Комуникации в JADE. Структура на JADE съобщения. Изпращане и получаване на съобщение

Аделина Алексиева-Петрова

Предаване на съобщения 1/2

- MTS – Message Transport Services е услуга, която управлява всички съобщения, които се предават между отделните агенти и в самите агенти.

- По подразбиране JADE винаги стартира протокол за пренос на данни базиран на HTTP свързаност с инициализация на главния контейнер и без стартиране на MTP за подчинените контейнери.

Предаване на съобщения 2/2

- Вътрешното разпределение на съобщенията се осъществява по подходящ протокол за пренос наречен IMTP (Internal Message Transport Protocol).
- JADE платформата осигурява рутиране за входящите и също за изходящите съобщения използвайки рутираща таблица, която изисква директна свързаност по IP адрес с отсрещната страна.

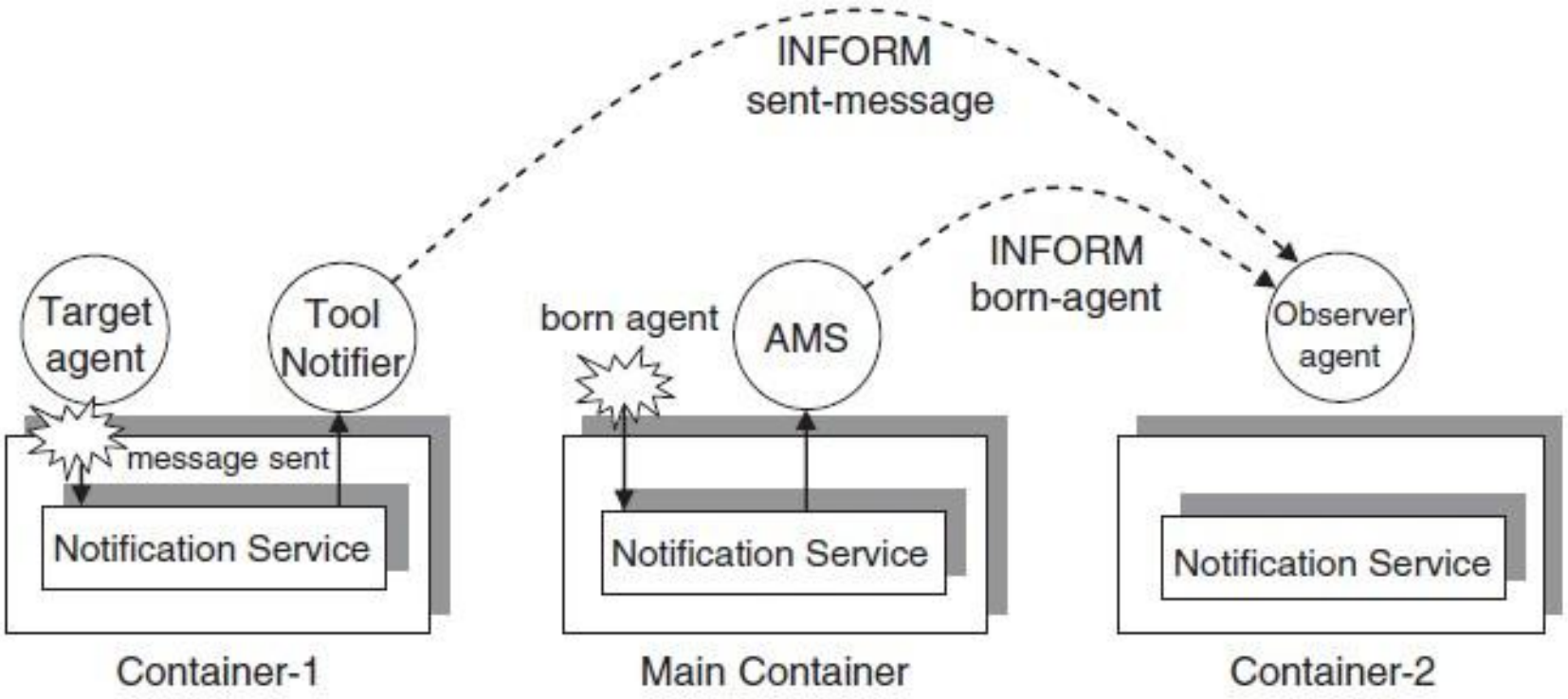
Вътрешно предаване на съобщения 1/2

- Вътрешното предаване на съобщения в JADE платформата (IMTP) се използва само за предаване на съобщения между агенти, които живеят в различни контейнери, но са в една и съща платформа.

