CONTRIBUCIÓ A LA MOBILITAT INTER-PLATAFORMA D'AGENTS MÒBILS: PROTOCOL DE MIGRACIÓ FRAGMENTADA

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Gerard Suades Méndez i dirigit per Jordi Cucurull Juan.

Bellaterra, setembre de 2008

El firmant, Jordi Cucurull Juan , professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Gerard Suades Méndez

Bellaterra, setembre de 2008

Firmat: Jordi Cucurull Juan

A tots aquells que amb petits detalls
han marcat les grans diferències

Agraïments

En primer lloc vull donar les gràcies al meu director de projecte, en Jordi Cucurull, per haver patit estoïcament la meva ignorància inicial vers al tema dels agents mòbils, per la seva predisposició en tot moment a resoldre els meus dubtes i pels consells, suggeriments i apreciacions que m'ha fet arribar, sempre amb bon criteri, durant l'elaboració d'aquest projecte.

També voldria donar les gràcies a la resta de professors i companys del grup SeNDA (Àlex, David, Oriol N., Xavis), al Juan, a l'Oriol M. (l'*Add-on* del SeNDA), als *MAGMA guys* (Víctor i Bernat) per l'intercanvi d'idees i les enriquidores tertúlies viscudes. Gràcies també al Nacho, per la seva predisposició a resoldre tots els problemes tècnics apareguts durant l'elaboració del projecte.

Gràcies a tots els companys que he tingut al llarg de la carrera i especialment a l'Abraham, el Carlos, el Ferran i sobretot al Xavi J. per tots els moments inoblidables que hem passat.

I gràcies al meu portàtil per haver aguantat i resistit tot aquest temps!

Finalment gràcies a la meva família pel suport que m'han donat durant tots aquests anys.

Índex

Índex de figures

Índex de taules

Glossari d'Acrònims

ACL Agent Communication Language

AID Agent unique IDentifier

AMM Agent Mobility Manager

AMS Agent Management Service

dEIC departament d'Enginyeria de la Informació i de les Comunicacions

DF Directory Facilitator

FIPA Foundation for Intelligent Physical Agents

FTP Fragmented Transfer Protocol

GNU GNU is Not Unix

HTTP HyperText Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

IPMA Inter-Platform Mobility Architecture

JADE Java Agent DEvelopment Framework

JAR Java ARchive

JIPMS JADE Inter-Platform Mobility Service

LGPL Lesser General Public License

MedIGS Medical Information Gathering System

MMP Main Migration Protocol

MTP Message Transport Protocol

MTS Message Transport Service

PCTP Push Cache Transfer Protocol

SeMoA Secure Mobile Agents

SeNDA Security of Networks and Distributed Applications

SL Semantic Language

TAC Tomografia Axial Computaritzada

TIC Tecnologies de la Informació i la Comunicació

TCP Transmission Control Protocol

UDP User Datagram Protocol

Capítol 1

Introducció

A finals del segle XX comença una nova etapa coneguda amb el nom d'era de la informació, en la qual el moviment d'informació esdevé més ràpid que el moviment físic, motivat per una millora i implantació globalitzada de les Tecnologies de la Informació i la Comunicació (TIC). Aquests canvis donen lloc a un nou sistema global d'informació on aquesta passa d'estar concentrada en petits nuclis inconnexos a estar distribuïda en una àmplia xarxa de comunicacions que facilita la compartició de la informació i que ha proporcionat les eines necessàries per començar a construir un nou model de societat coneguda com la societat del coneixement.

Tot aquest conjunt de canvis ha generat una gran quantitat de dades ubicades de manera dispersa en multitud de sistemes heterogenis. Aquest nou escenari, conegut com Mar-De-Dades, planteja un repte important: com localitzar i tractar aquesta informació. En els darrers anys han aparegut nous paradigmes que intenten tractar aquestes situacions, un d'aquests paradigmes són els agents mòbils. Els agents mòbils com que han de tractar dades que es troben distribuïdes en diferents nodes d'una xarxa es basen amb una idea que trenca amb tots els esquemes coneguts fins ara i és que el programa és l'encarregat d'anar a buscar les dades per

tractar-les en comptes d'importar les dades cap al programa, d'aquesta manera s'optimitza l'ús de recursos de la xarxa i s'evita congestionar-la. Un agent mòbil, doncs, el podem entendre com un navegant en un mar de dades que és capaç d'adaptar-se i sobreposar-se a les inclemències del temps i canviar el rumb per tal d'arribar a bon port.

En els últims anys s'han desenvolupat diverses plataformes (entorn on resideixen els agents) d'agents diferents totes elles incompatibles entre si. Amb l'objectiu de permetre interoperar diferents sistemes d'agents i les seves respectives plataformes es va crear la Foundation for Intelligent Physical Agents (FIPA) que defineix un conjunt d'estàndards per fomentar l'entesa entre sistemes heterogenis de plataformes d'agents. FIPA pertany al Institute of Electrical and Electronics Engineers (IEEE)

Una de les plataformes que està regida pels estàndards definits per FIPA és la plataforma Java Agent DEvelopment Framework (JADE) que està íntegrament desenvolupada amb llenguatge Java, característica que facilita la seva compatibilitat en un conjunt heterogeni d'arquitectures, fins i tot en dispositius mòbils. JADE no contempla, en un principi, el desplaçament d'agents entre plataformes sinó que només permet als agents viatjar dins d'una mateixa plataforma. Per aquest motiu el grup de recerca Security of Networks and Distributed Applications (SeNDA) del departament d'Enginyeria de la Informació i de les Comunicacions (dEIC) va iniciar un projecte per a la creació d'estàndards de mobilitat entre plataformes per a FIPA que es va materialitzar amb la proposta de l'arquitectura multiprotocol Inter-Platform Mobility Architecture (IPMA) [IPMA], a partir de la qual es va desenvolupar el servei de migració JADE Inter-Platform Mobility Service (JIPMS) [JIPMS]. JIPMS pretén oferir un servei de transferència flexible d'agents mòbils entre plataformes JADE gràcies a l'arquitectura multiprotocol IPMA que permet triar el protocol amb el què es vol realitzar la migració de l'agent.

Els protocols que hi ha actualment integrats en el JIPMS presenten certs problemes a l'hora de transferir agents mòbils de mides grans. En estudis anteriors com [JMT04] s'ha vist que el temps que suposa fer la migració d'agents mòbils amb un sol missatge creix exponencialment mentre la mida de l'agent que ha de migrar creix linealment. En [MIP07] s'ha fet una proposta de migració d'agents mòbils per a la plataforma JADE on la transferència de la part que correspon al codi d'un agent mòbil (el programa) s'envia dividida en diversos missatges més petits i s'ha comprovat que resulta més òptim que transmetre tot l'agent en un sol missatge. Aquesta darrera proposta, però, no ofereix una solució integral als problemes de migració trobats en [Med07] i [Med08], on un agent s'encarrega de recopilar dades obtingudes en el recorregut d'un itinerari per diverses plataformes. A mesura que aquest agent va recopilant informació, la seva mida augmenta i, en conseqüència, el temps que tarda en desplaçar-se d'una plataforma a una altra augmenta exponencialment. En certes ocasions, això pot provocar la cancel·lació de la migració com en [Med07], o bé que l'agent mòbil es vegi obligat a retornar a la plataforma origen precipitadament per descarregar les dades i seguir posteriorment l'itinerari des de l'última plataforma visitada com en [Med08].

Per tant, el present projecte pretén oferir un mecanisme òptim de migració per a agents mòbils que doni respostes a les necessitats anunciades anteriorment. Els resultats vistos en [MIP07] fan pensar que la solució passa per dissenyar un nou protocol de migració on tot l'agent es transfereixi dividit en missatges més petits.

1.1 Objectius

En aquest apartat s'enumeraran els objectius que s'han marcat per al desenvolupament del projecte:

1. Estudi dels protocols de migració implementats dins del mòdul JIPMS.

2. Anàlisi dels requisits que ha de tenir el protocol de migració que es desenvoluparà.

3. Disseny i implementació d'un protocol de migració fragmentada per a agents mòbils que respecti els estàndards definits per FIPA.

4. Desenvolupar un protocol de migració flexible. El protocol haurà d'oferir condicions de migració personalitzables que el permetin executar-se en diferents situacions.

5. Disseny i construcció d'un joc de proves que permeti avaluar el funcionament i rendiment del protocol implementat.

6. Integrar el protocol proposat en l'arquitectura multiprotocol (IPMA) que defineix el projecte JIPMS.

1.2 Estructura de la Memòria

A continuació es mostrarà com s'ha estructurat la memòria i s'explicarà amb una breu descripció el contingut de cada capítol.

- **Capítol 2: Estat de l'art**. En aquest capítol s'explicaran els conceptes previs necessaris per poder contextualitzar el projecte. L'ordre de presentació serà de més general a més concret. S'explicarà què són els agents mòbils i els aspectes relacionats amb l'entorn i funcionament d'aquests. Finalment es presentarà el marc on es desenvoluparà el projecte.

- **Capítol 3: Anàlisi**. En aquest capítol analitzarem els requisits que ha de tenir el protocol de migració que es vol desenvolupar, així com també un breu estudi de viabilitat i alguns aspectes que caldrà tenir en compte en les

fases de disseny i implementació. Finalment presentarem la planificació temporal prevista de les diferents etapes del projecte.

- **Capítol 4: Disseny**. En aquest capítol s'exposaran les decisions de disseny que s'han pres per complir amb els requisits establerts en el capítol d'anàlisi. Així mateix, s'explicarà detalladament el funcionament del protocol de migració i el disseny de les ontologies que permetran estructurar la informació que s'intercanvien les plataformes implicades en la migració.

- **Capítol 5: Implementació**. En aquest capítol s'explicarà com s'ha organitzat el codi a partir del disseny del protocol que s'ha fet, així com les decisions d'implementació que s'han pres per obtenir un millor rendiment del protocol. Finalment es presentaran els mètodes de suport al protocol que s'han implementat dins del servei de migració.

- **Capítol 6: Proves**. En aquest capítol s'explicarà l'entorn on s'han dut a terme els tests del protocol, així com el disseny que s'ha fet de les proves a realitzar. Finalment s'analitzaran els resultats obtinguts i se'n farà una interpretació exhaustiva en els diferents escenaris en què s'han realitzat els tests.

Per acabar la memòria s'ha fet un capítol de conclusions on es resumirà la feina feta i s'enumeraran els objectius assolits. Posteriorment s'explicaran les conclusions a les què s'ha arribat en base als resultats obtinguts i es presentaran una sèrie de línies de treball futures. Finalment es farà una valoració personal del projecte.

Capítol 2

Estat de l'art

En aquest capítol s'explicaran els conceptes previs necessaris per poder contextualitzar el projecte. L'ordre de presentació serà de més general a més concret. S'explicarà què són els agents mòbils i els aspectes relacionats amb l'entorn i funcionament d'aquests. Finalment es presentarà el marc on es desenvoluparà el projecte.

2.1 Agents Mòbils

Entenem per agent un component software autònom que és capaç d'interactuar amb el seu entorn, percebre els canvis i l'estat d'aquest i reaccionar als estímuls. Si aquest agent, a més a més, té la capacitat de desplaçar-se aleshores parlem d'agents mòbils. Per tant, un agent mòbil es caracteritza per ser:

- **Autònom**: És capaç de prendre decisions sobre les seves accions i el seu estat sense la intervenció directa d'una persona.

- **Sociable**: És capaç de cooperar amb persones o bé altres agents per tal d'assolir els seus objectius.

- **Reactiu**: És capaç de percebre el seu entorn i actuar en conseqüència als canvis que es produeixen.

- **Proactiu**: No només reacciona als canvis del seu entorn, sinó que a més a més pren la iniciativa i mostra certa intencionalitat en les seves accions.

- **Mòbil**: És capaç de suspendre la seva execució, moure's entre els diferents nodes que conformen una xarxa i posteriorment reprendre la seva execució.

Un cop definit què és un agent mòbil s'explicarà com és l'estructura d'aquest. Un agent mòbil està format bàsicament per tres parts: codi, dades i estat (veure Figura 2.1).

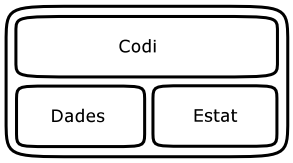

Figura 2.1: Estructura bàsica d'un agent mòbil

- **Codi**: El codi de l'agent és la part que conté el programa que ha d'executar l'agent per desenvolupar la tasca específica per la qual ha estat programat.

- **Dades**: Les dades de l'agent són les variables que aquest utilitza per emmagatzemar la informació que necessita.

- **Estat**: L'estat de l'agent és la part que emmagatzema tota la informació relativa a l'execució de l'agent com ara el comptador de programa, la pila d'execució, etc.

En la Figura 2.1 se'n mostra l'estructura bàsica, però aquesta pot patir algunes modificacions segons l'entorn on estigui l'agent. En la següent secció s'explicarà en quins casos l'estructura de l'agent es veu modificada i com afecta això a l'agent.

2.2 Tipus de Migracions

Com s'ha comentat anteriorment la propietat que caracteritza a un agent mòbil és la capacitat que té aquest per moure's entre els diferents nodes d'una xarxa. A l'acció de viatjar entre dos nodes d'una xarxa l'anomenem migració i aquesta pot ser bàsicament de dos tipus:

- **Migració Forta**: En aquesta els agents segueixen l'estructura bàsica definida en la Figura 2.1. Aquest tipus de migracions tenen l'avantatge de facilitar la implementació al programador, ja que aquest s'assegura que en qualsevol punt del codi pot posar una ordre de migració a l'agent i que aquest quan arribi al node de destí es seguirà executant des de la línia de codi següent. D'altra banda aquest tipus de migracions tenen el desavantatge que són dependents de l'arquitectura de la màquina on s'executa, fet que fa que siguin sistemes poc interoperables.

- **Migració Dèbil**: En aquesta els agents només contenen la part de codi i la part de dades. El fet que els agents no tinguin estat provoca que el codi s'executi des de l'inici un cop ha migrat, que aquest s'hagi de programar estructurant-lo com una màquina d'estats i que la part de les dades emmagatzemi variables de control específiques per saber en quin punt del codi es troba l'execució de l'agent. Aquest tipus de migració té l'avantatge de facilitar la implementació de la plataforma d'agents, ja que no ha d'accedir a recursos privilegiats de la màquina, i afavorir la interoperabilitat, ja que les implementacions poden ser independents de l'arquitectura de la màquina.

De cara al programador presenta el desavantatge que s'ha d'estructurar el codi d'una manera especial, cosa que no li permet fer una migració en qualsevol punt del codi i continuar a la línia següent com passa amb la migració forta.

2.3 Plataformes

Les plataformes o agències són els nodes de la xarxa on poden migrar els agents. Les plataformes, doncs, són l'entorn d'execució dels agents i els proporcionen un conjunt de serveis com per exemple serveis de comunicació i en el cas de les plataformes d'agents mòbils serveis de mobilitat. Existeixen diverses plataformes d'agents mòbils: AgentScape, Aglets, Grasshopper, JADE, Mansion, Secure Mobile Agents (SeMoA), Tracy, etc. En aquest projecte s'utilitzarà la plataforma JADE que està descrita a la secció 2.6.

2.4 FIPA

L'existència de múltiples plataformes d'agents, fa necessària la definició d'una sèrie de criteris que facin possible la comunicació entre els agents. En aquest projecte s'utilitzaran els estàndards definits per IEEE FIPA, que és una organització sense ànim de lucre, que es dedica a crear estàndards per a la interoperabilitat entre sistemes heterogenis d'agents i defineix aspectes com:

- **Elements de la Plataforma**: Perquè una plataforma sigui compatible amb els estàndards definits per FIPA ha de contenir els següents elements (veure Figura 2.2):

  Agent : Un element software capaç d'executar tasques de manera autònoma.

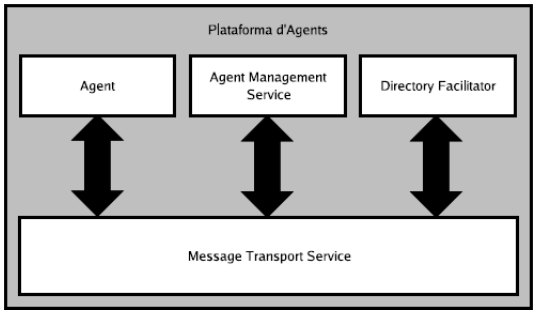

Figura 2.2: Elements d'una plataforma d'agents FIPA-compatible

Agent Management Service (AMS) : És un element que s'encarrega de registrar i mantenir el control de tots els agents que hi ha a la plataforma, com una guia de pàgines blanques.

Directory Facilitator (DF) : És un element que s'encarrega de registrar i mantenir el control de tots els serveis que ofereix la plataforma, com una guia de pàgines grogues.

Message Transport Service (MTS) : És l'element que s'encarrega d'oferir un sistema de comunicació entre agents, tant a nivell d'intra-plataforma com d'inter-plataforma.

- **Missatgeria**: El sistema de missatgeria definit per FIPA es divideix en tres nivells:

  Agent Communication Language (ACL) : Defineix l'estructura del cos dels missatges que s'intercanvien els agents, a més a més FIPA també defineix el llenguatge Semantic Language (SL) [FIPASL] de contingut dels missatges ACL.

  Message Transport Service (MTS) : Un servei que s'encarrega de dur a terme la transferència de missatges ACL entre els agents.

  Message Transport Protocol (MTP) : Si l'intercanvi de missatges es du

a terme entre plataformes diferents, el MTP s'encarrega d'encapsular els missatges ACL en un protocol de la capa d'aplicació.

- **Protocols d'Interacció**: Són models de comunicació que defineixen esquemes d'intercanvi de missatges ACL per facilitar la interacció entre agents [FIPAIP]. Típicament estan formats per una part iniciadora que s'encarrega d'engegar el procés comunicatiu i d'una o diverses parts receptores o participants.

2.5 Ontologies

El concepte d'ontologia en informàtica proposa la creació d'un esquema dins d'un domini establert que té com a finalitat facilitar la comunicació i la compartició de la informació entre diferents sistemes. Les ontologies a FIPA representen un model d'estructuració de les dades per als agents involucrats en una comunicació, aquestes ontologies pretenen oferir un model de representació de la informació comú per a tots els agents implicats en un mateix acte comunicatiu i permetre l'entesa d'aquests sense ambigüitats. Com s'ha comentat en l'apartat anterior FIPA utilitza el llenguatge SL per a respresentar la informació d'acord a una ontologia definida.

