Universidade Federal de Santa Catarina
Centro Tecnológico
DAS410058-41 - **Aprendizado de Máquina -** 2020/3
Departamento de Automação e Sistemas
Professor: Jomi F. Hübner
Trabalho prático - Contract Net Protocol
Acadêmica: Lara Popov Zambiasi Bazzi Oberderfer

Contract Net Protocol: uma comparação entre as linguagens Jade e Jason

Objetivo

O objetivo do trabalho é avaliar o uso de agentes e suas linguagens para implementar o *Contract Net Protocol*.[1]

Detalhes

Implemente um SMA com n $(1 < n < 200)$ agentes que serão *initiators* e m $(1 < m < 50)$ agentes que serão *participants*.

Cada *initiator* contrata ao mesmo tempo i $(0 < i < 10)$ serviços, i.e. executa i CNPs em paralelo.

Cada *participant* oferece um único tipo de serviço (com estratégias de proposta diferenciadas). As estratégias de preço e detalhes das propostas podem ser definidas durante o trabalho.

Faça uma implementação em JADE e outra em Jason. Opcionalmente, no lugar do JADE, pode ser escolhila outra linguagem de programação de agentes, como o ASTRA, SARL, 2APL, GOAL, ...

O trabalho pode ser feito em dupla.

Relatório

Conforme o cronograma da disciplina, deve ser entregue um relatório descrevendo os resultados do trabalho.

Procure incluir:

- Uma análise das variações de n, m e i. A definição das métricas para a análise faz parte do trabalho.
- Uma avaliação da abordagem de agentes para a solução do problema.
- Uma avaliação das ferramentas (JADE, Jason, ...) na implementação da solução.
- A definição de critério de comparação entre as implementações.
- A comparação entre as implementações.

Introdução

Os objetivos deste trabalho são a implementação de um Protocolo de Redes de Contrato (*Contract Net Protocol* - CNP) em duas linguagens de programação de Sistemas Multi-Agentes (SMA): JADE e Jason; avaliar a abordagem dos agentes para a solução do problema; avaliar as ferramentas na implementação da solução; e realizar uma comparação entre as implementações.

Agentes podem ser definidos como um sistema de computador baseado em hardware ou software que desfruta das propriedades de: autonomia, capacidade social, reatividade e pró-atividade. Nesse sentido, os SMAs consistem de uma rede de vários agentes que compõem uma organização que se comunica de alguma forma interagindo entre si (WOOLDRIDGE; JENNINGS, 1998).

Wooldridge e Jennings (1997) em sua definição de agentes, listam as seguintes características: reatividade é perceber o ambiente e responder de maneira oportuna (timely fashion) a mudanças que acontecem nele; autonomia é a capacidade de operar sem intervenção direta de humanos ou outros e ter alguma espécie de controle sobre suas ações e estado interno; habilidade social é a capacidade de interagir com outros agentes via algum tipo de linguagem de comunicação de agentes; pró-atividade é a capacidade de exibir comportamentos baseados em objetivos.

O SMA foi programado em uma versão estendida da linguagem AgentSpeak(L), por meio do interpretador Jason, baseado em sistemas de crenças, desejos e intenções (BDI - *Beliefs, Desires and Intentions*) e no *framework* de software JADE, para programação orientada a agente capaz de atender as especificações FIPA (TEIXEIRA, 2010).

Modelagem de um Contract Net Protocol

O CNP é um protocolo de abordagem de resolução distribuída de problemas, uma versão padronizada do protocolo foi desenvolvida pelo FIPA, que é uma entidade internacional que promove a indústria de agentes inteligentes desenvolvendo especificações para aplicações baseadas em agentes. Os agentes fazem parte de um sistema multi-agente. Eles têm que realizar tarefas específicas e podem pedir a outros agentes para realizar subtarefas para eles.

Conforme mostra a Figura 1 (FIPA, 2002) o Iniciador solicita m propostas de outros agentes emitindo um ato de chamada de propostas (cfp), que especifica a tarefa, bem como quaisquer condições que o Iniciador está colocando na execução da tarefa. Os participantes que recebem a chamada de propostas são vistos como contratantes em potencial e podem gerar n respostas. Destas, j são propostas para realizar a tarefa, especificadas como atos de proposição.