Вътрешно предаване на съобщения 12/2

- От самото стартиране са на разположение две имплементации на главния IMTP протокол.
  - Единият е на базата на Java RMI и е протоколът, който се използва по подразбиране.
  - Вторият е на базата на TCP сокетите, които се използват при отсъствието на първия протокол.
- И двата протокола предоставят няколко опции за конфигуриране, позволявайки финна настройка на IMTP за специфични мрежи и устройства.

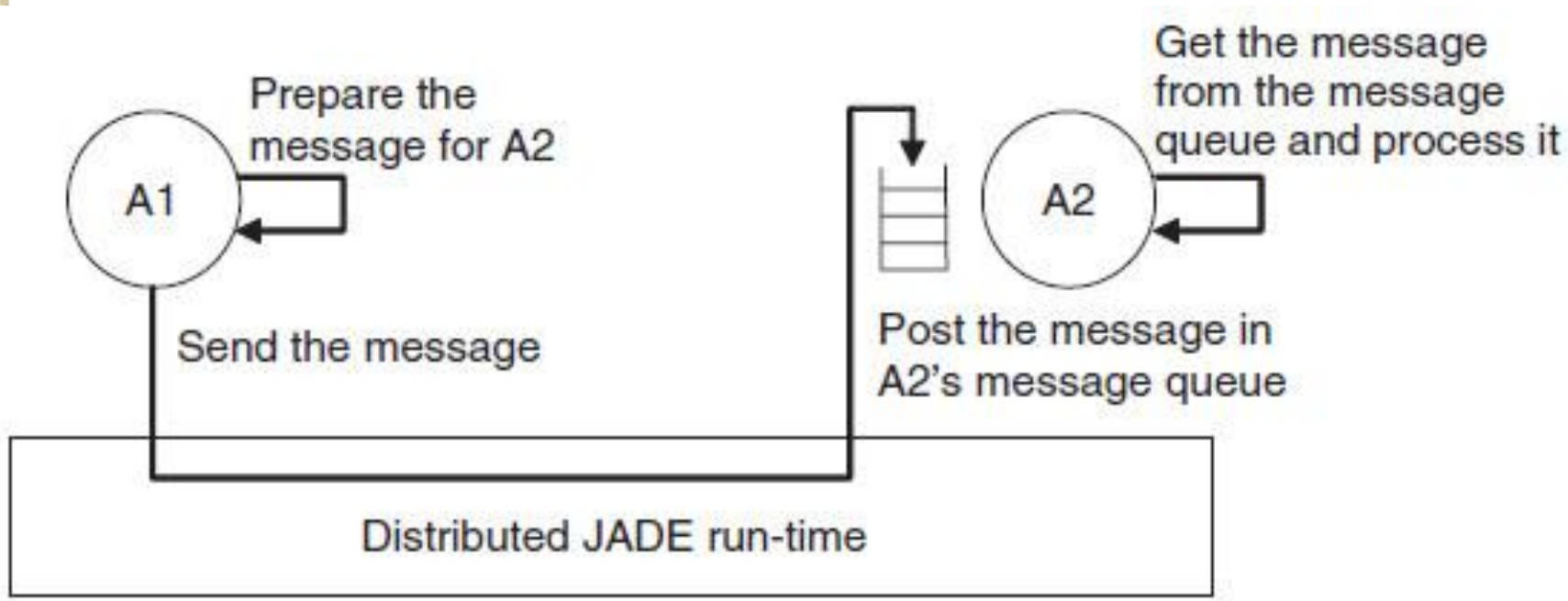
JADE система за съобщения



Комуникация 1/2

- Структурта на предаване на съобщения между агентите е базирана на асинхронната комуникация.
- Всеки агент има собствена пощенска кутия, в която постъпват получените за обработка съобщения.
- Всеки път когато ново съобщение влезне в пощенската кутия, агента е осведомяван за това.
- Съществува опцията агента да си избере дадено съобщение от пощенската си кутия по даден критерии, който е зададен от програмиста.

Комуникация 2/2



Формат на съобщението 1/3

- Подател на съобщението
- Списък с получателите на съобщението
- Цел на съобщението – показва какво иска да постигне подателя с изпращането на съобщението:
  - „Искане“ – подателя иска от получателя да извърши определено действие.
  - „Информиране“ – подателя иска да информира получателя на съобщението за дадено нещо.
  - „Предложение“ – подателя иска да започне преговори.

Формат на съобщението 2/3

- Съдържанието на съобщението пренася същинската информацията.
- Езика, който е използван за написването на съдържанието на съобщението. И получателя и подателя трябва да могат да декодират съобщението на базата на езика, на който е написано.
- Онтологията използвана при предаване на съобщението. И двете страни трябва да преписват на концепцията едно и също значение, за да е възможна комуникацията между тях.