2.6 JADE

JADE [JADEWEB] és una plataforma d'agents mòbils de codi obert implementada completament amb el llenguatge Java que desenvolupa i distribueix l'empresa Telecom Italia sota llicència Lesser General Public License (LGPL). JADE compleix amb els estàndards definits per FIPA, per tant conté tots els aspectes definits en l'apartat 2.4. A més a més, JADE incorpora altres elements no definits per FIPA

com ara el concepte de contenidors que és utilitzat per a l'execució distribuïda de la plataforma. Una plataforma basada en JADE està formada per un o més contenidors, però només pot tenir un contenidor principal (*main containter*) que és el que conté el DF i l'AMS. La resta de contenidors han de consultar al contenidor principal els serveis de pàgines grogues DF i de pàgines blanques l'AMS.

2.6.1 *Behaviours*

Els *Behaviours* en JADE són classes Java que s'utilitzen per implementar les tasques que ha de dur a terme l'agent mòbil, és a dir, defineixen el comportament de l'agent. Un agent mòbil pot tenir diversos *Behaviours* que són gestionats pel planificador de JADE mitjançant una cua. JADE defineix diferents tipus de *Behaviours* segons les característiques de les tasques que hagi de realitzar l'agent, com per exemple si són tasques que s'han d'executar iterativament o bé una sola vegada, etc.

2.6.2 Serveis de JADE

JADE a part dels serveis descrits en 2.4 permet incorporar nous serveis a través d'*Add-ons* que s'instal·len en forma de mòduls i es poden activar quan s'inicia la plataforma. JADE incorpora, en la seva infraestructura interna de serveis, un servei de mobilitat intra-plataforma que permet migrar agents entre contenidors d'una mateixa plataforma, però no disposa de cap servei de mobilitat inter-plataforma que possibiliti la migració d'agents entre diferents plataformes JADE. Aquest segon tipus de mobilitat el proporciona l'*Add-on* JIPMS [JIPMS].

2.7 JIPMS

El grup SeNDA va iniciar un projecte per a la creació d'estàndards de mobilitat per a FIPA que va finalitzar amb la proposta de l'IPMA. A partir de l'IPMA es va desenvolupar l'*Add-on* JIPMS que és un servei que es pot incorporar a la plataforma JADE i que està en continu desenvolupament.

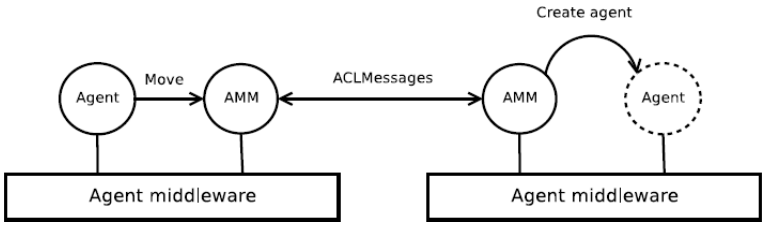

Figura 2.3: Esquema bàsic de migració del JIPMS

En la Figura 2.3 es mostra un esquema senzill on es poden veure els elements principals que prenen part en una migració d'un agent mòbil entre dues plataformes. Quan un agent vol traslladar-se a una altra plataforma fa una petició de migració a l'Agent Mobility Manager (AMM) que és l'agent que s'encarrega de gestionar les migracions entre diferents plataformes. L'AMM de la plataforma origen contacta amb l'AMM de la plataforma destí i en negocia la transferència de l'agent mòbil. L'agent és transferit mitjançant missatges ACL a la plataforma destí on és reconstruït.

En la Figura 2.4 es mostra l'estructura de l'IPMA on s'il·lustra el procés de negociació que s'estableix entre els AMM de les dues plataformes. Aquest procés està format pel Main Migration Protocol (MMP) que s'encarrega d'iniciar el procés de migració, pactar el protocol de migració, transferir (*Architecture Steps*) i registrar l'agent a la plataforma de destí, eliminar l'agent de la plataforma origen un cop acabada la transferència de l'agent i finalment de reprendre l'execució de l'agent en la plataforma de destí.

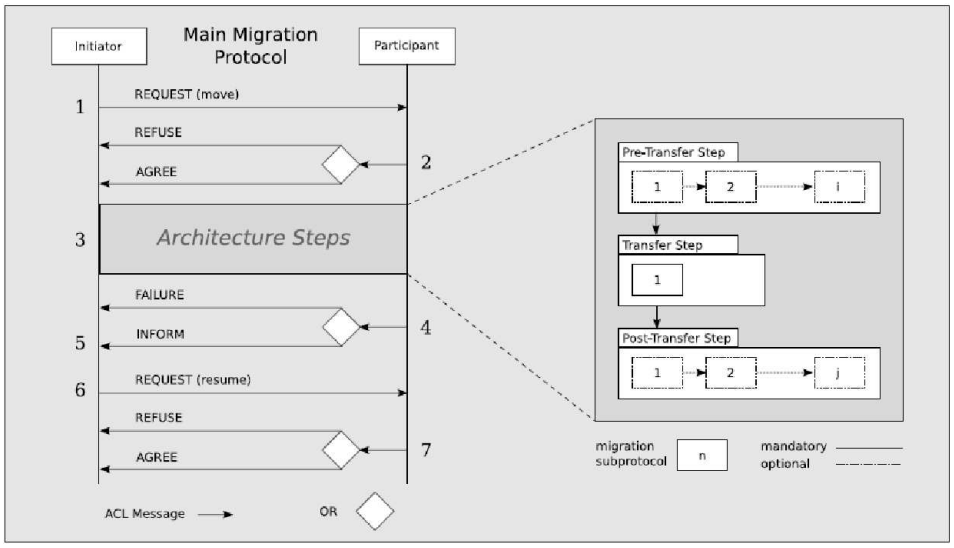

Figura 2.4: estructura de l'IPMA

El pas 3 (*Architecture Steps*) del MMP és el que caracteritza el procés de migració i està dividit en tres fases:

- **Pre-Transfer**: Conté els protocols que s'han d'executar abans de començar a transferir l'agent. S'utilitzen per negociar paràmetres addicionals, aspectes de control d'accés a recursos, autorització, etc. És una fase opcional.

- **Transfer**: Conté el protocol de transferència de l'agent. N'hi poden haver diversos, però només se'n pot triar un per a realitzar cada migració. És una fase obligatòria i és la que caracteritza la migració.

- **Post-Transfer**: Conté els protocols que s'han d'executar després d'haver acabat la transferència completa de l'agent. S'utilitzen per corroborar la integritat de l'agent, obtenir resultats addicionals de la transferència de l'agent, etc.

Aquests passos són els que defineixen l'IPMA com una arquitectura multiprotocol i és on es troba emmarcat el present projecte. En aquest projecte es pretén

dissenyar un protocol de transferència d'agents mòbils inclòs en la fase de *Transfer* de l'IPMA i desenvolupar-lo dins de l'*Add-on* JIPMS. Es pretén, doncs, fer una aportació a aquesta arquitectura definida i permetre una nova manera de transferir els agents mòbils. Aquest protocol serà una alternativa als protocols de migració ja existents a l'IPMA.

2.8 Resum

En aquest capítol s'han explicat el conjunt de conceptes previs necessaris per entendre el projecte. Així mateix s'han presentat tots els elements que conformen l'entorn on s'emmarca el projecte començant pels aspectes més generals com la definició dels agents mòbils, tipus de migracions d'agents mòbils i el concepte de plataformes. Finalment acabant amb conceptes més concrets com ara els estàndards i la plataforma que s'han utilitzat en aquest projecte, així com el marc on està inclòs el projecte desenvolupat (IPMA)

Capítol 3

Anàlisi

En aquest capítol analitzarem els requisits que ha de tenir el protocol de migració que es vol desenvolupar, així com també un breu estudi de viabilitat i alguns aspectes que caldrà tenir en compte en les fases de disseny i implementació. Finalment presentarem la planificació temporal prevista de les diferents etapes del projecte.

3.1 Anàlisi de Requisits

En aquest projecte es pretén dissenyar i desenvolupar un nou protocol de migració per agents mòbils. Aquest protocol estarà emmarcat dins de l'*Add-on* de migració JIPMS de la plataforma JADE. A continuació s'enumeraran els requisits que ha de tenir el protocol.

3.1.1 Requisits Funcionals

- El protocol ha de permetre l'enviament d'un agent mòbil dividit en diverses parts o fragments.

- La fragmentació serà tant a nivell del codi com de les dades de l'agent.
- Només s'enviarà el codi de l'agent si és estrictament necessari.
- La mida dels fragments escollida serà fixa per als fragments corresponents a una mateixa migració.
- La mida del fragment ha de ser parametritzable. Aquest paràmetre pot venir definit tant per la plataforma com pel propi agent mòbil.
- Ha de ser un protocol flexible, cada migració ha de poder personalitzar la mida de fragment amb el que desitja transferir l'agent mòbil.
- Ha de ser un protocol que permeti generalització, no només ha de permetre l'enviament fragmentat de l'agent, sinó que també ha de permetre la migració de l'agent sense fragmentar.

3.1.2 Requisits No Funcionals

- El protocol haurà de respectar l'arquitectura de protocols definida per l'*Add-on* de migració JIPMS.
- El protocol haurà d'implementar un sistema de migració per a plataformes JADE.
- El protocol de migració s'haurà de desenvolupar en el llenguatge de programació Java.
- La migració de l'agent es basarà en els estàndards de comunicació definits per FIPA que utilitzen la missatgeria ACL.
- Es disminuirà el temps de migració entre plataformes que presenta el protocol de migració per defecte de l'*Add-on* de migració JIPMS.

3.2 Estudi de Viabilitat

3.2.1 Viabilitat Tècnica

Aquest projecte es desenvoluparà amb el llenguatge de programació Java, el coneixement del qual s'ha adquirit al llarg de la carrera. Serà necessari un treball previ de documentació i formació en relació a les eines i l'entorn de desenvolupament, explicats en les seccions 2.6 i 2.7, que està previst en la planificació temporal del projecte. El projecte no requereix cap maquinari específic, simplement s'utilitzarà un pc estàndard amb el sistema operatiu GNU/Linux. Val a dir, que el grup de recerca SeNDA posa a disposició dels seus projectistes de fi de carrera un laboratori d'ordinadors amb tot el programari i eines necessàries per al desenvolupament dels projectes relacionats amb agents. A més a més, el departament disposa d'un *cluster* dedicat per a poder fer les proves de rendiment de migració d'agents.

3.2.2 Viabilitat Econòmica

Tot el programari utilitzat per a la realització del projecte és programari lliure, per tant no suposarà cap cost afegit perquè no s'haurà de contemplar la compra de cap llicència. Com ja s'ha comentat a l'apartat de viabilitat tècnica, el sistema operatiu utilitzat és un Fedora Core 5, l'entorn de desenvolupament és JADE i l'entorn integrat de desenvolupament és l'Eclipse. Per calcular el cost que suposaria desenvolupar el projecte, caldria aplicar el preu/hora de desenvolupament estipulat per cada empresa al nombre d'hores previstes per a la seva realització (descrit en l'apartat de Planificació Temporal del Projecte).

3.2.3 Viabilitat Operativa

Aquest projecte neix per una necessitat derivada d'uns resultats extrets en altres projectes. En [JMT04] s'ha vist que el temps que suposa fer la migració d'agents mòbils amb un sol missatge creix exponencialment quan la mida de l'agent que ha de migrar creix linealment. En [MIP07] s'ha fet una proposta de migració d'agents mòbils on la transferència de la part que correspon al codi d'un agent mòbil s'envia dividida en diversos missatges més petits i s'ha comprovat que resulta més òptim que transmetre tot l'agent en un sol missatge. Per tant, la viabilitat operativa del projecte està garantida.

3.2.4 Viabilitat Legal

Tant la plataforma JADE com el propi projecte JIPMS estan sota llicència LGPL, així que no hi haurà cap restricció legal en quan al seu ús. Cal tenir en compte, que aquest projecte implementarà un protocol dins de l'arquitectura de protocols definida pel projecte JIPMS, per tant, el projecte desenvolupat també restarà sota llicència LGPL.

3.3 Fragmentació de l'Agent

Com s'ha comentat en l'apartat 2.1 un agent mòbil es composa de tres parts: estat, dades i codi. En aquest apartat s'explicarà quin enfoc es donarà en la migració de cadascuna d'aquestes parts. En les fases de disseny i implementació es veurà amb més detall com s'ha dut a terme.

L'estat de l'agent és tota aquella informació relativa a l'execució de l'agent, representat típicament pel comptador de programa, la pila d'execució, registres del processador, etc. La mida d'aquest tipus de dades sol ser similar per a tots els agents i es creu que la seva mida no serà excessivament gran, per aquesta raó s'ha

cregut oportú no sotmetre l'estat de l'agent a un procés de fragmentació per tal de realitzar la migració.

Les dades o atributs d'un agent són la part que caracteritzen l'agent, en una migració aquesta part és capturada i enviada de la plataforma origen a la plataforma destí, on l'agent és reconstruït.

Típicament la part corresponent a les dades d'un agent sol ser de mida reduïda (2-3 Kbytes), ja que només conté variables pròpies de l'agent, variables de control i, eventualment, alguna estructura de dades simple per a què l'agent pugui emmagatzemar informació concreta relacionada amb l'aplicació per la qual ha estat programat específicament. Aquest tipus de condicions són les que havien predominat fins ara, però darrerament han aparegut nous escenaris on aplicar el paradigma dels agents mòbils. Les aplicacions d'aquest tipus d'entorns utilitzen volums de dades cada vegada més grans i més complexes. Com a conseqüència directa, les dades associades als agents poden arribar a grandàries considerables de fins a centenars de Kbytes. L'actual protocol de migració, on totes les dades s'encapsulen en un únic missatge, imposa una gran penalització de temps quan els agents d'aquestes aplicacions han de dur a terme migracions entre plataformes. Com hem comentat en el capítol anterior, el temps de migració creix de manera exponencial a mesura que l'agent augmenta la seva grandària linealment.

Un clar exemple d'aquestes aplicacions són les aplicacions mèdiques com Medical Information Gathering System (MedIGS) [MedIGS], on un agent s'encarrega de recopilar tota la informació mèdica requerida d'un pacient que es troba distribuïda en diversos hospitals. Aquestes dades mèdiques poden contenir l'historial sencer del pacient o bé informes de proves mèdiques específiques realitzades al pacient. Tot aquests tipus de dades, però, són atributs de l'agent i necessiten, doncs, un model de migració que permeti moure dades de l'agent que ocupin uns quants centenars de Kbytes. Fins ara, el temps de migració en aquests casos era

molt elevat, arribant en ocasions a cancel·lar la migració de l'agent com a conseqüència d'haver-se superat el temps màxim previst.

Per aquesta raó, s'ha cregut convenient exigir com un dels requisits primordials la fragmentació de les dades, ja que en anteriors estudis [JMT04] [MIP07] s'ha vist que l'enviament fragmentat del codi pot resultar més ràpid i eficient que l'enviament en un sol missatge ACL. S'estima, doncs, que l'enviament i recepció de la part de les dades fragmentada també contribueixi a millorar significativament els temps de migració de l'agent.

L'altra part que s'ha d'enviar d'un agent és el codi. A [MIP07] es va demostrar que l'enviament fragmentat del codi resulta molt més eficient que no pas enviar-lo en un sol missatge ACL per a mides de codi grans. L'anàlisi d'aquest protocol de migració ens pot ser d'ajuda a l'hora de realitzar el disseny del nou protocol.

En [JMT04] s'ha vist que el tractament de missatges ACL grans, en el cas de JADE, és molt ineficient. Aquest aspecte s'optimitzarà amb la reducció de la mida dels missatges ACL a enviar. Una conseqüència directa d'aquesta iniciativa és l'augment del nombre de missatges ACL que s'hauran d'enviar. Convé destacar que caldrà trobar una solució òptima per tal que l'augment de missatges no esdevingui excessiu i acabi resultant més costós que migrar l'agent en un sol missatge ACL. Per tal de trobar aquesta solució equilibrada, caldrà reduir en la mesura del possible l'intercanvi de missatges ACL durant el procés de migració, sense augmentar-ne massa la mida.

Per tant, serà decisiva la tria d'una mida de fragment òptima. La mida dels fragments determina tant el nombre de fragments, és a dir, de missatges ACL que s'enviaran com la grandària d'aquests missatges ACL. Quan més grans siguin els fragments, menys missatges s'hauran d'enviar, però més temps es trigarà a gestionar aquests missatges. En canvi, quan més petits siguin els fragments més missatges ACL es generaran, però aquests es podran tractar més ràpidament. Amb

tot, serà necessari definir un conjunt de proves adequat que ens permeti trobar una solució de compromís que minimitzi el temps de migració dels agents mòbils.

3.4 Planificació Temporal del Projecte

El projecte l'hem dividit en sis etapes. En l'etapa de documentació s'hi preveu l'estudi de l'entorn i les eines de desenvolupament que haurem d'utilitzar per a desenvolupar el projecte, així com l'estudi del servei de migració JIPMS. Dins d'aquesta etapa de documentació també s'hi preveu la realització de l'informe previ del projecte que té com a data límit d'entrega el 14 de Gener de 2008. Un cop acabada la fase de formació específica per al projecte, ja es tindran els coneixements necessaris per a poder realitzar un anàlisi de requisits del nostre projecte que ens permetrà fer un disseny adequat del protocol. Posteriorment s'iniciarà la fase d'implementació i proves, i preveiem un mes per a l'escriptura de la memòria. Val a dir que malgrat les etapes s'han disposat d'una manera seqüencial en la planificació, és de preveure que les fases de disseny i implementació i implementació i proves siguin etapes que segueixin un model evolutiu, fent ús del concepte de *backtracking* per poder resoldre els problemes que puguin anar sorgint al llarg del projecte.

Per a la realització del projecte s'estima una dedicació a temps parcial (mitja jornada) durant un semestre més una part prèvia de documentació i familiarització amb l'entorn i eines de desenvolupament. Amb tot es preveu una dedicació de:

20h Documentació + 40h Anàlisi + 40h Disseny + 120h Implementació + 40h Test + 80h Memòria = 340 hores.