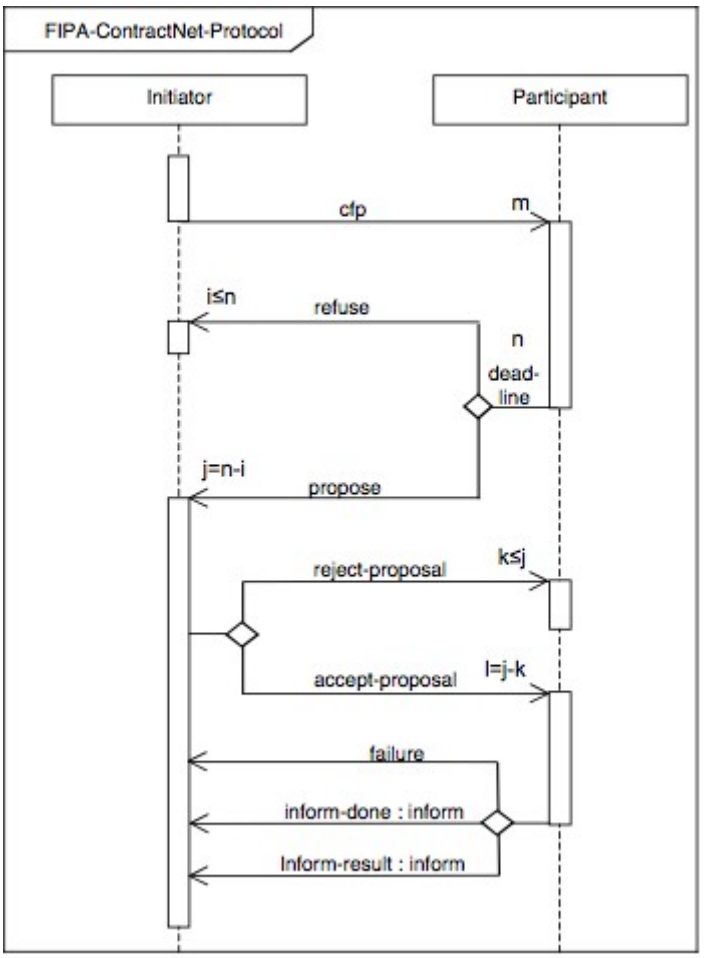

Figura 1: Contract Net Interaction Protocol da FIPA (FIPA, 2002)

A proposta do Participante inclui as pré-condições que o Participante está estabelecendo para a tarefa, que podem ser o preço, o tempo em que a tarefa será realizada, etc. Alternativamente, os participantes i = nj podem recusar a propor. Passado o prazo, o Iniciador avalia as propostas j recebidas e seleciona os agentes para realizar a tarefa; um, vários ou nenhum agente pode ser escolhido. Os i agentes da proposta selecionado(s) irá ser enviado um aceitar-proposta ato e os restantes k agentes receberá uma rejeitar-proposta agir. As propostas são vinculativas para o Participante, de forma que uma vez que o Iniciador aceita a proposta, o Participante adquire o compromisso de realizar a tarefa. Uma vez que o Participante tenha concluído a tarefa, ele envia uma mensagem de conclusão ao Iniciador na forma de um informe-feito ou uma versão mais explicativa na forma de um informe-resultado. No entanto, se o participante não conseguir concluir a tarefa, uma mensagem de falha será enviada.

Observe que esse IP exige que o iniciador saiba quando recebeu todas as respostas. No caso de um participante não responder com uma proposta ou um ato de recusa , o Iniciador pode ficar esperando indefinidamente. Para se proteger contra isso, a lei

cfp inclui um prazo no qual as respostas devem ser recebidas pelo Iniciador. As propostas recebidas fora do prazo são automaticamente rejeitadas com o motivo de o atraso ser apresentado. O prazo é especificado pelo parâmetro de resposta na mensagem ACL.

Qualquer interação usando este protocolo de interação é identificada por um parâmetro de id de conversação não nulo globalmente exclusivo , atribuído pelo Iniciador. Os agentes envolvidos na interação devem marcar todas as suas mensagens ACL com este identificador de conversação. Isso permite que cada agente gerencie suas estratégias e atividades de comunicação, por exemplo, permite que um agente identifique conversas individuais e raciocine através de registros históricos de conversas.