Формат на съобщението 3/3

- Допълнителни полета, които позволяват да се контролират няколко едновременни разговора, ако има такива. Може да определя и крайното време за получаване на отговор. Такива полета са: **conversation-id, reply-with, in-reply-to and reply-by.**

Изпращане на съобщение

- jade.lang.acl.ACLMessage

```
ACLMessage msg = new
      ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter",
      AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Weather-forecast-ontology");
msg.setContent("Today it’s raining");
send(msg);
```

Получаване на съобщение

- При получаване на съобщение, то постъпва в кутията на получателя.
- За да вземе дадено съобщение от кутията получателя извиква метода receive().
- Този метод връща първото съобщение в пощенската кутия или null ако няма съобщения.

```
ACLMessage msg = receive();
if (msg != null) {
// Process the message
}
```

Работа с услуга “жълти страници’

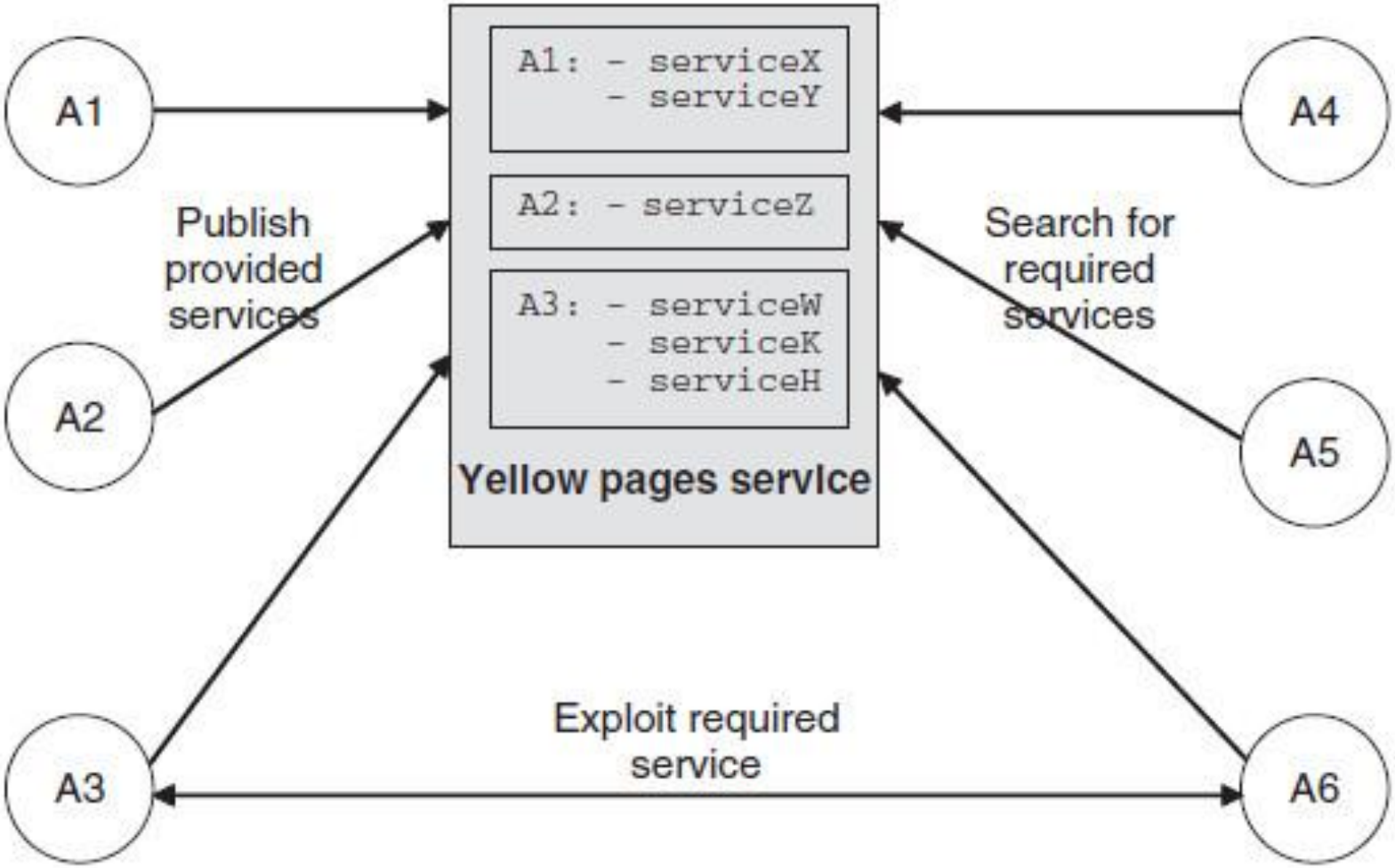
Аделина Алексиева-Петрова

Работа с услуга "Жълти страници"

- Регистърът на помощни средства ***(Directory Facilitator)*** осигурява услугата „жълти страници" за агентите в платформата.