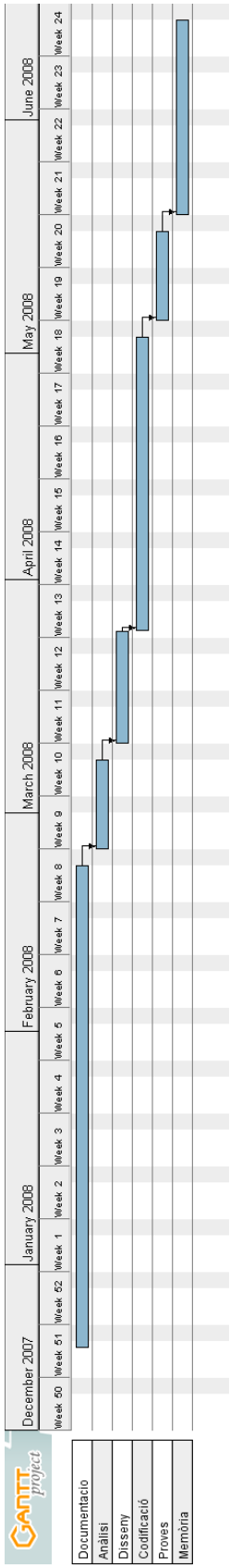

Figura 3.1: Diagrama de Gantt

3.5 Resum

En aquest capítol s'ha realitzat un anàlisi dels requisits que ha de tenir el protocol de migració per a agents mòbils que es vol desenvolupar. A més a més s'ha fet un estudi de viabilitat del projecte on s'han analitzat i comentat aspectes tècnics, operatius, econòmics i legals del projecte a realitzar, així com també s'han introduit algunes idees que permetran començar a enfocar les fases de disseny i implementació del protocol. Finalment s'ha fet una planificació temporal del projecte on s'ha detallat una estimació del temps que es destinarà a realitzar les diferents fases del projecte.

Capítol 4

Disseny

En aquest capítol s'exposaran les decisions de disseny que s'han pres per complir amb els requisits establerts en el capítol d'anàlisi. Així mateix, s'explicarà detalladament el funcionament del protocol de migració i el disseny de les ontologies que permetran estructurar la informació que s'intercanvien les plataformes implicades en la migració.

4.1 Disseny del Protocol

Un cop s'han concretat els requisits que ha de tenir protocol de migració fragmentada d'agents mòbils i la definició de certs aspectes que haurà de complir el projecte, en la fase de disseny es veurà quines decisions s'han pres per garantirlos. A continuació es comentaran aspectes que s'han tingut en compte a l'hora de realitzar el disseny del protocol.

- El protocol ha de suportar l'ús d'un sistema de *cache* de codi, basat en la idea que s'ha anunciat en l'apartat 3.1.1. La plataforma origen preguntarà a la plataforma destí si cal migrar la part del codi de l'agent o no, de manera que només s'enviarà el codi de l'agent si és estrictament necessari. Això

és possible gràcies a què codi i dades de l'agent són parts que es poden tractar de manera totalment independent. D'aquesta manera podem evitar enviar informació redundant, aconseguint reduir així el nombre de missatges ACL a enviar, ja que només s'envia aquella informació que és realment necessària.

- Evitar confirmar la recepció dels missatges rebuts sempre que sigui possible. La plataforma de destí no confirmarà la recepció de cada fragment de l'agent rebut com es fa en [MIP07]. S'estima que rarament es donarà el cas que un dels missatges ACL enviats es perdi i no acabi arribant al seu destí, ja que habitualment el protocol que hi ha per sota de la migració és HyperText Transfer Protocol (HTTP) i aquest protocol per si mateix ja disposa de sistemes de control d'errors. D'aquesta manera disminuïm també l'enviament de missatges ACL.

4.1.1 Funcionament del Protocol

En la Figura 4.1 es mostra l'esquema de funcionament en AUML [AgentUML] [AIPUML] del protocol de migració fragmentada que s'ha dissenyat. L'esquema mostra l'intercanvi de missatges ACL entre les plataformes d'origen i destí que prenen part en una migració d'un agent mòbil. El disseny s'ha fet seguint l'arquitectura definida per FIPA.

El protocol s'ha dividit en sis passos, quatre dels quals es realitzaran sempre en totes les migracions i els dos restants només s'executaran en cas que es perdi algun dels fragments enviats. Tot seguit s'explicarà amb detall els sis passos dels que consta el protocol.

- Pas 1: La plataforma origen sol·licita fer la migració d'un agent a la plataforma destí (REQUEST). En aquest missatge es llistaran les condicions

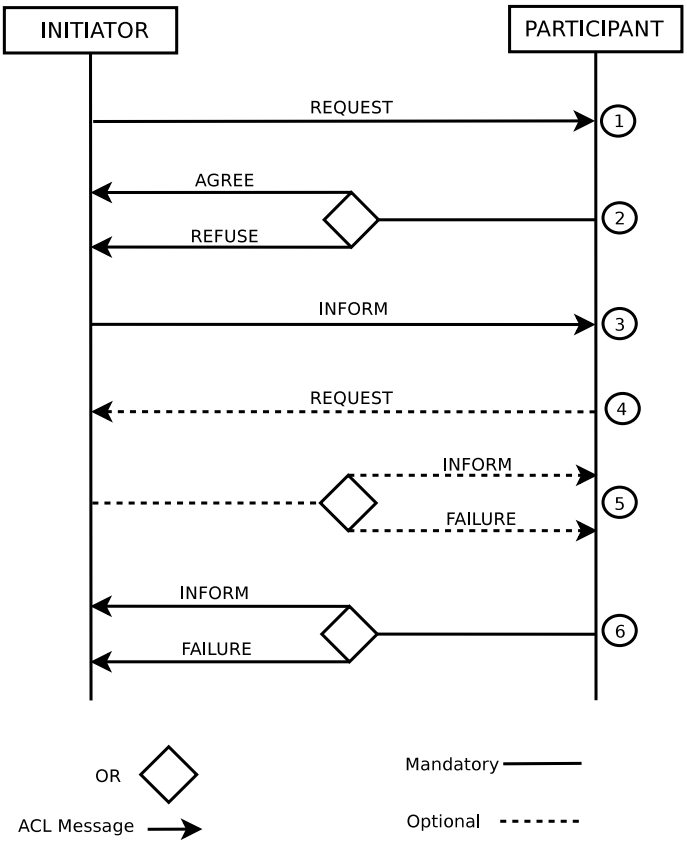

Figura 4.1: Esquema del Protocol de Migració Fragmentada

amb les que es vol dur a terme la migració. Es proporcionarà informació relacionada amb la mida de l'agent que es vol enviar i la mida dels fragments amb què es desitja fer aquesta migració. A més a més, es facilitarà un identificador únic del codi de l'agent (CID) per tal que la plataforma destí pugui avaluar si cal que s'enviï el codi o no.

- Pas 2: La plataforma destí avalua les condicions de migració i decideix si les accepta o no. Si no està d'acord amb les condicions envia un missatge refusant la sol·licitud de migració (REFUSE) i aquesta es cancel·la, passant directament al pas 6. En cas contrari, comprova a través del CID si el codi ja es troba a la plataforma. En cas afirmatiu no serà necessari transferir el codi de l'agent i només caldrà migrar les dades. Altrament la plataforma origen haurà d'enviar les dades i el codi de l'agent. Aquesta informació anirà continguda dins del missatge d'acceptació de la sol·licitud de migració (AGREE).

- Pas 3: Creació i enviament seqüencial dels fragments de la part de les dades de l'agent i, si és requerit, creació i enviament seqüencial dels fragments del codi de l'agent. Cada fragment implicarà l'enviament d'un missatge de la plataforma origen a la plataforma destí. La plataforma destí quan hagi rebut tots els fragments de l'agent el reconstruirà i es passarà al pas 6. Si per alguna raó, la plataforma destí no rep algun dels fragments, passat un cert temps, saltarà al pas 4 on sol·licitarà l'enviament dels fragments que no s'hagin rebut. Per determinar aquest temps d'espera, caldrà definir un *timeout* adient.

- Pas 4: Esgotat el temps d'espera previst per a la recepció de tots els fragments s'activarà el sistema de recuperació de fragments. Per cada fragment no rebut s'enviarà una sol·licitud de reenviament d'aquest fragment a la

plataforma origen. Aquest sistema de confirmació implícit de fragments evitarà carregar el sistema quan la migració de l'agent es desenvolupi amb normalitat, ja que els passos 4 i 5 no hi intervindran.

- Pas 5: Per cada sol·licitud, la plataforma origen s'encarregarà de tornar a crear i enviar el fragment requerit encapsulat en un missatge INFORM. A la plataforma destí es rebran els fragments prèviament sol·licitats i es reconstruirà l'agent. En cas de no poder recuperar la informació corresponent al fragment sol·licitat, s'enviarà un missatge d'error a la plataforma destí, que s'encarregarà de cancel·lar la migració.

- Pas 6: Si tot ha anat bé i l'agent s'ha pogut reconstruir en el destí, s'envia un missatge a la plataforma origen informant que la migració ha anat bé. Si hi ha hagut qualsevol problema en el transcurs de la migració s'envia un missatge amb la causa de l'error a la plataforma d'origen.

4.1.2 Protocols d'Interacció

Com s'ha explicat en l'apartat 2.4, FIPA té una sèrie de protocols d'interacció estàndards que defineixen esquemes d'intercanvi de missatges ACL [FIPAIP]. Un cop vist el funcionament del protocol, s'explicarà com s'ha organitzat l'intercanvi de missatges, és a dir, quins protocols d'interacció s'han utilitzat.

De tots els protocols d'interacció definits per FIPA el que s'adiu més bé al disseny proposat és el *FIPA Request Interaction Protocol* [FIPARIP] l'esquema del qual es pot observar a la figura 4.2.

El protocol es divideix en dues parts ben diferenciades: Una part principal que s'encarrega de dur a terme la funcionalitat bàsica del protocol amb els passos 1, 2, 3 i 6 i una altra part secundària que s'encarrega del tractament de fragments perduts durant la migració amb els passos 4 i 5.

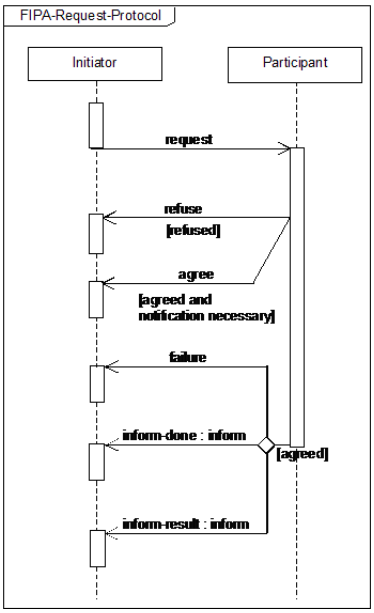

Figura 4.2: *FIPA Request Interaction Protocol*

Malgrat el *FIPA Request Interaction Protocol* no s'ajusta a la totalitat d'intercanvi de missatges de la part principal del disseny del protocol, sí que s'adapta perfectament als passos 1, 2 i 6. A més a més la part secundària sí que s'adapta perfectament al *FIPA Request Interaction Protocol*, ja que els missatges AGREE/REFUSE poden ser missatges opcionals en circumstàncies on l'acció que es vol dur a terme s'hagi de fer en un curt espai de temps. En aquests casos s'obvien els missatges d'acceptació de la sol·licitud i es retorna directament el resultat de l'execució de l'acció. Per tant es pot assignar un *FIPA Request Interaction Protocol* a cada una de les dues parts de les què consta el protocol dissenyat.

D'altra banda també hi ha la possibilitat de crear un protocol d'interacció fet a mida per al disseny del nostre protocol que respongui a l'esquema de comunicació proposat i defineixi exactament l'intercanvi de missatges que hem dissenyat.

Per tant una decisió important de disseny que cal prendre és si per a realitzar la migració s'utilitzen dos *FIPA Request Interaction Protocol* o bé és dissenya un

protocol d'interacció fet a mida. Després de valorar-ne els pros i els contres, s'ha decidit utilitzar el *FIPA Request Interaction Protocol* per les següents raons:

1. L'esquema proposat s'adiu amb el *FIPA Request Interaction Protocol* proposat per FIPA.

2. Aquesta opció permet fer un desenvolupament modular del protocol. Una primera etapa en què s'implementaria el protocol de migració sense tractament d'errors, i una segona etapa on s'hi afegiria el tractament de fragments perduts:

   - La funcionalitat bàsica del protocol: Passos 1, 2, 3 i 6. Per al pas 3 caldrà fer algunes modificacions que s'explicaran amb més detall en el capítol d'implementació.
   - Tolerància a errors: Passos 4 i 5. Implementaran la recuperació dels fragments de l'agent perduts.

3. S'aprofita un model d'interacció estàndard ja creat, fet que pot ajudar a realitzar l'etapa de codificació més ràpidament.

4.1.3 Timeouts

Com s'ha comentat al pas 3 de l'apartat 4.1.1, és necessari definir un temps d'espera màxim per a la recepció de fragments. S'han definit dos *timeouts*.

Un *timeout* de migració que establirà el temps màxim amb què s'ha de dur a terme la transferència sencera d'un agent mòbil, transcorregut aquest temps, si l'agent no ha migrat completament es cancel·larà la migració. Aquest *timeout* de migració es podrà definir com a paràmetre d'entrada en arrancar una plataforma.

L'altre *timeout* que cal definir és el de la recepció de fragments, que definirà el temps d'espera màxim que suportarà la plataforma destí per a la recepció de tots

els fragments de l'agent mòbil, és a dir, marcarà el punt d'inflexió en el qual la plataforma destí passa d'estar esperant fragments a demanar explícitament aquells que no han arribat. La pregunta que sorgeix en aquest punt és: Esperarem el mateix temps per a la recepció d'un agent de 1MB que per a la recepció d'un agent de 10 KB? La resposta és no, per tant s'ha cregut oportú que el temps d'espera de recepció de fragments no sigui un valor fix, sinó que sigui un valor que depengui de la mida de l'agent a transferir, tenint en compte, però, que aquest *timeout* adaptatiu mai pot ser superior al *timeout* general del protocol. El *timeout* adaptatiu no es podrà definir com a paràmetre, sinó que es determinarà automàticament pel propi protocol de migració fragmentada a partir de les condicions específiques de cada migració, com per exemple la mida de l'agent, la mida de fragment, etc.

4.2 Sistema de Recuperació de Fragments

Com s'ha comentat en l'apartat 3.3 el protocol de transport dels missatges ACL acostuma a ser l'HTTP, que alhora va damunt del protocol Transmission Control Protocol (TCP) de la capa de transport, el qual ja disposa d'un sistema d'entrega fiable de la informació. Per tant, en principi els missatges que s'envien arriben al seu destí i rarament se'n perdrà cap. Per aquesta raó s'ha decidit que no era necessari confirmar la recepció dels missatges i d'aquesta manera agilitzar el procés de migració.

Pel cas en què el protocol de la capa de transport no sigui fiable, és a dir, que no implementa cap sistema que garanteixi l'entrega de les dades enviades com per exemple User Datagram Protocol (UDP), s'ha decidit dissenyar un sistema de recuperació de fragments. El sistema tindrà les següents característiques:

1. La seva implementació no penalitzarà el temps de migració, ja que si es reben tots els fragments correctament els passos 4 i 5 no s'arribaran a executar

mai.

2. La pèrdua d'un fragment no provocarà la cancel·lació de tota la migració, i per tant no es perdrà tota la feina feta. La idea és dotar d'una segona oportunitat al protocol en cas d'aparèixer un problema puntual en la migració de l'agent mòbil.

3. Oferirà una certa fiabilitat en situacions on el protocol de la capa de transport no ofereixi les garanties suficients per a què la informació arribi al seu destí.

El seu funcionament consistirà en què la plataforma remota, un cop esgotat el *timeout* de recepció de fragments, enviarà un missatge per a cada fragment de l'agent que no s'ha rebut. Aquest missatge contindrà tota la informació necessària per a què la plataforma d'origen pugui tornar a crear i enviar la porció de l'agent que no ha aconseguit arribar al destí. En l'apartat d'ontologies s'explicarà amb detall quina és aquesta informació necessària. La plataforma origen quan rebi aquesta sol·licitud tornarà a crear el fragment demanat i l'enviarà a la plataforma remota.

4.3 Ontologia

Les ontologies serveixen per estructurar informació dins els missatges ACL i facilitar l'entesa i l'intercanvi d'aquesta. Per al protocol desenvolupat s'ha decidit dissenyar una ontologia ja que el protocol segueix un sol flux d'execució i, a més a més, part de la informació que s'ha d'enviar és reutilitzada en els diversos passos en què s'ha dividit el protocol.

Com s'ha explicat en l'apartat 2.5 les ontologies estan formades per un conjunt d'accions, conceptes i predicats. Cada acció té un concepte associat i els predi-

cats poden tenir un concepte associat o no, depenent del tipus i funció del predicat. L'ontologia del protocol de migració fragmentada consta de dues accions, tres conceptes i quatre tipus de predicats que s'explicaran en detall a continuació. L'ordre seguit es correspon amb l'ordre d'ús en els diferents passos del protocol.

En el pas 1 s'envia una sol·licitud de migració a la plataforma remota. Aquest missatge té associada l'acció `request-transfer-agent` definida a la taula 4.1. Aquesta acció té un concepte associat on s'hi especifica informació referent a la migració.

Function	`request-transfer-agent`
Ontology	`fragmented-transfer-protocol-ontology`
Supported by	amm
Description	The AMM agent issues a `request-transfer-agent` request to transfer the agent to the remote platform
Domain	`ftp-rules`
Arity	1

Taula 4.1: Request Transfer Agent Action

El concepte `ftp-rules` associat a l'acció `request-transfer-agent`, definit a la taula 4.2, conté informació referent a les condicions en què es vol fer la migració.

Frame	`ftp-rules`		
Ontology	`fragmented-transfer-protocol-ontology`		
Parameter	**Description**	**Presence**	**Type**
`ftp-instance-size`	Agent instance size	Mandatory	`integer`
`ftp-code-size`	The agent code size	Mandatory	`integer`
`ftp-fragment-size`	The fragment size which the origin platform wants to do the migration process	Mandatory	`integer`
`ftp-cid`	Agent code unique identifier	Mandatory	`string`
`ftp-state`	Agent state	Optional	`byte-stream`

Taula 4.2: FTP Rules Concept

El concepte `ftp-rules` conté camps amb informació de la mida de l'agent així com de la mida del fragment amb què es desitja realitzar la migració. A més

a més envia un identificador del codi de l'agent per tal que la plataforma remota pugui avaluar si caldrà enviar el codi de l'agent o no. El camp corresponent a l'estat de l'agent s'ha decidit posar-lo com a opcional, ja que en funció del tipus de migració l'estat estarà (migració forta) o no estarà definit (migració feble).