No caso de protocolos ou sub-protocolos de interação 1: N, o iniciador é livre para decidir se o mesmo parâmetro de id de conversação deve ser usado ou se um novo deve ser emitido. Além disso, as mensagens podem especificar outras informações relacionadas à interação, como um tempo limite no parâmetro de resposta que denota a hora mais recente em que o agente de envio gostaria de ter recebido a próxima mensagem no fluxo de protocolo.

O SMA escolhido para esta aplicação no CNP consiste em 50 restaurantes (*iniciators*) oferecendo seus cardápios (pratos) e 200 clientes (*participant*) desejando pedir comida (pratos) diferentes.

Modelagem do SMA no JASON

A Figura 3 apresenta a criação do projeto Restaurante criado em Jason e a configuração dos agentes.

```
1  MAS restaurantes {
2      infrastructure: Centralised
3      agents:
4          restaurante #50;   // CNP initiator
5          cliente #200;      // participants
6          clienteRecusa;     // um participante sempre recusa
7          clienteN;          // um participante não responde
8  }
```

Figura 3: Projeto CNP Restaurantes em JASON

A estrutura da criação de um SMA em Jason (Figura 3) se apresentou muito simples, em 8 linhas criamos os 250 agentes, no qual após o nome do agente segue o símbolo “#” e após esse símbolo é indicado o número desejado de agentes de cada tipo. Ainda dois agentes terão tratamento diferenciado nas ações, um sempre recusa a negociação e outro sempre não responde.

```
Crença:
    comida(sushi)

Objetivo:
   !setup.

Plano
+!setupComida([Head|Tail]) <-
        ?comida(Head);
        .random(R);
        -priceCurrentComida(_);
        +priceCurrentComida(math.floor((R*100)+20));
        ?priceCurrentComida(Tot);
        .print("Restaurante vendendo ",Head,
         " por no minimo R$ ", Tot, " reais");
        !cnp(math.floor((R*100)),Head);
        !setupComida(Tail).
```

Figura 4: Exemplos de Crenças, objetivos e planos do restaurante (*iniciator*)

A Figura 4 apresenta trechos do código-fonte do restaurante (*iniciator*) contendo exemplos do BDI: a crença da Comida sushi, o objetivo setup, e um plano referente a comida que mostra o prato que o restaurante está tentando vender e um valor mínimo que pede.

```
[restaurante] Vendo:
[sushi,mexicana,burguer,fritas,pizza,massa,arabe,churrasco,camarao,tilapia]
[cliente2] Tenho R$ 1540 reais
[cliente1] Tenho R$ 1523 reais
[restaurante] Restaurante vendendo sushi por no minimo R$ 58 reais
[restaurante] Esperando clientes para pedir sushi...
[restaurante] Enviando CFP 38 para [clienteN,cliente1,clienteRecusa,cliente2]
[restaurante] As ofertas dos clientes foram [offer(100,cliente1),offer(100,cliente2)]
espero pelo menos R$ 58 reais
[restaurante] O cliente2 venceu o pedido por R$ 100 reais
[restaurante] Saldo: R$ 100 reais
[cliente2] Ganhei o CNP 38. Minha proposta foi de R$ 100 reais para sushi!
[cliente1] Perdi o CNP 38.
[cliente2] Comprei sushi por R$ 100 reais. Meu saldo ficou R$ 1440 reais
```

Figura 5: Compilação do Jason com 1 restaurante e 2 clientes

A Figura 5 apresenta a compilação do programa em Jason com 1 restaurante e 2 clientes para ilustrar a modelagem do SMA do restaurante, no qual os clientes apresentam seus valores iniciais, o restaurante oferece um prato por um valor mínimo e envia o CFP para os clientes, os clientes fazem suas ofertas e a melhor oferta vence, o saldo do restaurante é incrementado e o saldo do cliente é decrementado.

Modelagem do SMA em JADE

Para a modelagem no projeto JADE, tivemos que criar a classe de criação dos agentes, no qual, para cada agente iniciador precisa-se estruturar o seguinte código para a criação dos agentes participantes:

```
-agents iniciator1:demo.ContractNetInitiatorAgent("resp1","resp2");
resp1:demo.ContractNetResponderAgent;resp2:demo.ContractNetResponderAgent -gui
```