- Услугата „жълти страници" позволява на агентите да публикуват описание на услугите, които те предлагат за да може другите агенти в мрежата лесно да ги откриват и използват.

Услугата „жълти страници”



Структура на описание на агент в регистъра DF

```
DFAgentDescription
      Name:          AID    //  Required for registration
      Protocols:     set of Strings
      Ontologies:    set of Strings
      Languages:     set of Strings
      Services:      set of
      {    Name:   String //Required
           Type:   String   //  Required
           Owner: String
           Protocols:    set of Strings
           Ontologies:   set of Strings
           Languages:    set of Strings
           Properties: set of
                { Name:    String
                  Value:   String
                }
      }
```

Услугата „жълти страници”

- Всеки агент може да регистрира услуга, която предлага, но може и да търси услуга. Като тези действия могат да се извършват по всяко време от жизнения цикъл на агента.
- Агента, който осигурява *Directory Facilitator* има възможността да взаимодейства с други агенти чрез обмяна на ACL съобщения, използвайки подходящ език (SLo) и онтология (тази, която е предоставена от FIPA).
- За да се опрости целия този процес, JADE предоставя специален клас jade.domain.DFService, който има възможността да публикува и да търси дадени услуги.

Публикуване на услуги

- Ако агент иска да публикува една или няколко услуги трябва да предостави описание на DF, което да съдържа:
  - неговия уникален номер (AID)
  - списък с предоставените услуги
  - списък на езиците и онтологията, които другите агенти трябва да използвата за да осъществят контакт с него.

Описание на услугата

- Описанието на всяка публикувана услуга трябва да съдържа:
  - типа на предлагана услуга
  - името на услугата
  - езика и онтологията, с които може да се използва услугата
  - набор от специални свойства представени чрез двойката ключ-стойност.

Пример за регистриране на услуга

```
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
…
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(myAgent.getAID());
ServiceDescription sd = new ServiceDescription();
sd.setName(”Seller”);
sd.setType(”Book-Trading");
dfd.addServices(sd);
try{
   DFService.register( myAgent, dfd );
}catch (FIPAException fe) {…}
```

Пример за търсене

```
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
…
DFAgentDescription template = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setName(”Seller”);
sd.setType(”Book-Trading");
template.addServices(sd);
try {
   DFAgentDescription[] result = DFService.search(myAgent,
   template);
   for(int i=0; I < result.length; i++){
        String name=result[i].getName().getLocalName(); ...
   }
}catch (FIPAException fe) {…}
```

Основни моменти при търсене на услуга

- Не се специфицира име на агента.
- За търсенето не е необходимо името на услугата, само нейният тип.
- Търсенето винаги връща един масив, който е с дължина 0, ако не са открити точните агенти.
- За да се получи някой наличен агент, се използва празно DFD, без специфициране на услугата.
- Търсенето може да върне повече от един резултат.

Дерегистриране на агент

- Добра практика е да се изтрие вписването на агент в регистъра, когато приключи:

```
protected void takeDown() {
// Deregister from the yellow pages
try {
DFService.deregister(this);
}
catch (FIPAException fe) {
fe.printStackTrace();
}
...
}
```

Мобилност на агента

```
import jade.core.ContainerID;
import jade.core.Location;
…
Location loc = here();
if (!loc.getName().equals("Main-Container")){
        doMove( new ContainerID("Main-Container", null) );
}
…
protected void beforeMove() {}
protected void afterMove() {
   System.out.println( here().getAddress() );
   System.out.println( here().getName() );
}
```

- The actual transition happens after the behavior that called *doMove()* ends.
- *beforeMove() is called before actual transition and can be* implemented to do some preparing actions.
- *afterMove() is called after the* transition is complete and can be implemented to perform some actions.

Клониране на агента

```
doClone( new ContainerID("Main-Container", null),
   "copyOfJohn");
…
protected void beforeClone() {
…
}
protected void afterClone() {
}
```

- The actual cloning happens after the behavior that called *doClone() ends*.
- *beforeClone() is called for mother agent before actual cloning and can be* implemented to do some preparing actions.
- *afterClone() is called for* daughter agent after the cloning is complete and can be implemented to perform some actions.

Ресурси

- **Developing Multi-Agent Systems with JADE**
  Fabio Bellifemine, Giovanni Caire, Dominic Greenwood
- **Design of Agent-Based Systems (TIES433)**
  Artem Katasonov, University of Jyväskylä