En el pas 2 la plataforma remota respon a la sol·licitud de migració amb un predicat acceptant (AGREE) o rebutjant (REFUSE) les condicions de migració establertes per la plataforma iniciadora. Cap dels dos predicats tindrà cap concepte associat, ja que el propi predicat ja aportarà la informació necessària per a gestionar el procés de migració. Si s'accepten les condicions de la migració la plataforma remota comprovarà si té el codi a través del CID i retornarà el predicat adient definit a la taula 4.3.

Communicative Act	`agree`	
Ontology	`fragmented-transfer-protocol-ontology`	
Predicate symbol	**Arguments**	**Description**
`code-is-needed`		The agent code is not present in the remote platform and it must be sent.
`code-is-not-needed`		The agent code is already present in the remote platform and it does not have to be sent.

Taula 4.3: Agree Predicates

En canvi si les condicions en què s'ha de dur a terme la migració no s'accepten, es retornarà un predicat de tipus REFUSE, definit a la taula 4.4, indicant que les condicions no són correctes i es cancel·larà la migració.

Communicative Act	`refuse`	
Ontology	`fragmented-transfer-protocol-ontology`	
Predicate symbol	**Arguments**	**Description**
`not-enough-space`		There is not enough space in the remote platform.
`agent-too-big`		Agent size is too big and can not be accepted in the remote platform.
`reject-fragment-size`		Fragment size is not suitable.

Taula 4.4: Refuse Predicate

Si s'han acceptat les condicions, es passa al pas 3 on s'envien els fragments

de l'agent encapsulats en missatges de tipus INFORM. Aquests missatges seran predicats, definits a la taula 4.5, amb un concepte associat definit a la taula 4.6. Convé fer notar que aquest predicat i el seu concepte associat, són els mateixos que s'utilitzen en el pas 5.

Communicative Act	`inform`	
Ontology	`fragmented-transfer-protocol-ontology`	
Predicate symbol	**Arguments**	**Description**
`Fragment Sent`	FTP Transfer Agent Concept	Message filled with agent fragment.

Taula 4.5: Inform Predicates

El concepte `ftp-transfer-agent` contindrà tota aquella informació necessària per tal que la plataforma remota pugui identificar quin fragment de l'agent ha rebut. Aquestes dades són `ftp-fragment-id` que servirà per situar cada fragment dins del flux de la transferència, `ftp-fragment-aid` que és l'identificador únic de l'agent (AID) al qual correspont el fragment, `ftp-fragment-type` que servirà per saber si el fragment rebut forma part de les dades o bé del codi de l'agent i finalment el `ftp-fragment-content` que és el propi fragment de l'agent.

Frame	`ftp-transfer-agent`		
Ontology	`fragmented-transfer-protocol-ontology`		
Parameter	**Description**	**Presence**	**Type**
`ftp-fragment-content`	agent portion which fits to `ftp-fragment-size`	Mandatory	`byte-stream`
`ftp-fragment-id`	fragment number that identifies the fragment in the migration	Mandatory	`integer`
`ftp-fragment-aid`	agent unique identifier that fragment belongs	Mandatory	`aid`
`ftp-fragment-type`	identifies if the fragment belongs to data ("instance") or code ("code")	Mandatory	`string`

Taula 4.6: FTP Transfer Agent Concept

Si algun dels fragments no ha arribat en el temps previst, aleshores el pas 4

s'encarrega de tornar-lo a demanar. Es defineix l'acció `request-fragment` en la taula 4.7 amb el concepte associat `ftp-fragment-information` definit a la taula 4.8.

Function	`request-fragment`
Ontology	`fragmented-transfer-protocol-ontology`
Supported by	amm
Description	The AMM agent issues a `request-fragment` request to transfer a lost fragment to the remote platform
Domain	`ftp-fragment-information`
Arity	1

Taula 4.7: Request Fragment Action

El concepte `ftp-fragment-information` conté tota la informació necessària perquè la plataforma origen pugui cercar i reconstruir el fragment sol·licitat per la plataforma remota. Aquesta informació és: `ftp-fragment-aid` és l'AID al qual correspon el fragment sol·licitat, el `ftp-fragment-id` i el `ftp-fragment-size` que serviran per cercar i tornar a crear la porció de l'agent al que correspon el fragment sol·licitat i el `ftp-fragment-type` per saber si el fragment sol·licitat forma part de les dades o bé del codi de l'agent.

Frame	`ftp-fragment-information`		
Ontology	`fragmented-transfer-protocol-ontology`		
Parameter	**Description**	**Presence**	**Type**
`ftp-fragment-aid`	agent unique identifier that lost fragment belongs	Mandatory	`aid`
`ftp-fragment-id`	fragment number that identifies the fragment in the migration	Mandatory	`integer`
`ftp-fragment-type`	identifies if the fragment belongs to data or code	Mandatory	`string`
`ftp-fragment-size`	The fragment size which the origin platform wants to do the migration process	Mandatory	`integer`

Taula 4.8: FTP Fragment Information Concept

Si sorgeix qualsevol problema en algun dels passos del protocol es genera un missatge de tipus FAILURE i es cancel·la la migració. En la taula 4.9 es mostren els possibles missatges d'error que poden aparèixer en el transcurs de la migració.

Communicative Act	`failure`	
Ontology	`fragmented-transfer-protocol-ontology`	
Predicate symbol	**Arguments**	**Description**
`instance-size-error`	String	Error instance size not found.
`code-size-error`	String	Error code size not found.
`fragment-size-error`	String	Error fragment size not found.
`cid-error`	String	Error cid not found.
`bad-formed-msg-error`	String	Message received not properly created.
`extracting-content-error`	String	Error extracting message content.
`registration-error`	String	Error registering agent to the remote location.
`action-error`	String	Action received is not valid.
`null-action-error`	String	null action received.
`protocol-error`	String	Migration protocol is not valid.
`conversation-id-error`	String	Conversation ID is not valid
`agent-entry-error`	String	Agent entry not found to the remote location.
`migration-service-error`	String	Error contacting migration service to notice failure.
`out-of-sequence-error`	String	Message received is out of sequence.
`fragment-id-error`	String	Error fragment number identifier not found.
`fragment-aid-error`	String	Error agent unique identifier not found.
`fragment-type-error`	String	Error fragment type not found.

Taula 4.9: Failure Predicates

4.4 Resum

En aquest capítol s'ha explicat el disseny del protocol de migració fragmentada que s'ha fet atenent als requisits establerts en el capítol d'anàlisi. S'ha explicat amb detall els passos amb què s'ha dividit el protocol així com també els protocols d'interacció definits per FIPA que s'han utilitzat. Finalment s'ha detallat el disseny de l'ontologia associada al protocol dissenyat que permetrà estructurar la informació dels missatges que prenen part en una migració d'un agent mòbil.

Capítol 5

Implementació

En aquest capítol s'explicarà com s'ha organitzat el codi a partir del disseny del protocol que s'ha fet, així com les decisions d'implementació que s'han pres per obtenir un millor rendiment del protocol. Finalment es presentaran els mètodes de suport al protocol que s'han implementat dins del servei de migració.

5.1 Organització del Codi

Un cop fet el disseny del protocol s'explicarà com s'ha implementat aquest disseny, és a dir, com s'ha estructurat el codi, quines classes s'han fet servir i quines decisions d'implementació s'han pres per tal d'oferir la màxima eficiència en l'execució del protocol.

El protocol Fragmented Transfer Protocol (FTP) proposat s'ha implementat amb el llenguatge de programació Java i s'ha desenvolupat dins de l'*Add-on* de mobilitat JIPMS per a la plataforma JADE. Cada classe en què s'ha dividit el protocol correspon a un *Behaviour* (veure Figura 5.1).

Un aspecte que cal tenir en compte és que si bé l'enviament de fragments des de la plataforma origen es fa de manera seqüencial i ordenada, no es pot garantir

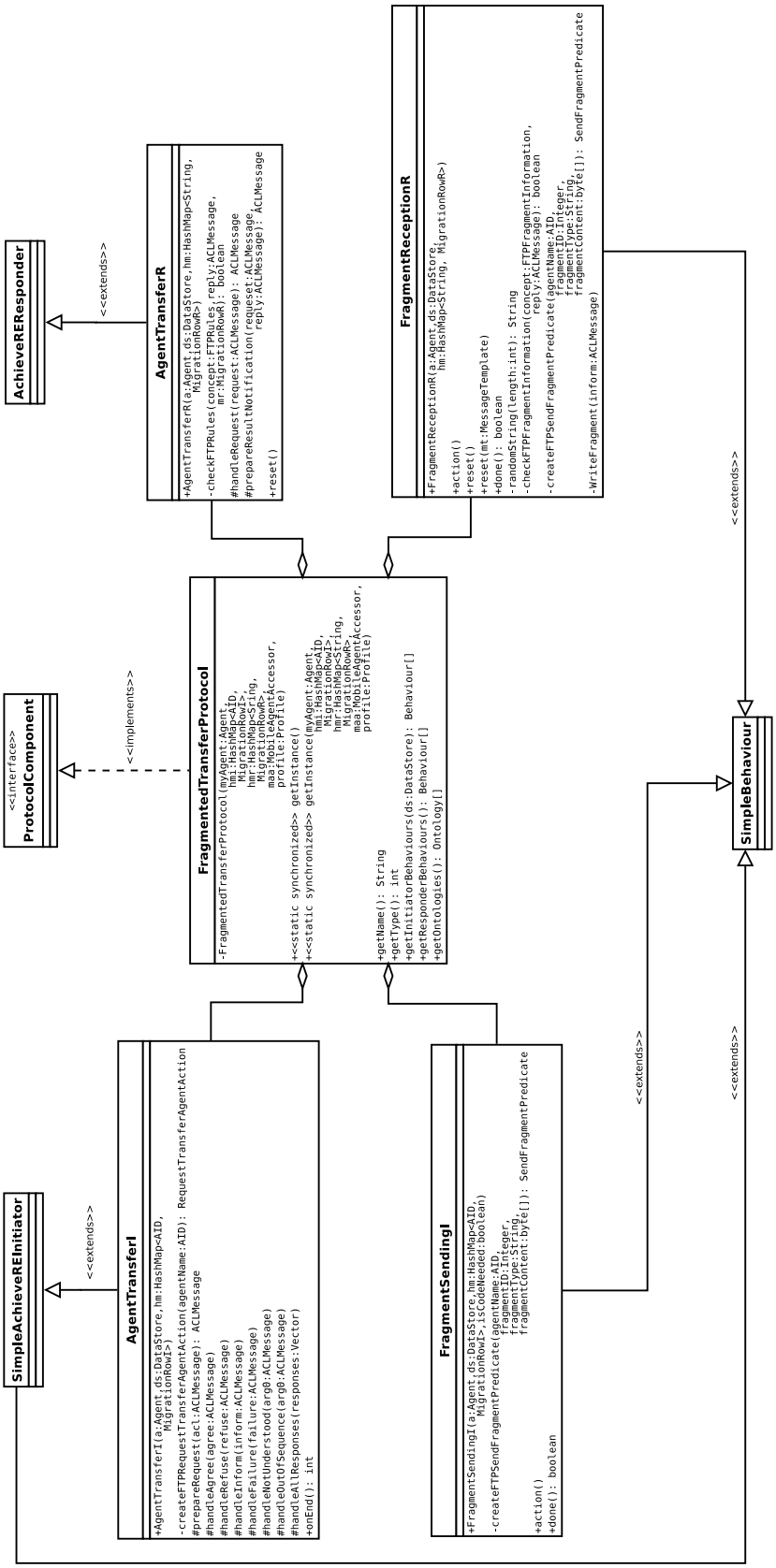

Figura 5.1: Diagrama de Classes del Protocol de Migració Fragmentada

que els fragments arribin amb ordre. Per tant s'ha dissenyat un sistema que en tot moment té en compte quins fragments han arribat i quins encara no ho han fet, a més a més també es controla el *timeout* adaptatiu de recepció de fragments explicat en la secció 4.1.3 i definit per la fórmula:

```
timeout=(2*((instanceSize+codeSize)*TIMEOUT1K)+TIMEINIT);
```

El temps d'espera màxim per a la recepció de tots els fragments vindrà determinat per la mida de l'agent a transferir. Es donarà de marge dues vegades el temps esperat per a la recepció dels fragments, per tal d'evitar sol·licituds de fragments innecessàries provocats per situacions de congestió puntuals de la xarxa o de la pròpia plataforma. La constant TIMEOUT1K correspon al temps que s'estima que pot tardar un fragment d'1KB en arribar, i la constant TIMEINIT és un cert marge de temps que es dóna corresponent a l'arrencada i estabilització de la plataforma. Ambdues variables s'acabaran d'ajustar amb valors adients extrets dels resultats de tests.

En el cas de recepció dels fragments de codi, com que és un fitxer Java ARchive (JAR), s'ha implementat la reconstrucció de manera que no sigui necessari mantenir tot el fitxer carregat en memòria, sinó que a mesura que arriben els fragments s'escriuen directament en un fitxer a disc. Per realitzar aquesta funcionalitat s'han hagut d'incorporar un conjunt de mètodes específics dins del servei de migració JIPMS, aquests mètodes s'explicaran en l'apartat 5.2.

En el següent apartat relacionarem els diferents *Behaviours* dels què consta el protocol amb els passos del disseny que implementa. A més a més per a la implementació de l'ontologia s'ha creat un *package* que conté totes les classes corresponents a les accions, conceptes i predicats de l'ontologia.

5.1.1 *Behaviours*

La implementació del protocol consta d'una classe principal anomenada *FragmentedTransferProtocol* que implementa la interfície *ProtocolComponent*, que és la classe base de tots el protocols a JIPMS. Aquesta classe principal conté quatre *Behaviours*: dos *Initiator Behaviour* que s'encarreguen d'iniciar l'acte comunicatiu i dos *Responder Behaviour* que s'encarreguen d'atendre i gestionar els missatges rebuts.

- **AgentTransferI**: És l'*Initiator Behaviour* principal i s'encarrega d'implementar la funcionalitat dels passos 1, 2 i 6 (veure Figura 4.1)de la part iniciadora de la migració. Estén la classe *SimpleAchieveREInitiator* de JADE. Aquesta és una implementació senzilla del protocol d'interacció *FIPA Request Interaction Protocol*, la qual defineix l'estructura de l'intercanvi de missatges d'aquest protocol d'interacció amb un sèrie de mètodes que s'han de sobreescriure per obtenir la funcionalitat desitjada de la part iniciadora de la migració.

  Com s'ha comentat en l'apartat 4.1.2 no hi ha cap protocol d'interacció definit per FIPA que s'ajusti exactament al disseny del protocol proposat, per això s'han hagut de crear *behaviours* específics per a completar la funcionalitat que es volia aconseguir. El pas 3 s'ha implementat a través del *behaviour* **FragmentSendingI**. Quan l'execució del *behaviour* **FragmentSendingI** acaba, es retorna al punt d'execució del mètode *handleAgree* on havia quedat i passa al pas 6 on espera el resultat de la migració.

- **FragmentSendingI**: És el *behaviour* iniciador que s'encarrega de crear i enviar els fragments de l'agent del pas 3 del protocol. Estén la classe *SimpleBehaviour* i és llançat un cop s'ha rebut la confirmació de les condicions de migració per part de la plataforma iniciadora. En cada execució només

envia un fragment en comptes crear i enviar els fragments amb un bucle, ja que s'aprofita el fet que el *Behaviour* s'executa iterativament. Aquesta decisió d'implementació s'ha pres perquè JADE té un planificador de *behaviours* que emmagatzema tots els *behaviours* que s'han d'executar en una cua i els hi va assignant temps d'execució a mesura que hi ha recursos disponibles. Si s'hagués implementat l'enviament de fragments amb un bucle el *behaviour* no hagués permès executar altres *behaviours* fins que no hagués enviat tots els fragments d'una migració, en cas d'haver-hi migracions simultànies en una mateixa plataforma. Aquest fet penalitzaria aquelles migracions d'agents petits en front de migracions d'agents grans, ja que per realitzar la migració d'un agent petit seria necessari esperar que es creessin i enviessin tots els fragments de la migració de l'agent gran en curs. Aquesta situació podria provocar que s'exhaurissin els *timeouts* en aquelles migracions on es transferissin agents petits, ja que el seu temps de migració estimat també seria petit.

Amb la implementació realitzada però, es crea i s'envia un fragment i es para l'execució del behaviour. Aquest va a parar a la cua del planificador de *behaviours* de JADE, que li tornarà a otorgar recursos quan estiguin disponibles.

- **AgentTransferR**: És el *Responder Behaviour* principal i s'encarrega d'implementar la funcionalitat dels passos 1 i 2 de la part participant de la migració descrits en la Figura 4.1. Estén la classe *AchieveREResponder* de JADE que és una implementació del protocol d'interacció *FIPA Request Interac tion Protocol* i que defineix l'estructura de l'intercanvi de missatges d'aquest protocol d'interacció amb un sèrie de mètodes que s'han de sobreescriure per a obtenir la funcionalitat desitjada de la part participant de la migració. La principal diferència entre les classes *SimpleAchieveRERes-*

ponder i *AchieveREResponder* rau en què la segona permet registrar un *behaviour* per a qualssevol dels estats del protocol d'interacció *FIPA Request Interaction Protocol* que implementa. Per aquest motiu s'ha considerat oportú utilitzar la classe *AchieveREResponder* i registrar el mètode *prepareResultNotifica*

tion amb un *Responder Behaviour* (**FragmentReceptionR**) que s'encarregui d'implementar els passos 3, 4, 5 i 6 del protocol de migració fragmentada.

- **FragmentReceptionR**: És el *Responder Behaviour* que s'encarrega de rebre els fragments i reconstruir l'agent a la plataforma remota, també s'encarrega de gestionar la recuperació de fragments en cas que s'hagin perdut. Per poder realitzar totes les funcionalitats dels passos 3, 4, 5 i 6 (mostrats a la Figura 4.1) s'ha pensat implementar aquest *behaviour* com una màquina d'estats, on cada estat correspongui a un dels passos del protocol a més a més d'algun estat intermig de control. La màquina d'estats dissenyada es pot veure en la figura 5.2.

  Com desenvolupar aquesta màquina d'estats suposa prendre una nova decisió d'implementació. Es pot desenvolupar bé amb un *FSMBehaviour* on cada estat corresponent a un dels passos del protocol s'implementaria amb un *behaviour*, o bé amb un model similar al de l'*AchieveREResponder* on es defineix la interacció dels missatges ACL del protocol de migració fragmentada (FTP). De les dues opcions proposades cap presenta a priori un avantatge en termes de rendiment. Seguint el principi de la navalla d'*Ockham*[1], s'ha decidit utilitzar un model basat en *AchiveREResponder* perquè es preveu que el seu desenvolupament serà més senzill i ràpid. A més a més,

[1]Navalla d'*Ockham*: principi que prové del postulat *entia non sunt multiplicanda praeter necessitatem*, que traduït voldria dir *Els ens no s'han de multiplicar sens necessitat*. Dit amb altres paraules: essent dues solucions iguals en tots els aspectes, la més simple serà la millor.

d'aquesta manera s'utilitzaran menys *behaviours* i en condicions de migracions concurrents pot afavorir la descongestió del planificador de JADE.