A Figura 5 apresenta a criação do projeto Restaurante criado em JADE e a configuração dos agentes.

```
1   public class StartJade {
2       public static void main(String[] args) throws Exception {
3          int nIniciator = 50;     //total de iniciators
4          int nResponders = 200; //total de responders
5          List aux1 = new ArrayList();
6          List aux2 = new ArrayList();
7          for (int i=1; i<=nResponders; i++) {
8                 aux1.add("cliente"+i+"");
9                 aux2.add("cliente"+i+":demo.ContractNetResponderAgent");
10         }
11         String aux3 = String.join(",", aux1);
12         String aux4 = String.join(";", aux2);
13         String aux5 = "";
14         for (int i=1; i<=nIniciator; i++) {
15                aux5 =
16  aux5+"restaurante"+i+":demo.ContractNetInitiatorAgent("+aux3+");";
17         }
18         aux5 = aux5+aux4;
19         String[] parametros = {"-gui", "-localhost", "127.0.0.1", aux5,};
20         jade.Boot.main(parametros);
21      }
    }
```

Figura 6: Projeto CNP Restaurantes em JADE

A estrutura da criação de um SMA em JADE (Figura 6) se apresentou mais complexa, em 22 linhas criamos os 250 agentes.

```
private void buildAgents() {
        availabilityMoney = getRandomVar(400,2000);
        Interests.add(new Interest("sushi",1));
        Interests.add(new Interest("mexicana",2));
        Interests.add(new Interest("burguer",3));
        Interests.add(new Interest("fritas",4));
        Interests.add(new Interest("pizza",5));
        Interests.add(new Interest("massa",6));
        Interests.add(new Interest("arabe",7));
```

```
        Interests.add(new Interest("churrasco",8));
        Interests.add(new Interest("camarao",9));
        Interests.add(new Interest("tilapia",10));
    }
```

Figura 7: Trecho de código-fonte do programa em JADE do restaurante (iniciator)

A Figura 7 apresenta trecho do código-fonte do cliente (*responder*) contendo o método que constrói um agente cliente contendo um valor inicial (randômico) que ele pode gastar nas suas compras de alimentos e os interesses, pratos que ele pode adquirir (crenças).

```
Cliente: cliente1 Tenho R$ 636.0 reais. Interesses : sushi,mexicana,burguer,fritas,pizza,massa,
arabe,churrasco,camarao,tilapia, aguardando por CFP...
Cliente: cliente2 Tenho R$ 882.0 reais. Interesses : sushi,mexicana,burguer,fritas,pizza,massa,
arabe,churrasco,camarao,tilapia, aguardando por CFP...
Restaurante restaurante1@192.168.1.113:1099/JADE vende {sushi , 30.3} para um dos 2 clientes
Restaurante restaurante2@192.168.1.113:1099/JADE vende {sushi , 30.3} para um dos 2 clientes
CFP: 1731986790: Cliente cliente2 Propos {sushi, R$ 30.603} para C142342681_restaurante2
CFP: 1421377711: Cliente cliente1 Propos {sushi, R$ 30.603} para C142342681_restaurante2
Aceitando a proposta R$ 30.603 do cliente cliente2@192.168.1.113:1099/JADE
CFP: 1421377711: Cliente cliente1: recusou a proposta do C142342681_restaurante2
CFP: 1731986790: Restaurante C142342681_restaurante2: aceitou a proposta do cliente2
CFP: 177876345 Cliente: cliente2 ganhou o CNP. Proposta foi de R$ 30.603
Cliente cliente2@192.168.1.113:1099/JADE realizou a compra.
Excluindo item: sushi
Proximos pratos do Restaurante restaurante2@192.168.1.113:1099/JADE
Prato mexicana Valor R$ 70.5
Prato burguer Valor R$ 15.2
Prato fritas Valor R$ 10.2
Prato pizza Valor R$ 50.3
Prato massa Valor R$ 40.5
Prato arabe Valor R$ 20.3
Prato churrasco Valor R$ 80.3
Prato camarao Valor R$ 60.3
Prato tilapia Valor R$ 20.3
```

Figura 8: Compilação do JADE com 1 restaurante e 2 clientes

A Figura 8 apresenta a compilação do programa em JADE com 2 restaurante e 2 clientes para ilustrar a modelagem do SMA do restaurante, no qual os clientes apresentam seus valores iniciais, e seus interesses aleatórios de comida, o restaurante oferece um prato por um valor mínimo e envia o CFP para os clientes, os clientes fazem suas ofertas e a melhor oferta vence, o saldo do restaurante é incrementado e o saldo do cliente é decrementado.

Resultados e discussões

Nesta seção seguem algumas comparações entre as implementações Jason e JADE.

