | CPU2 accepted the deadline
```

2

```
INFO    | Client1    | CPU1 accepted the deadline
INFO    | Client1    | Client received PROPOSE from CPU2: 10909ms
INFO    | Client1    | Client accepted new deadline from CPU2
```

Exemplos detalhados de Execução

Se o deadline for exequível ou se o Cliente aceitar o novo deadline proposto, o CPU compila os ficheiros que lhe competem e envia-os para o Cliente

```
INFO    | CPU1   | Successfully compiled menu.cpp
INFO    | CPU1   | Successfully compiled data.cpp
INFO    | CPU1   | Successfully compiled utilities.cpp
INFO    | CPU1   | Successfully compiled project "aeda" in 1.32 seconds
INFO    | CPU1   | CPU sending INFORM: CompiledProject aeda
```

Posteriormente, o Cliente recebe os ficheiros compilados e faz link do projeto caso a compilação seja efetuada sem erros e sem exceder o deadline

Sucesso

```
INFO    | Client1    | Successfully received CompiledProject from CPU1
INFO    | Client1    | Successfully received CompiledProject from CPU2
INFO    | Client1    | Successfully built project aeda
```

Insucesso

```
INFO    | Client2    | Successfully received CompiledProject from CPU2
ERROR   | Client2    | CPU1 exceeded the deadline by 0.2659248 seconds!
ERROR   | Client2    | Failed to receive the entire project successfully!
```

Classes Implementadas

Neste projeto, foram implementadas as seguintes classes:

- **ExtendedAgent** - classe com atributos comuns aos agentes Cliente e CPU que é usada com o fim de permitir a visualização das ações de cada agente
- **Cliente -** agente que pretende compilar um projeto
- **CPU** - agente que compila ficheiros de um projeto de um Cliente
- **Bid -** classe que auxilia nas negociações e que usa deadlines com o fim de as licitar
- **ProjectInfo** - classe que armazena as informações de um projeto de um Cliente
- **CompilationFile** - classe relativa aos ficheiros provenientes do Cliente
- **CompiledProject -** classe que representa um projeto compilado por um CPU
- **Pair -** classe auxiliar que constrói um par de elementos
- **Macros -** classe auxiliar que possui Macros utilizadas nos diversos ficheiros do projeto

Classes Implementadas: Cliente

Atributos

- **cpus -** CPUs disponíveis
- **info -** do tipo ProjectInfo. Contém informação sobre o projeto do Cliente
- **tolerance -** float(%) que representa a tolerância de um cliente
- **deadline -** do tipo Bid, contém o valor do deadline inicial do projeto do Cliente
- **totalCompilationTime -** tempo de compilação estimado do projeto

Behaviours

- **OfferProjectBehaviour -** behaviour que verifica os CPUs registados na DF e chama o outro behaviour do Cliente para principiar as negociações
- **NegotiationInitiator -** behaviour que extende o protocolo ContractNetInitiator e é responsável pela interação do Cliente na negociação

Classes Implementadas: CPU

Atributos

- **clientAID-** AID do Cliente com o qual é estabelecida a negociação
- **compilationTimes-** contém o tempo que cada CPU demorou a compilar um ficheiro. Usado para verificar a exequibilidade da compilação dos ficheiros do Cliente no deadline inicialmente recebido
- **acceptableDeadline-** boolean que indica se o deadline

Behaviours

- **NegotiationResponder -** behaviour aplica o protocolo **ContractNetResponder** e que verifica se a compilação é possível. Posteriormente, comunica a sua resposta ao Cliente, informando se é possível compilar no tempo desejado ou propondo um novo deadline, ficando à espera da resposta do Cliente. Quando esta é recebida, no caso de a negociação ser aceite, os ficheiros são compilados e enviados para o Cliente.

Classes Implementadas: ProjectInfo

Atributos

- **name -** nome do projeto
- **deadline -** contém a deadline do projeto
- **files -** ArrayList que contém ficheiros do projeto
- **toBeCompiled-** usado pelo CPU. Representa índices do ArrayList **files** que o CPU ficou encarregado de compilar