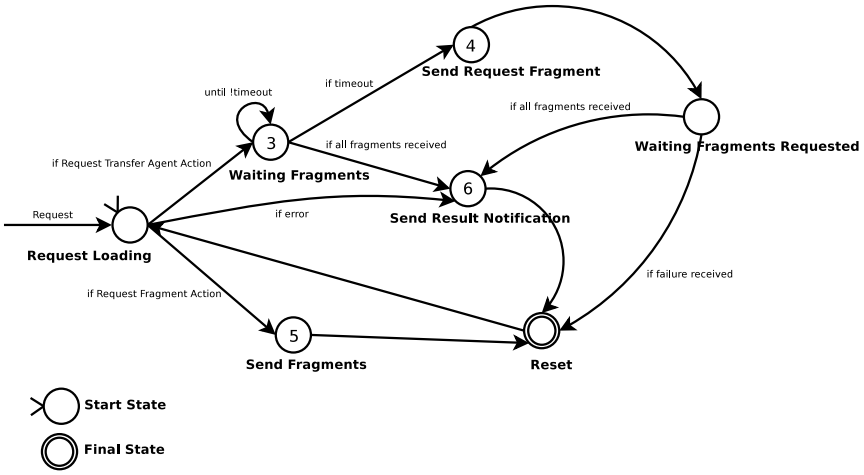

Figura 5.2: Diagrama d'estats del *Behaviour* FragmentReceptionR

La Figura 5.2 mostra els estats amb què s'ha dividit el *Behaviour* **Fragment ReceptionR** que s'encarrega d'executar les funcionalitats del passos 3,4,5 i 6 del protocol presentats en la Figura 4.1. Com que aquest *Behaviour* és força complex, s'explicarà de manera succinta cada un dels estats que el formen.

– **Request Loading State**: L'estat *Request Loading* s'encarrega de recuperar l'acció del darrer missatge REQUEST rebut. Aquest pas és necessari, ja que tots els missatges de tipus REQUEST d'una mateixa migració es reben en el pas 1. Per diferenciar els missatges de diferents migracions s'utilitza el *MigrationID*, que és un identificador únic de la migració associat a tots els missatges ACL que hi prenen part, i per diferenciar els missatges dels diferents protocols d'interacció executant-se concurrentment s'utilitza el *ConversationID*. La particularitat en la rebuda del primer missatge (REQUEST) en el pas 1 per part del *be-*

haviour és que encara no té associat cap *ConversationID* concret, per tant rebrà tots els missatges de tipus REQUEST que intervinguin en una mateixa migració. Això suposa que els missatges dels passos 1 i 4 els atendrà el mateix *Responder Behaviour*, concretament el *behaviour* **AgentTransferR**. A partir de l'acció recuperada corresponent a l'últim missatge REQUEST rebut es podrà determinar si ens trobem en el pas 3 o bé en el pas 5 del protocol.

- **Waiting Fragments State**: L'estat **Waiting Fragments** s'encarrega de rebre els fragments de l'agent i posteriorment reconstruir-ne la instància (dades) i el codi si tot ha anat correctament. Si s'esgota el *time out* i no s'han rebut tots els fragments, es passarà a l'estat **Send Request Fragment** (pas 4), i si s'han rebut tots els fragments correctament, es passarà a l'estat **Send Result Notification** (pas 6).
- **Send Request Fragment State**: L'estat **Send Request Fragment** s'encarrega de dur a terme la funcionalitat del pas 4, és a dir, d'enviar un missatge REQUEST a la plataforma origen per a cada fragment de l'agent que no s'ha rebut. Quan ha sol·licitat tots els fragments que no s'han rebut, passa a l'estat **Waiting Fragments Requested**.
- **Waiting Fragments Requested State**: L'estat **Waiting Fragments Requested** s'encarrega de dur a terme la funcionalitat del pas 5 de la part participant, és a dir, rebre els fragments sol·licitats i reconstruir la part del codi i la instància de l'agent. Aquest estat realitza la mateixa funció que l'estat **Waiting Fragments**, però amb la particularitat que pot rebre missatges de tipus FAILURE, per aquest motiu s'ha decidit implementar-ho en un nou estat, ja que en el pas 3 només es poden rebre missatges de tipus INFORM. Si en l'estat **Waiting Fragments Requested State** es rep un missatge de tipus FAILURE, aquest es

reenviarà a la plataforma origen.

– **Send Result Notification State**: L'estat **Send Result Notification** s'encarrega de dur a terme la funcionalitat del pas 6 del protocol. Si han arribat tots els fragments correctament, s'encarrega de registrar el codi a la plataforma a través d'uns mètodes que hem implementat específicament dins del servei de migració JIPMS que s'explicaran amb més detall en l'apartat 5.2. També s'encarrega d'instal·lar la instància de l'agent a la plataforma i d'enviar el missatge corresponent del resultat de la migració. Enviarà un INFORM si tot ha anat correctament o un FAILURE amb el tipus d'error si hi ha hagut qualsevol problema durant el procés de transferència de l'agent mòbil.

– **Send Fragment State**: L'estat **Send Fragment** s'encarrega de dur a terme la funcionalitat del pas 5 de la part iniciadora, és a dir, s'encarrega de cercar i crear els fragments de l'agent sol·licitats per la plataforma remota.

– **Reset State**: L'estat **Reset** s'encarrega de restaurar els valors per defecte de les variables de control del *behaviour* **FragmentReceptionR**. Alguns exemples de variables de control són l'estat de la màquina d'estat en el que es troba l'execució del *behaviour*, la llista de fragments rebuts, la mida de l'agent, etc. Aquest estat és necessari, ja que la creació dels *behaviours* es fa quan s'inicia la plataforma (20 *Initiator Behaviours* i 20 *Responder Behaviours*), aquests *behaviours* seran reutilitzats posteriorment per futures migracions . Per aquest motiu cal restaurar els valors per defecte de les variables de control.

5.1.2 Ontologia

S'ha implementat l'ontologia del protocol de migració fragmentada seguint el disseny proposat a la secció 4.3, la qual s'ha implementat segon el sistema definit per JADE on cada part de l'ontologia està representada per una classe. Totes les classes implementades s'han agrupat al *package jade.core.migration.ontology.ftp* i estan representades en el diagrama de classes de la figura 5.3.

En l'ontologia implementada es poden distingir 5 tipus d'elements: La classe principal de l'ontologia, el vocabulari, les accions, els conceptes i els predicats.

- **Ontologia**: És la classe principal i és on es defineix l'estructura de l'ontologia: les accions, predicats i conceptes que té, així com els atributs de cada concepte.

- **Vocabulari**: S'hi defineixen com a constants el nom dels atributs i classes dels conceptes, accions i predicats que conté l'ontologia. Dóna contingut semàntic a les expressions que fa servir l'ontologia per comunicar-se.

- **Accions**: Defineixen les tasques que s'han de dur a terme. Cada acció té un concepte associat que es pot configurar a través dels mètodes *set* i *get* de l'acció. S'han definit 2 accions: una pel pas 1 i l'altra pel pas 4 del protocol.

- **Predicats**: Defineixen una afirmació (sentència) de la qual no se n'espera resposta (al contrari que les accions). Hi ha predicats com el definit per al pas 3 que tenen un concepte associat que poden gestionar a través dels mètodes *set* i *get*. En canvi hi ha predicats com els definits per al pas 2 que no tenen cap concepte associat, perquè la informació que han de transmetre es pot extreure de la mateixa instància del predicat.

- **Conceptes**: Defineixen la informació a transmetre i estan associats a una acció o bé a un predicat. A través dels mètodes *set* i *get* es pot configurar el

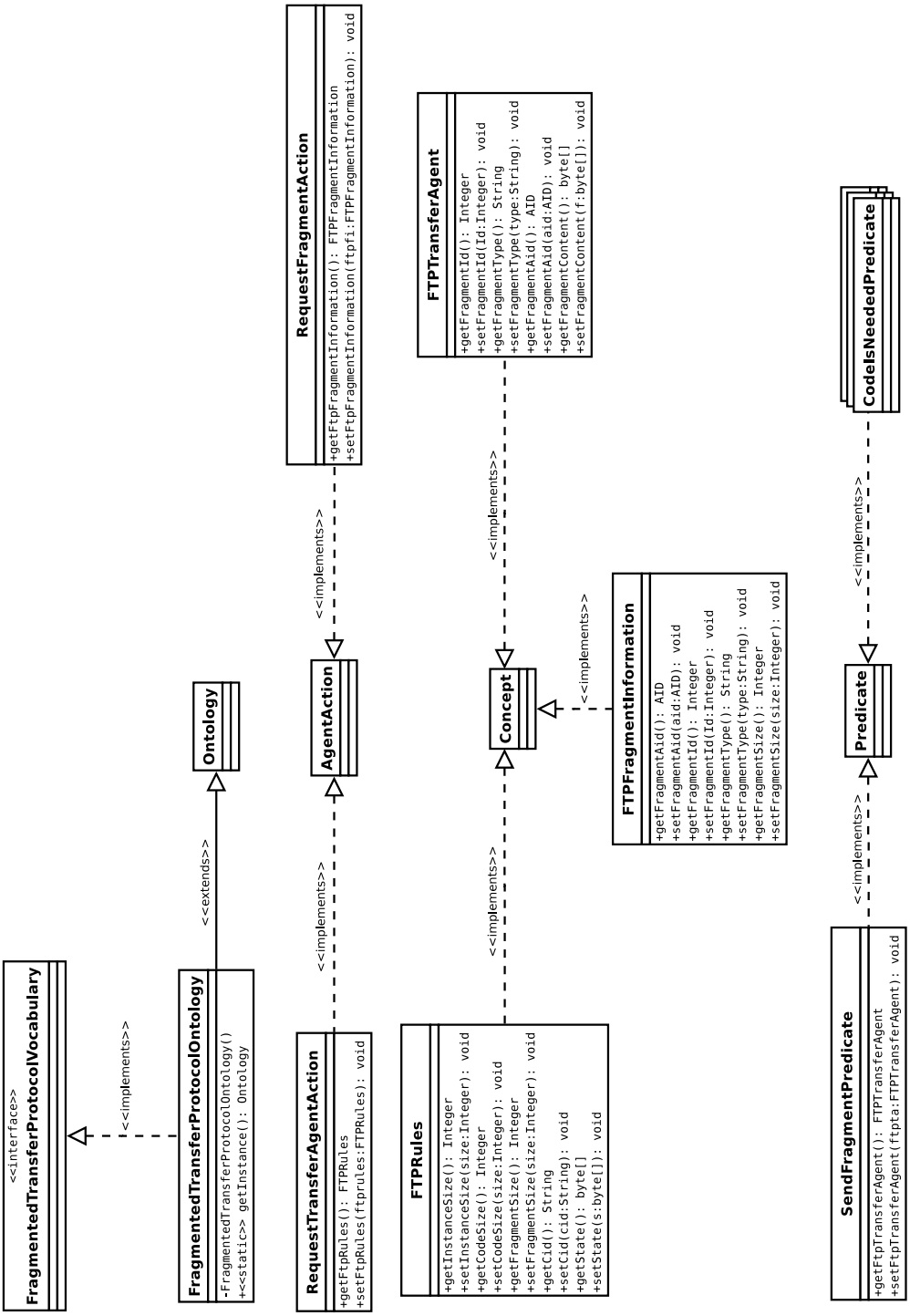

Figura 5.3: Diagrama de Classes de l'Ontologia del Protocol de Migració Fragmentada

valor dels atributs. S'han definit 3 conceptes: un pel pas 1, un pel pas 3 i 5 i un pel pas 4 del protocol.

5.2 Mètodes de suport al protocol

Com s'ha explicat anteriorment els fragments de codi a mesura que es reben s'escriuen a disc a la posició corresponent de l'arxiu JAR. Un cop s'han rebut tots els fragments s'ha de registrar el codi a la plataforma i vincular-lo a l'agent mòbil. Com que en l'actual versió del servei de migració JIPMS no hi havia cap mètode que permetés registrar el codi d'un agent directament des d'un fitxer JAR ja gravat a disc, s'han implementat un conjunt de mètodes amb l'objectiu d'optimitzar el procés de registre del codi de l'agent a la plataforma remota.

S'han implementat dues noves funcions en la classe *MobileAgentAccessor* del servei de migració. Aquesta classe proporciona un conjunt de mètodes per tractar les diferents parts de les què consta un agent mòbil.

```
public synchronized boolean registerJar(String cid, File codeJarPath)
```

La funció *registerJar* permet registrar el codi d'un agent a la plataforma remota a través de l'identificador únic i de la ruta del fitxer JAR. El valor lògic que retorna la funció serveix per indicar si s'ha pogut realitzar el registre del codi correctament. Aquesta funció ha estat definida com a *synchronized* ja que ha d'accedir a recursos de la plataforma que poden ser consultats i modificats per altres processos de migració concurrents. Aquesta funció és cridada des de la funció *registerAgentCodePath*.

```
public boolean registerAgentCodePath(AID agent, String jarCid, String codeJarPath)
```

La funció *registerAgentCodePath* registra el codi a través de la funció *registerJar* i vincula el fitxer JAR registrat a l'agent mòbil que el fa servir.

5.3 Resum

En aquest capítol s'ha explicat com s'ha organitzat el codi a partir del disseny que s'ha realitzat del protocol de migració fragmentada explicat en el capítol de disseny. També s'han comentat amb detall les decisions d'implementació que s'han adoptat amb l'objectiu d'oferir la màxima eficiència en l'execució del protocol. A més a més s'han mostrat el conjunt de conceptes, accions i predicats que s'han creat per implementar l'ontologia del protocol. Finalment s'han comentat els mètodes de suport al protocol FTP que s'han implementat i incorporat dins del servei de migració JIPMS.

Capítol 6

Proves

En aquest capítol s'explicarà l'entorn on s'han dut a terme els tests del protocol, així com el disseny de les proves a realitzar. Finalment s'analitzaran els resultats obtinguts i se'n farà una interpretació exhaustiva en els diferents escenaris en què s'han realitzat els tests.

6.1 Introducció

Un cop realitzada la implementació del protocol de migració fragmentada d'agents mòbils sobre la plataforma JADE, és moment de dur a terme els tests que verifiquin el seu correcte funcionament. S'han realitzat dos tipus de test per al protocol: un test de funcionament que comprovi que la migració completa d'un agent mòbil es fa correctament i uns tests de rendiment que permetin avaluar i comparar el temps que es tarda en transferir diversos agents mòbils de diferents grandàries.

6.2 Entorn de Proves

Les proves de funcionament s'han realitzat en els ordinadors que el grup de recerca SeNDA posa a disposició dels seus projectistes per a la realització dels projectes fi de carrera d'agents mòbils. Els ordinadors tenen instal·lat el sistema operatiu GNU/Linux. El maquinari utilitzat en les proves de funcionament no és especialment rellevant, ja que el què pretén aquest test és comprovar que es duen a terme correctament tots els passos descrits en el disseny del protocol.

Les proves de rendiment s'han realitzat en el *cluster* dedicat del què disposa el departament per poder dur a terme els tests de migració d'agents mòbils. Aquest *cluster* està format per dos ordinadors idèntics amb un processador Intel Pentium 4 a 2.0 Ghz, amb una memòria cau de 512 KB i 755 MB de memòria RAM. Ambdós ordinadors estan connectats entre si a través d'un commutador D-Link 10/100 Ethernet dedicat. La connexió és de 100 Mbps amb entrega directa. Els dos ordinadors tenen instal·lat el sistema operatiu GNU/Linux, concretament la distribució Fedora Core 5 amb el *kernel* 2.6.17-2145 i una memòria virtual de 2047 MB.

6.3 Disseny de les Proves

Per als tests de funcionament no s'ha fet cap disseny específic. S'ha hagut de configurar la *suite* de test de funcionament que ofereix el servei de migració JIPMS perquè fes servir el protocol de migració fragmentada. En el test de funcionament s'inicien dues plataformes (en un mateix ordinador) i es transfereix l'agent de la plataforma origen a la plataforma destí. L'agent que s'utilitza per a la migració és un agent estàndard sense cap funcionalitat específica associada, simplement llegeix l'itinerari d'un fitxer i fa una migració a la plataforma destí. Aquest agent és de mida reduïda, la instància (dades) de l'agent ocupa 3 KB i el codi ocupa 5 KB

aproximadament. Mitjançant un paràmetre a l'iniciar la plataforma es pot configurar la mida del fragment amb què es vol transferir l'agent, s'han fet diverses proves de migració amb mides de fragment entre 1 KB i 10 KB per comprovar que el protocol crea, envia, rep i reconstrueix correctament els fragments de l'agent.

Per als test de rendiment s'han creat un conjunt d'agents específics que s'han incorporat a la *suite* de tests de rendiment del servei de migració JIPMS [JIPMS], aquesta *suite* de tests mesura el temps que tarda un agent en fer un itinerari complet, és a dir, en migrar de la plataforma inicial a la plataforma destí i tornar a la plataforma inicial. Els tests de rendiment del protocol desenvolupat s'han fet tenint en compte diversos aspectes:

- **Mida dels Agents**: Aquest aspecte s'ha dividit en dos grans blocs: tests d'instància i tests de codi. S'han creat agents amb diverses mides de codi (entre 5 KB i 1 MB) deixant la gràndaria de la instància fixa (3 KB aproximadament) per als tests de codi. També s'han creat agents amb el mateix disseny però per als tests d'instància, és a dir, agents amb múltiples grandàries d'instància (entre 5 KB i 1 MB) deixant la mida del codi fixa (5 KB aproximadament). Cal destacar que per a la creació dels agents de codi s'ha desactivat l'opció de compressió dels arxius JAR, ja que d'aquesta manera era més àgil obtenir una mida exacta dels arxius de codi a partir dels fitxers de *padding* dels JARs.

- **Concurrència**: Un aspecte interessant és el d'avaluar el temps promig d'una migració d'un agent amb el de diverses migracions simultànies en paral·lel d'aquest mateix agent. Convé destacar que la migració de cada un dels N agents concurrents no tarda 1/N del temps total de migració, sinó que el temps que tarda és pràcticament N/N a causa de l'efecte *pipeline* en les migracions concurrents. Els temps obtinguts amb els tests de concurrència, però, s'han calculat com el promig (1/N) de temps que el *middleware*

dedica a la migració de l'agent, tot i que l'agent en realitat tarda més a migrar.