Tabela 1: Comparações de linhas de código dos programas Jason e JADE

Arquivos implementados	**Jason**	**JADE**
SMA	8	21
Restaurante / Iniciator	100	171
ClienteRecusa	6	-
ClienteN	2	-
PerformRequest	-	90
Cliente / Respondent	-	130
PerformResponse	44	100
Classe Comida	-	42
Classe Interesses	-	19
Total de Linhas de código	160	573

A Tabela 1 apresenta as comparações de linhas de código das implementações do Jason e do JADE, como pode-se perceber o Jason necessita de menos arquivos e muito menos linhas de código comparado ao JADE. A diferença é significativa: são praticamente 413 linhas de código a menos que o Jason precisou para o mesmo programa funcionar.

Tabela 2: Compilação dos programas Jason e JADE

Restaurantes	**Clientes**	**JASON**	**JADE**
1	20	2 min 00 seg	11 segundos
1	200	2 min 00 seg	11 seg e 26 ms
5	200	2 min 00 seg	10 seg e 334 ms
6	200	2 min 00 seg	não teve retorno*
50	200	3 min 46 seg	não teve retorno

A partir de 6 iniciators o JADE mostrou-se que não conseguiu trabalhar com os agentes em paralelo como mostra o trecho na Figura 8 do exemplo de de uma compilação de 25 agentes iniciators e 200 responders.

```
Restaurante restaurante1@192.168.1.113:1099/JADE vende {sushi , 56.209999999999994}
para um dos 200 clientes.
Timeout expired: faltam 200 clientes responderem o CNP
```

```
Prato : sushi está a venda. Valor original R$ 80.3 Novo valor R$ 56.209999999999994
Restaurante restaurante12@192.168.1.113:1099/JADE vende {sushi , 56.209999999999994}
para um dos 200 clientes.
Timeout expired: faltam 200 clientes responderem o CNP
Prato : mexicana está a venda. Valor original R$ 80.5 Novo valor R$ 56.35
Restaurante restaurante16@192.168.1.113:1099/JADE vende {mexicana , 56.35} para um
dos 200 clientes.
Restaurante restaurante11@192.168.1.113:1099/JADE vende {mexicana , 80.5} para um dos
200 clientes.
Restaurante restaurante4@192.168.1.113:1099/JADE vende {mexicana , 80.5} para um dos
200 clientes.
Restaurante restaurante24@192.168.1.113:1099/JADE vende {mexicana , 80.5} para um dos
200 clientes.
```

Figura 8: Exemplo de compilação do JADE com falha na resposta dos agentes

Considerações finais

Os objetivos deste trabalho foram a implementação de um Protocolo de Redes de Contrato (*Contract Net Protocol* - CNP) em duas linguagens de programação de Sistemas Multi-Agentes (SMA): JADE e Jason; avaliar a abordagem dos agentes para a solução do problema; avaliar as ferramentas na implementação da solução; e realizar uma comparação entre as implementações.

Podemos concluir que toda a estrutura e codificação do Jason se apresentou muito mais simples que a do JADE, bem como a capacidade do Jason para lidar com muitos agentes em paralelo é muito maior que a do JADE, trabalhando de forma muito eficiente os conceitos do CNP, pois os agentes interagiram entre si, de forma paralela, apresentando suas características principais como reatividade, pró-atividade e autonomia sem dificuldade. Já o JADE se mostrou um framework um pouco engessado para aplicações SMAs, para fazer a mesma aplicação que o Jason necessitamos muito mais linhas de código, e mesmo assim não conseguimos alcançar o mesmo desempenho.

Referências

BELLIFEMINE, F.; CAIRE, G; GREENWOOD, D. (2007). Developing Multi-Agent Systems with JADE. John Wiley & Sons Ltd., Chichester.

FIPA (2002). Foundation for Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification. Disponível em: http://www.fipa.org/specs/fipa00029/SC00029H.html

M. Wooldridge (2002). An Introduction to MultiAgent Systems. Sussex: JohnWiley and Sons.

M. Luck et al (2005). Agent Technology: Computing as Interaction. Southamp-ton: University of Southampton.

TEIXEIRA, Fábio. (2010). JADE: Java Agent Development Framework. Disponível em: https://www.dca.fee.unicamp.br/~gudwin/courses/IA009/artigos/IA009_2010_12.pdf