- **Nombre de Migracions**: Els tests han de permetre obtenir resultats fiables per poder-ne extreure conclusions fermes. El nombre de migracions ha variat segons la grandària dels agents a transferir, ja que pels agents més grans hi havia problemes de memòria, sobretot amb les migracions concurrents. A la Taula 6.1 es detalla el nombre de migracions que es duran a terme per cada tipus d'agent i tenint en compte la concurrència.

	No Concurrència	**Concurrència**
5 KB - 100 KB	1 instància * 100 iteracions * 2 migracions/iteració * 5 repeticions = **1000 migracions**	10 instàncies * 100 iteracions * 2 migracions/iteració * 5 repeticions = **10000 migracions**
250 KB - 1000 KB	1 instància * 10 iteracions * 2 migracions/iteració * 5 repeticions = **100 migracions**	10 instàncies * 10 iteracions * 2 migracions/iteració * 5 repeticions = **1000 migracions**

Taula 6.1: Configuració del nombre de migracions per al test de rendiment

- **Mida Fragments**: S'utilitzaran diverses mides de fragments per als tests tant d'instància com de codi. La grandària dels fragments va des dels 5 KB fins als 50 KB. Els fragments més grans de 50 KB no s'han tingut en compte ja que s'estima que no permetran obtenir millors resultats.

- **Escenaris**: S'ha cregut convenient executar les proves en diferents escenaris per tal d'avaluar el comportament del protocol en diverses situacions. La situació per defecte és la descrita en l'apartat 6.2 on les plataformes tenen una connexió de 100 Mbps amb entrega directa. Un segon escenari s'ha configurat afegint una latència de 60 ms entre les dues màquines.

Els aspectes anteriors també s'han aplicat a protocols de transferència ja exis-

tents com el Push Cache Transfer Protocol (PCTP) que és el protocol que s'utilitza per defecte en el servei de migració JIPMS per fer la transferència dels agents i també s'ha comparat amb el protocol descrit en [MIP07]. D'aquesta manera es podrà comparar la resposta del protocol FTP amb diferents configuracions d'agents a migrar i en diferents escenaris amb altres protocols.

En [MFBBR07] s'apunta que perquè un test doni uns resultats fiables s'ha de repetir l'execució com a mínim 5 vegades, per tant per qüestions de temps s'ha decidit que l'execució dels tests es repetirà 5 vegades.

Cal destacar que a causa de l'elevat nombre de migracions i que gran part dels tests es fan amb concurrència s'ha decidit augmentar els *timeouts* del protocol per tal que les migracions dels agents més grans no s'acabin cancel·lant.

Convé comentar que a l'iniciar les plataformes per a executar els diferents tests, s'ha introduït un temps d'espera de 120 segons per permetre que la plataforma iniciï tots els serveis i s'estabilitzi. Posteriorment entre les diferents repeticions dels tests s'ha introduït un temps d'espera de 60 segons pel mateix motiu. A més a més s'ha desactivat el sistema de *cache* de codi de manera que per a cada migració es transfereixi l'agent mòbil complet independentment de si la plataforma destí ja disposa del codi de l'agent o no.

6.4 Resultats

En aquest apartat es mostraran els resultats obtinguts en les proves descrites en l'apartat anterior. La presentació dels resultats s'ha dividit en tres grans blocs: Un primer bloc amb els resultats de les migracions de les dades (instància), un segon bloc de les migracions de codi i un últim bloc on es mostraran els resultats de migracions de codi en un escenari amb latència. Cada un d'aquests blocs té una primera part amb els resultats de migracions d'agents sols i una segona part amb

els resultats de migracions concurrents de 10 agents.

6.4.1 Resultats Instància

Els tests de la instància pretenen mostrar com respon el protocol de migració fragmentada davant d'agents amb diferents mides de la part de les dades.

Resultats Instància No Concurrent

		Fragment Size					
	PCTP	5 KB	10 KB	15 KB	20 KB	25 KB	50 KB
Instance 5 KB	133	300	288	288	292	292	324
Instance 10 KB	148	243	203	206	207	208	206
Instance 25 KB	227	305	398	308	445	434	420
Instance 50 KB	499	347	347	521	478	567	1114
Instance 100 KB	1760	495	476	576	684	776	1525
Instance 250 KB	11546	947	856	935	1071	1194	2219
Instance 500 KB	39869	1692	1499	1594	1670	1826	3099
Instance 1000 KB	141939	-	2836	2962	3057	3343	5832

Taula 6.2: Resultats Migració de la Instància No Concurrent (ms)

A la Taula 6.2 es poden veure els resultats obtinguts amb els tests d'instància no concurrents. En la Figura 6.1 s'observa com amb el protocol PCTP a mesura que creix la mida de la instància de l'agent a transferir el temps augmenta de manera exponencial, mentre que amb el protocol FTP el temps creix de forma més lineal.

En la Figura 6.2 es pot veure més en detall com responen ambdós protocols en migracions d'agents petits. S'observa que per a agents amb instàncies entre 5 KB i 25 KB el protocol PCTP és lleugerament més ràpid. Aquest fet es deu a què en el protocol PCTP la instància de l'agent s'envia en un primer missatge juntament amb les condicions de migració, mentre que en el protocol proposat la instància s'envia després d'haver establert les condicions de migració. Per tant per a mides de instàncies petites (5 KB -25 KB) no surt a compte fragmentar la instància.

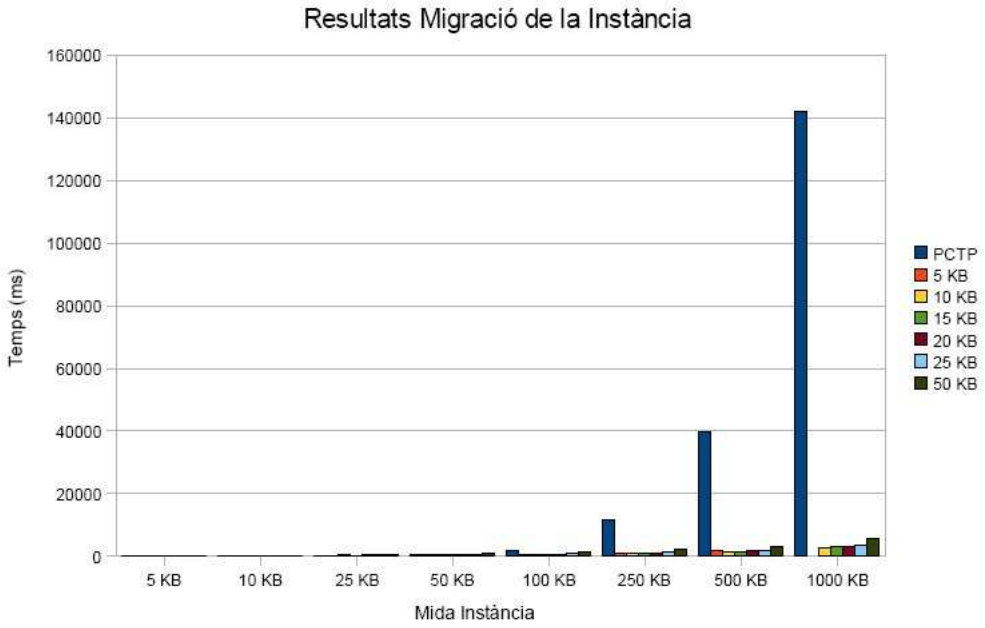

Figura 6.1: Gràfic Resultats Migració de la Instància No Concurrent

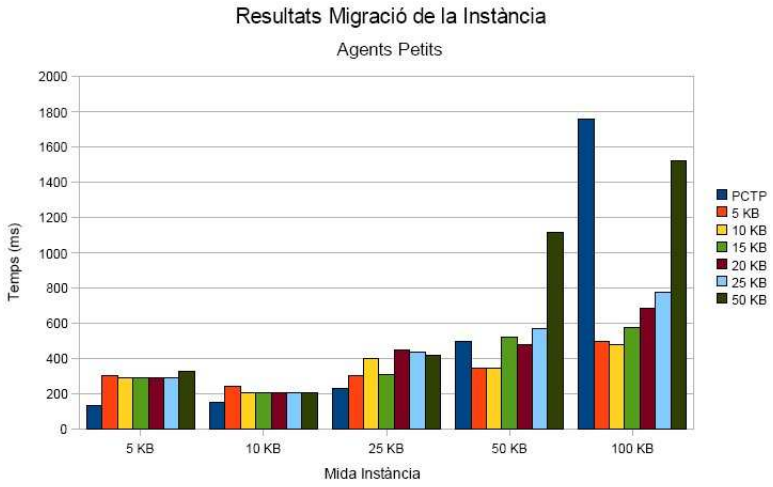

Figura 6.2: Gràfic Resultats Migració de la Instància No Concurrent d'Agents Petits

En canvi a partir de les instàncies de 50 KB el protocol FTP presenta millors resultats. Si ens fixem en la Figura 6.1 veiem com a mesura que la grandària de la instància creix, les diferències de temps entre el protocol PCTP i FTP són cada vegada més importants. En el cas de la instància de 1000 KB amb el fragment de 10 KB s'obté una millora del 98%.

En la Figura 6.3 es poden veure els resultats dels tests d'instància per les diferents mides de fragment triades. S'han exclòs els resultats del protocol PCTP per veure més clarament els resultats de la resposta del protocol FTP. S'observa com amb el fragment de 50 KB la progressió dels temps és exponencial i que la mida de fragment òptima és la de 10 KB.

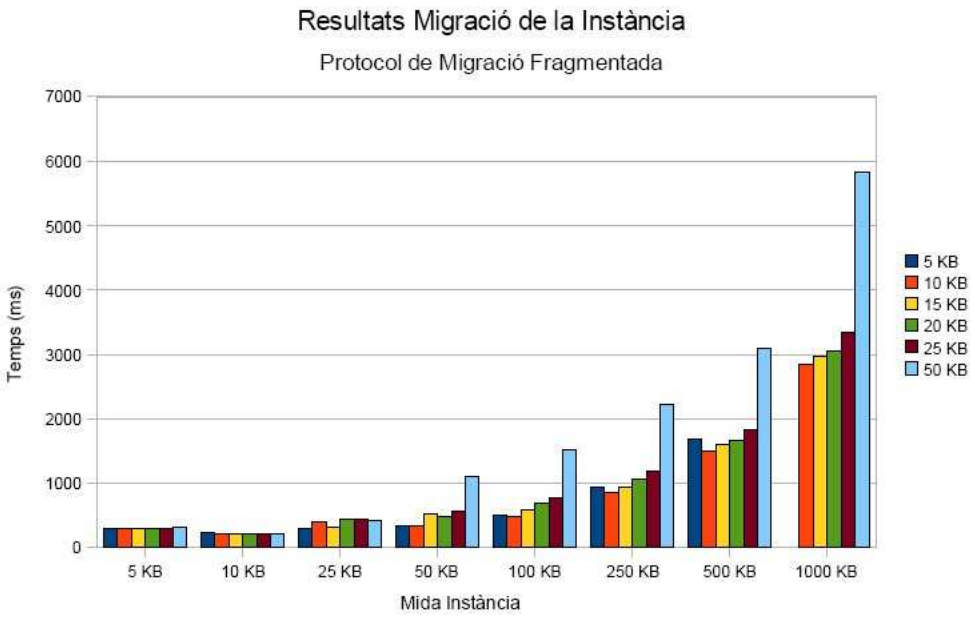

Figura 6.3: Gràfic Resultats Migració de la Instància No Concurrent del protocol FTP

		Fragment Size					
	PCTP	5 KB	10 KB	15 KB	20 KB	25 KB	50 KB
Instance 5 KB	81	128	185	185	186	192	192
Instance 10 KB	91	138	146	138	140	139	139
Instance 25 KB	160	186	209	222	321	337	310
Instance 50 KB	413	264	256	326	360	408	923
Instance 100 KB	1614	397	369	448	536	588	1141
Instance 250 KB	10726	887	744	811	840	1042	1762
Instance 500 KB	33456	1815	1406	1409	1489	1609	2944
Instance 1000 KB	113339	-	3806	3936	3722	4031	7480

Taula 6.3: Resultats Migració de la Instància Concurrent (ms)

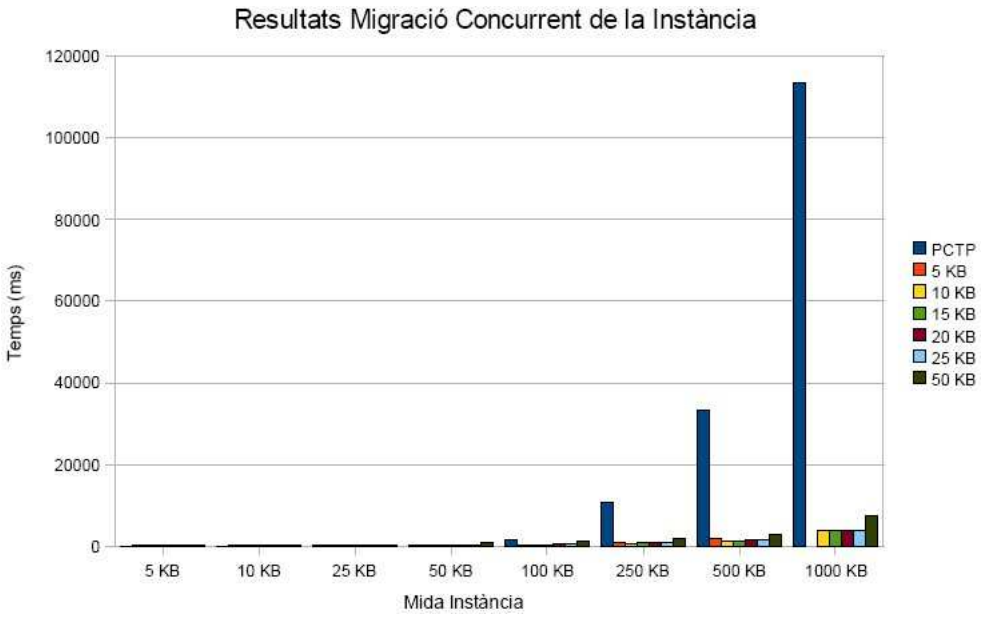

Figura 6.4: Gràfic Resultats Migració de la Instància Concurrent

Resultats Instància Concurrent

A la Taula 6.3 es poden veure els resultats obtinguts amb els tests d'instància concurrents. La Figura 6.4 mostra l'evolució dels temps de migracions simultàncies de deu agents amb instàncies entre 5 KB i 1000 KB. Es pot observar com la tendència és la mateixa que la que s'ha vist a la Figura 6.1, és a dir, el protocol PCTP presenta una resposta exponencial a mesura que creix la mida de la instància, en canvi el protocol FTP mostra una resposta més moderada.

En la Figura 6.5 es poden veure els resultats dels tests d'instància concurrents per les diferents mides de fragment triades. S'observa com amb el fragment de 50 KB la progressió dels temps és més marcadament exponencial. Pels agents més petits (5 KB - 25 KB) el fragment de 5 KB dóna millors resultats, en canvi pels agents de més de 25 KB el fragment de 10 KB és el que presenta un millor rendiment. Per tant amb els tests concurrents de la instància no hi ha una mida de fragment òptima, sinó que aquesta va en funció de la mida de la instància de l'agent a transferir.

En la Figura 6.6 es pot veure més en detall les diferències entre els tests de les migracions concurrents de la instància i les migracions de la instància amb un únic agent migrant simultàniament. S'observa com en general les migracions concurrents presenten millors resultats envers la migració del mateix agent en solitari a causa de l'efecte *pipeline* que es produeix en les migracions concurrents. A més a més JADE gestiona els agents amb *threads* [GCJADE], característica que poden aprofitar les migracions concurrents. No obstant això en l'agent de 1000 KB s'observa com les migracions concurrents presenten temps més elevats. La causa d'aquest canvi de tendència (que ja s'observa en l'agent de 500 KB amb el fragment de 5 KB) és per la sobrecàrrega de recursos que suposa la creació dels fragments de l'agent amb instàncies grans, ja que per a crear els fragments de la instància es mantenen dues còpies en memòria de la instància. Una prova

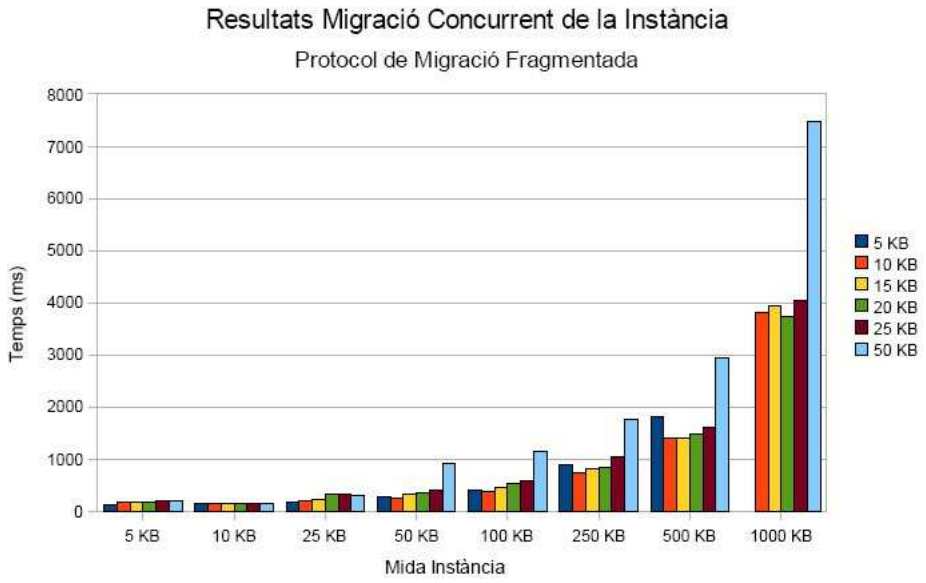

Figura 6.5: Gràfic Resultats Migració de la Instància Concurrent del protocol FTP

d'aquest fet és que no s'ha pogut dur a terme el test concurrent de la instància de 1000 KB amb el fragment de 5 KB, pel sobrecost de memòria RAM i de recursos que suposa gestionar la gran quantitat de missatges ACL que es creen per a la transferència dels agents de 1000 KB amb fragments de 5 KB.

6.4.2 Resultats Codi

Els tests de codi pretenen mostrar com respon el protocol de migració fragmentada davant de migracions amb diferents mides de la part del codi de l'agent. Aquest codi en JADE està representat per un fitxer JAR on hi ha emmagatzemades les classes compilades corresponents als diferents *behaviours* que implementen la funcionalitat de l'agent.

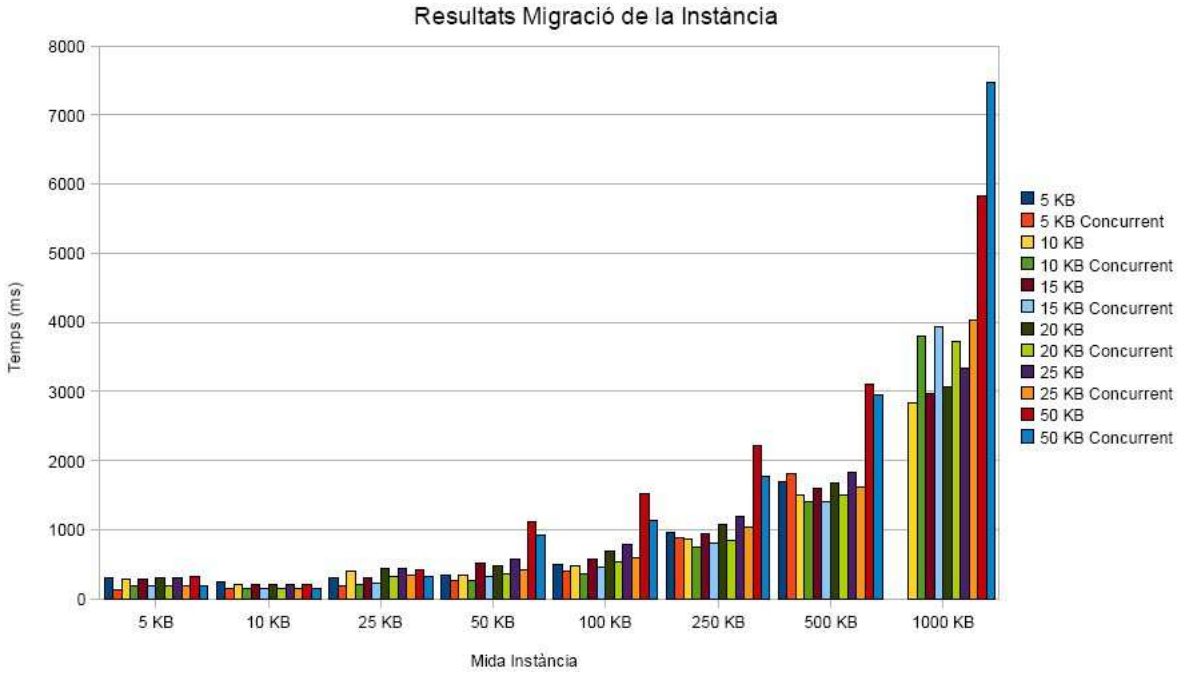

Figura 6.6: Gràfic Resultats Migració de la Instància del protocol FTP

		Fragment Size					
	PCTP	5 KB	10 KB	15 KB	20 KB	25 KB	50 KB
Code 5 KB	126	165	161	161	161	161	161
Code 10 KB	140	189	204	205	206	204	203
Code 25 KB	222	253	290	360	424	551	530
Code 50 KB	493	368	444	600	668	838	1629
Code 100 KB	1734	593	705	1020	1179	1314	2868
Code 250 KB	11943	1289	1506	2658	3439	3789	7632
Code 500 KB	41499	2298	2853	4750	5662	6750	13908
Code 1000 KB	150162	4016	4571	7337	8321	10286	23343

Taula 6.4: Resultats Migració del Codi No Concurrent (ms)

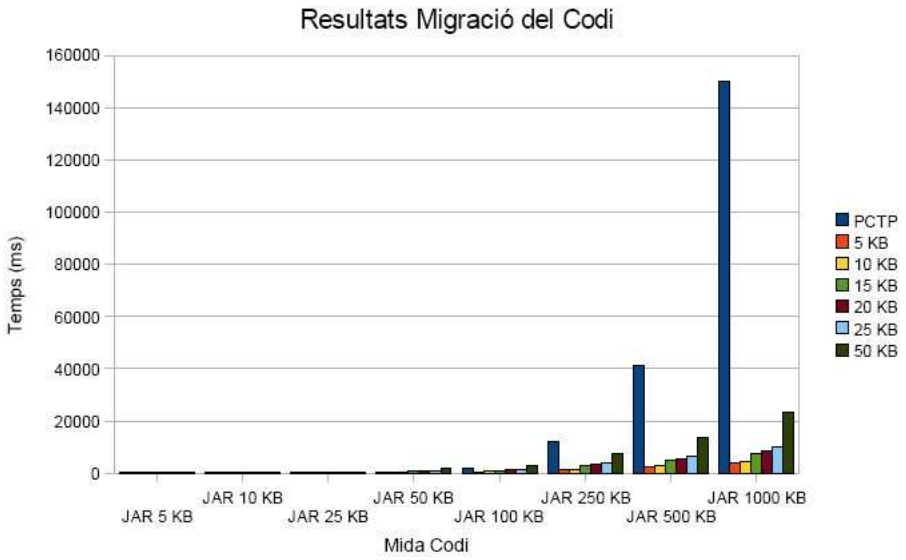

Figura 6.7: Gràfic Resultats Migració del Codi No Concurrent

Resultats Codi No Concurrent

A la Taula 6.4 es poden veure els resultats obtinguts amb els tests de codi no concurrents. En la Figura 6.7 s'observa com amb el protocol PCTP a mesura que creix la mida del codi de l'agent a transferir el temps augmenta de manera exponencial, mentre que amb el protocol FTP el temps creix de forma més moderada.

En la Figura 6.8 es pot veure més en detall com responen ambdós protocols en migracions d'agents petits. S'observa que per a agents amb JARs entre 5 KB i 25 KB el protocol PCTP és lleugerament més ràpid. Com s'ha comentat anteriorment aquest fet es deu a què en el protocol PCTP la instància de l'agent s'envia en un primer missatge amb les condicions de migració, mentre que en el protocol proposat la instància s'envia després d'haver establert les condicions de migració. Tot i que la instància és de mida reduïda (3KB) el fet d'enviar un missatge únicament amb la instància afegit al fet que amb agents petits es generen pocs fragments i per tant s'envien pocs missatges, fa que per a mides de codis petits (5 KB -25 KB) la fragmentació del codi no millori els resultats del protocol actual.

Malgrat això les diferències de temps no són gaire elevades.

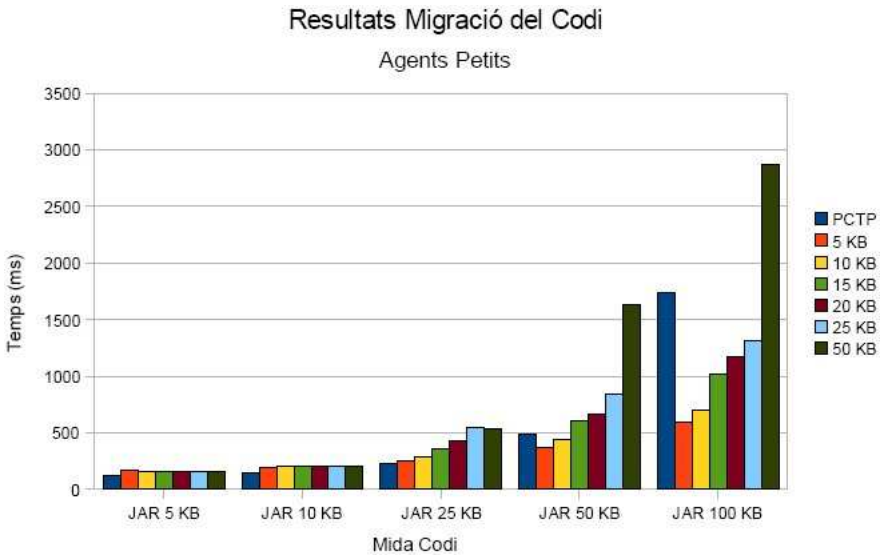

Figura 6.8: Gràfic Resultats Migració del Codi No Concurrent d'Agents Petits

En canvi a partir de codis de 50 KB el protocol FTP presenta millors resultats. Si ens fixem en la Figura 6.7 veiem com a mesura que la grandària dels JARs creix, les diferències de temps entre el protocol PCTP i FTP són cada vegada més importants. En el cas del codi de 1000 KB amb el fragment de 5 KB s'obté una millora del 97%.

En la Figura 6.9 es poden veure els resultats dels tests de codi per les diferents mides de fragment triades. S'han exclòs els resultats del protocol PCTP per veure més clarament els resultats de la resposta del protocol FTP. S'observa com amb el fragment de 50 KB, de la mateixa manera que passava amb els tests d'instància, la progressió dels temps és exponencial. Es pot observar com la mida de fragment òptima és la de 5 KB.

A la Taula 6.5 es poden veure els resultats obtinguts amb els tests de codi no concurrents amb el fragment de 15 KB per als protocols FTP i MIP07.

En la Figura 6.10 es mostra la resposta al test de migració del codi amb frag-

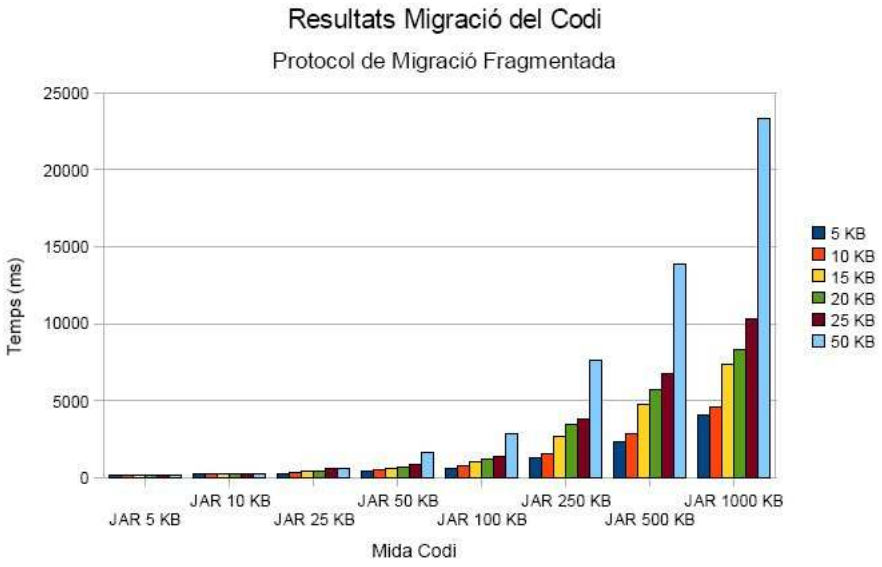

Figura 6.9: Gràfic Resultats Migració del Codi No Concurrent del protocol FTP

	Protocol	
	FTP	MIP07
Code 5 KB	161	148
Code 10 KB	205	191
Code 25 KB	360	357
Code 50 KB	600	699
Code 100 KB	1020	1281
Code 250 KB	2658	2975
Code 500 KB	4750	5886
Code 1000 KB	7337	11439

Taula 6.5: Resultats Migració amb fragment de 15 KB del Codi No Concurrent (ms)

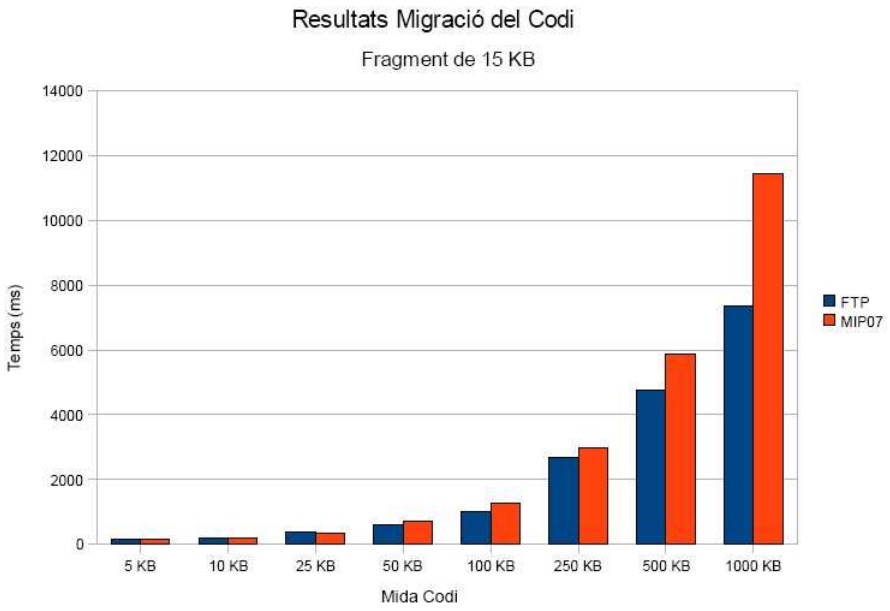

Figura 6.10: Gràfic Resultats Migració del Codi No Concurrent amb el fragment de 15 KB

ment de 15 KB del protocol proposat (FTP) i el protocol presentat en [MIP07]. Pels agents petits entre 5 KB i 25 KB s'observa com el protocol MIP07 presenta lleugers millors resultats que el protocol proposat. Per aquests agents amb mides de codi fins a 25 KB pràcticament no hi ha fragmentació del codi, per tant el codi s'envia en un o dos missatges ACL. El protocol MIP07 transfereix la instància de l'agent en un missatge ACL a l'inici de la migració juntament amb les condicions de migració. En canvi en el protocol FTP la instància s'envia en un missatge ACL específic un cop s'han establert les condicions de migració, per tant aquest missatge extra que s'envia en el protocol FTP vers el protocol MIP07 en el cas concret d'agents fins a 25 KB penalitza el temps de migració total. Si ens fixem amb les respostes de les migracions d'agents superiors a 25 KB, podem veure com a mida que creix la mida de l'agent augmenta la diferència de temps entre els dos protocols. Amb la migració de l'agent d'1MB s'observa la màxima diferència on

el protocol MIP07 tarda fins a un 56% més de temps en transferir l'agent. Això succeeix a causa de la confirmació de fragments que realitza el protocol MIP07, ja que per cada fragment que envia n'espera una confirmació i fins que no la rep no envia el següent fragment. Quan hi ha una gran quantitat de fragments a enviar, la resposta del protocol es veu afectada en gran mesura.

Resultats Codi Concurrent

		Fragment Size					
	PCTP	5 KB	10 KB	15 KB	20 KB	25 KB	50 KB
Code 5 KB	77	112	107	107	106	107	107
Code 10 KB	85	132	150	142	143	142	142
Code 25 KB	140	190	223	283	346	460	414
Code 50 KB	339	297	356	503	572	735	1339
Code 100 KB	1200	448	534	805	932	1033	2426
Code 250 KB	8300	919	906	1372	1705	2098	4532
Code 500 KB	32710	1822	1548	1729	2183	2792	6034
Code 1000 KB	116721	4733	3065	3061	3162	3438	8147

Taula 6.6: Resultats Migració del Codi Concurrent (ms)

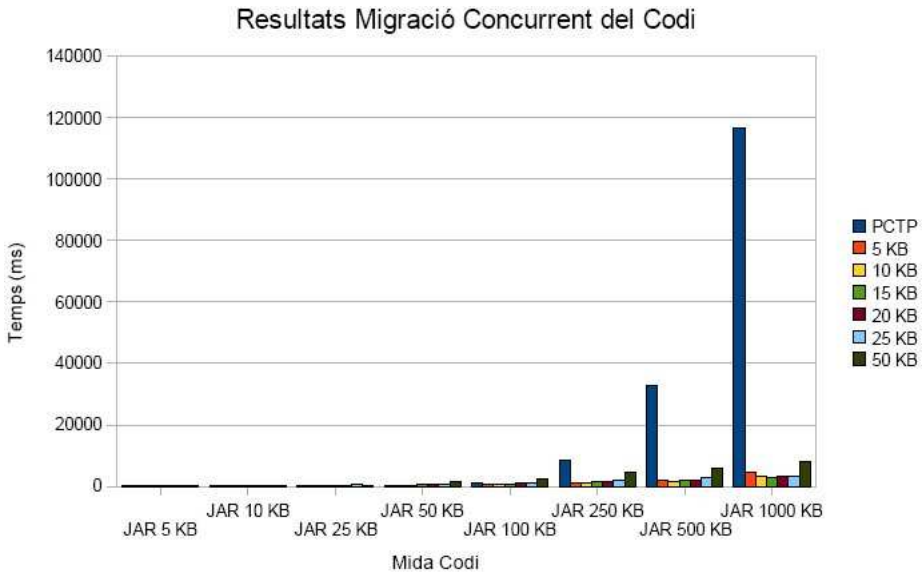

Figura 6.11: Gràfic Resultats Migració del Codi Concurrent

A la Taula 6.6 es poden veure els resultats obtinguts amb els tests de codi concurrents. La Figura 6.11 mostra l'evolució dels temps de migracions simultàncies de deu agents amb JARs entre 5 KB i 1000 KB. Es pot observar com la tendència és la mateixa que la que s'ha vist a la Figura 6.7, és a dir, el protocol PCTP presenta una resposta exponencial a mesura que creix la mida del codi, en canvi el protocol FTP té una resposta més moderada.

En la Figura 6.12 es poden veure els resultats dels tests de codi concurrents per les diferents mides de fragment triades. S'observa com amb el fragment de 50 KB la progressió dels temps és més marcadament exponencial. Pels agents entre 5 KB i 100 KB el fragment de 5 KB dóna millors resultats, en canvi pels agents de més de 100 KB el fragment de 10 KB és el que presenta un millor rendiment. Per tant amb els tests concurrents del codi, igual que en els tests d'instància, no hi ha una mida de fragment òptima, sinó que aquesta va en funció de la mida del JAR de l'agent a transferir.

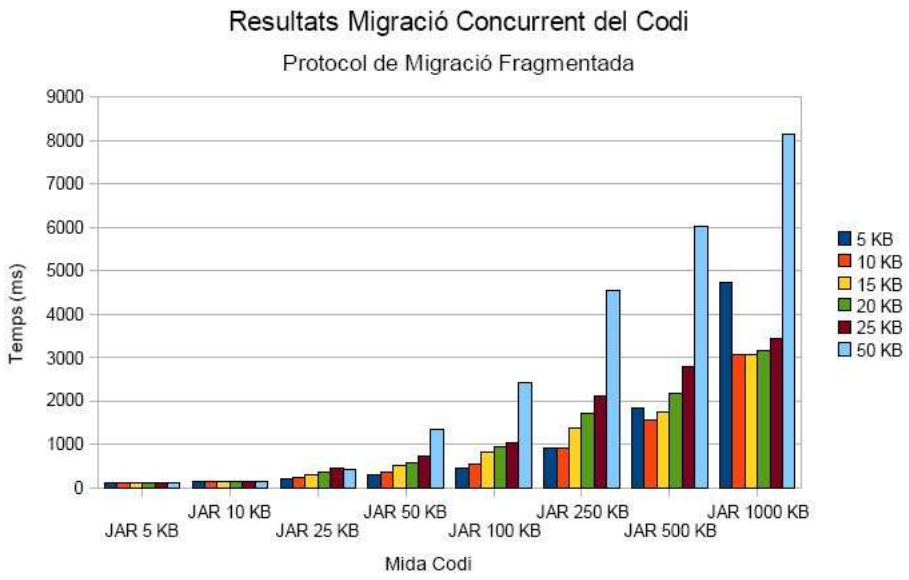

Figura 6.12: Gràfic Resultats Migració del Codi Concurrent del protocol FTP

En la Figura 6.13 es pot veure més en detall les diferències entre els tests de

les migracions concurrents de codi i les migracions de codi amb un únic agent migrant simultàniament. S'observa que les migracions concurrents presenten millors resultats envers la migració del mateix agent en solitari a causa, com ja s'ha comentat en les migracions d'instància, de l'efecte *pipeline* que es produeix en les migracions concurrents. A més a més JADE gestiona els agents amb *threads* [GCJADE], característica que també poden aprofitar els tests concurrents de codi. No obstant això en l'agent de 1000 KB s'observa com la migració concurrent amb el fragment de 5 KB presenta un temps més elevat que la migració homòloga de l'agent de 1000 KB en solitari. La causa d'aquest canvi de tendència ja s'ha comentat en l'apartat dels tests de la instància corresponent i és per la sobrecàrrega de recursos que suposa gestionar la gran quantitat de missatges ACL que es creen per a la transferència dels agents de 1000 KB amb fragments de 5 KB. En aquest cas, però, no es mantenen dues còpies en memòria del codi ja que es manipula directament el fitxer JAR per a la creació dels fragments. Per tant el fragment de 5KB amb el codi de 1000KB marca el llindar a partir del qual és més òptim tractar missatges de mida més gran però en menys quantitat en comptes de tractar missatges més petits però en major quantitat.

A la Taula 6.7 es poden veure els resultats obtinguts amb els tests de codi concurrents amb el fragment de 15 KB per als protocols FTP i MIP07.

	Protocol	
	FTP	MIP07
Code 5 KB	107	97
Code 10 KB	142	130
Code 25 KB	283	267
Code 50 KB	503	557
Code 100 KB	805	992
Code 250 KB	1372	2474
Code 500 KB	1729	5169
Code 1000 KB	3061	10109

Taula 6.7: Resultats Migració amb fragment de 15 KB del Codi Concurrent (ms)

En la Figura 6.14 es mostra la resposta al test de migració concurrent del codi

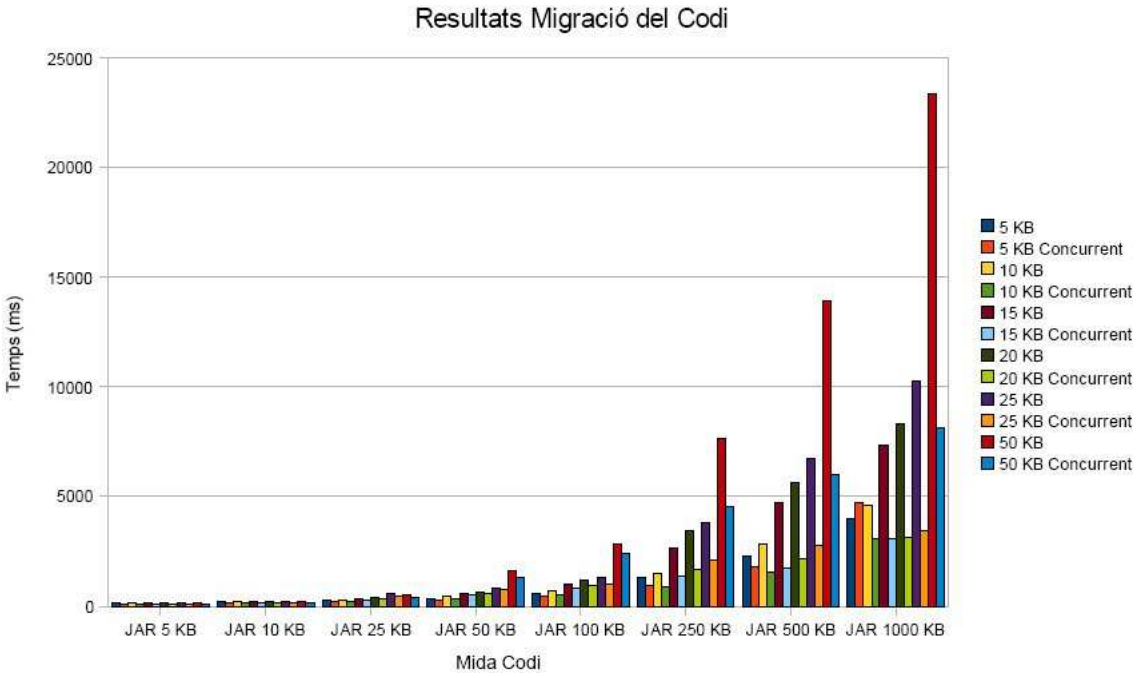

Figura 6.13: Gràfic Resultats Migració del Codi del protocol FTP

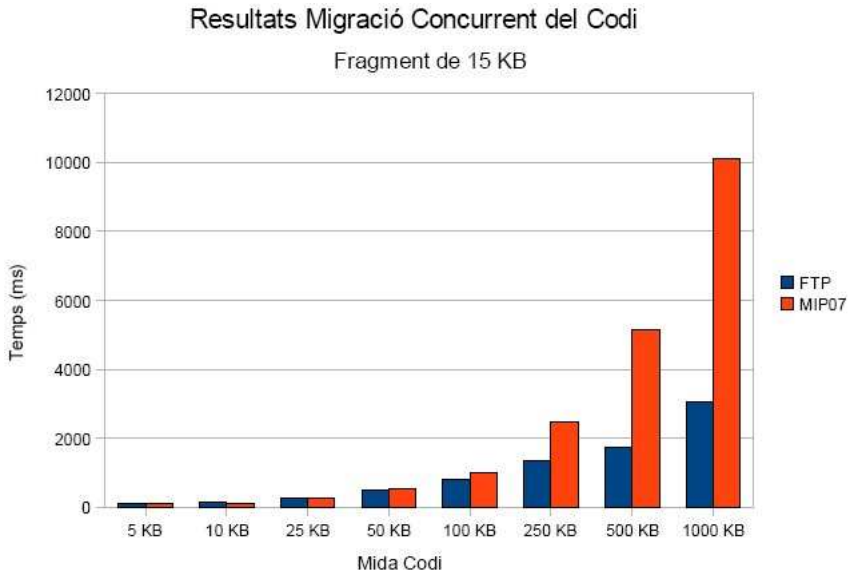

Figura 6.14: Gràfic Resultats Migració del Codi Concurrent amb el fragment de 15 KB

amb fragment de 15 KB del protocol proposat (FTP) i el protocol presentat en [MIP07].

Pels agents petits entre 5 KB i 25 KB s'observa com el protocol MIP07 presenta lleugers millors resultats que el protocol proposat. Per aquests agents amb mides de codi fins a 25 KB pràcticament no hi ha fragmentació del codi, per tant succeeix el mateix que amb els tests de migració no concurrents per al fragment de 15 KB explicats anteriorment. Si ens fixem amb les respostes de les migracions d'agents superiors a 25 KB, podem veure com a mida que creix la mida de l'agent, augmenta la diferència de temps entre els dos protocols. Amb la migració de l'agent d'1MB s'observa la màxima diferència on el protocol MIP07 tarda fins a un 230% més de temps en transferir l'agent. Això succeeix a causa, com hem explicat anteriorment, de la confirmació de fragments que realitza el protocol MIP07, ja que per cada fragment que envia n'espera una confirmació i fins que no la rep no envia el següent fragment. Quan hi ha una gran quantitat de fragments a enviar, la resposta del protocol es veu afectada en gran mesura i en el cas de les migracions concurrents de 10 agents hi ha encara més quantitat de fragments a transferir, la qual cosa fa que les diferències també augmentin. És fàcil adonar-se'n si es fa un cop d'ull als valors absoluts de les migracions d'agents d'1 MB en les Taules 6.5 i 6.7, on el protocol FTP presenta una millora del 60% aproximadament, mentre que la millora de temps en el protocol MIP07 no arriba al 12%.

6.4.3 Resultats Codi amb Latència

Els tests de codi amb latència pretenen mostrar com respon el protocol de migració fragmentada proposat davant de migracions amb les mateixes mides de la part del codi de l'agent que s'han utilitzat per als tests de codi però afegint un retard de 60 ms entre les dues plataformes. Aquests tests també s'han provat amb la proposta de migració proposada en [MIP07] de manera que permetrà veure i comparar el

rendiment d'ambdós protocols en un escenari amb latència que intenta simular el retard de temps que hi hauria entre dues plataformes d'agents mòbils JADE separades geogràficament per una llarga distància.

Convé recordar que el temps que es mesura és el d'un itinerari complet, és a dir, el temps que passa des del punt en què l'agent comença a migrar de la plataforma origen a la plataforma destí fins que l'agent torna de la plataforma destí a la plataforma origen i és reconstruït. Per tant el retard afegit en realitat és de 120 ms (60 ms d'anada i 60 ms de tornada).

Els tests s'han realitzat per ambdós protocols únicament amb el fragment de 15 KB, ja que per poder-los comparar no cal tornar a fer els tests per a totes les mides de fragments.

Resultats Codi amb Latència No Concurrent

	Protocol	
	FTP	MIP07
Code 5 KB	2298	2078
Code 10 KB	2565	2361
Code 25 KB	3372	3409
Code 50 KB	4824	5318
Code 100 KB	7550	8567
Code 250 KB	15903	18893
Code 500 KB	30061	36215
Code 1000 KB	58032	70333

Taula 6.8: Resultats Migració amb fragment de 15 KB del Codi No Concurrent i amb Latència (ms)

A la Taula 6.8 es poden veure els resultats obtinguts amb els tests de codi no concurrents amb una latència afegida de 60 ms. En la Figura 6.15 es mostra la resposta al test de migració del codi amb fragment de 15 KB del protocol proposat (FTP) i el protocol presentat en [MIP07] amb una latència de 60 ms entre les plataformes d'origen i destí. Pels agents petits entre 5 KB i 10 KB s'observa com el protocol MIP07 presenta millors resultats que el protocol proposat. Per

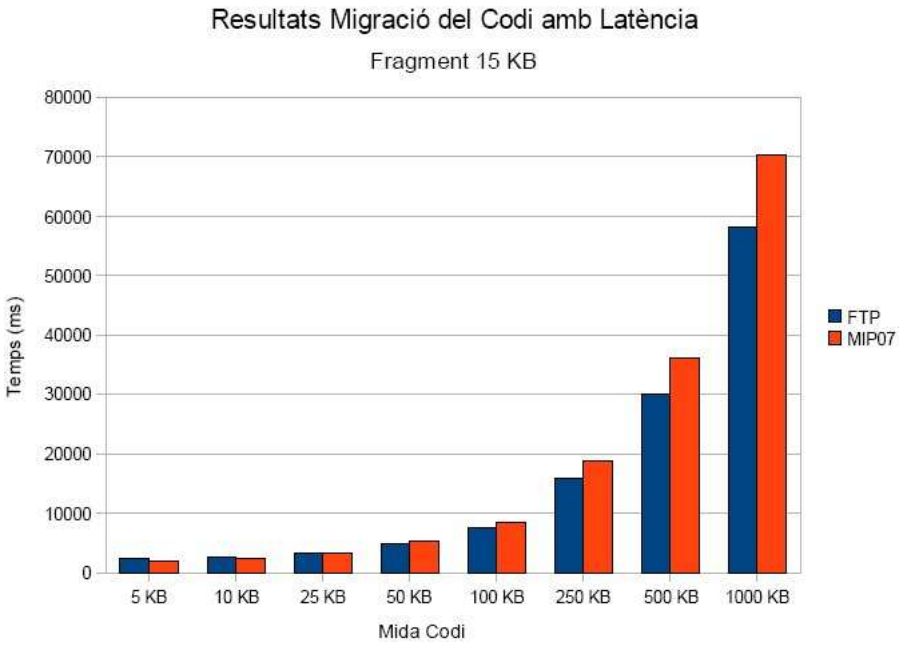

Figura 6.15: Gràfic Resultats Migració del Codi No Concurrent amb Latència de 60 ms

aquests dos agents amb mides de codi inferior a 15 KB no hi ha fragmentació del codi, per tant el codi s'envia en un sol missatge ACL. El protocol MIP07 transfereix la instància de l'agent de la mateixa manera que ho fa el protocol PCTP explicat en l'apartat 6.4.1, és a dir, que la instància s'envia en un missatge ACL a l'inici de la migració juntament amb les condicions de migració. En canvi en el protocol FTP la instància s'envia en un missatge ACL específic un cop s'han establert les condicions de migració, per tant aquest missatge extra que s'envia en el protocol FTP vers el protocol MIP07 en el cas concret d'agents més petits de 15 KB penalitza el temps de migració total i a més a més la latència afegida de 60 ms fa que aquesta penalització de temps es vegi ampliada.

No obstant això, la migració d'agents amb JARs de 25 KB amb el protocol FTP ja presenta una lleugera millora respecte el protocol MIP07. Aquest millora en el rendiment s'observa com va augmentant a mesura que la mida del codi creix.

Es pot observar com la migració de l'agent amb el codi de 1000 KB el protocol MIP07 tarda fins a un 21% de temps més que el protocol FTP. Aquest canvi de tendència per a agents més grans és perquè en el protocol MIP07 per cada fragment de codi que s'envia s'espera rebre un missatge ACL de confirmació de la plataforma destí, conforme s'ha rebut el fragment correctament, abans d'enviar el següent fragment. La latència afegida al test fa que la transmissió de l'agent sigui més lenta.

En canvi en el protocol FTP no es realitza una confirmació explícita dels fragments, sinó que s'envien tots els fragments de codi seguits un darrera l'altre i si hi ha algun fragment que no arriba a la plataforma destí aquesta és la que s'encarrega de sol·licitar-lo. A tenor dels resultats es pot apreciar com aquesta decisió de disseny beneficia el temps de migració per a agents de mida mitjana i gran.

Resultats Codi amb Latència Concurrent

	Protocol	
	FTP	MIP07
Code 5 KB	1742	1778
Code 10 KB	1958	1942
Code 25 KB	2806	2800
Code 50 KB	4310	4273
Code 100 KB	7132	7035
Code 250 KB	15499	15522
Code 500 KB	29538	29750
Code 1000 KB	57600	57686

Taula 6.9: Resultats Migració amb fragment de 15 KB del Codi Concurrent i amb Latència (ms)

A la Taula 6.9 es poden veure els resultats obtinguts amb els tests de codi concurrents amb una latència afegida de 60 ms. En la Figura 6.16 es mostra la resposta al test de migració del codi amb fragment de 15 KB del protocol proposat (FTP) i el protocol presentat en [MIP07] amb una latència de 60 ms entre les plataformes d'origen i destí. S'observa com els dos protocols tenen respostes

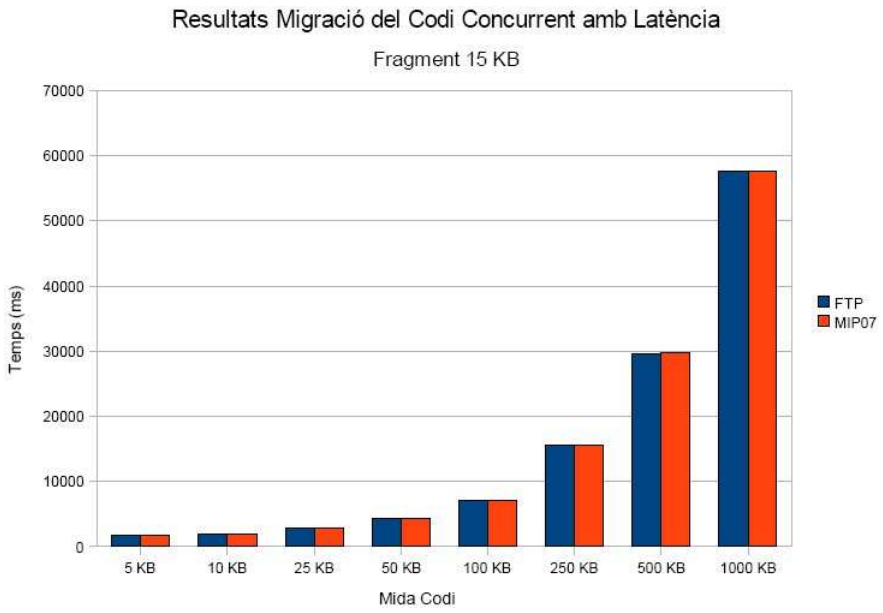

Figura 6.16: Gràfic Resultats Migració del Codi Concurrent amb Latència de 60 ms

semblants per a totes les migracions d'agents mòbils. Si ens parem atenció en els valors absoluts i els comparem amb els resultats no concurrents de la Taula 6.8 ens n'adonem que els temps del protocol FTP pràcticament no han variat, mentre que la resposta del protocol MIP07 en els tests concurrents ha millorat notablement especialment en la transferència d'agents grans.

D'aquestes dades es desprèn que els resultats no concurrents per al protocol FTP amb latència ja eren força bons i les migracions concurrents no milloren gaire aquells resultats. En canvi, amb el protocol MIP07 l'espera provocada per l'efecte *pipeline* i la latència en les migracions concurrents no és inactiva com en el protocol FTP, sinó que s'aprofita per iniciar altres processos que permeten millorar els temps obtinguts en les migracions dels agents en solitari.

6.5 Resum

En aquest capítol s'han explicat els tests que s'han dut a terme per al protocol de migració fragmentada proposat. Primer s'ha descrit l'entorn de proves on s'han executat els tests. Tot seguit s'ha detallat el disseny dels tests que s'han realitzat emfatitzant aquelles característiques que es volen avaluar com la concurrència, les mides de fragment, les diferents mides d'agents tant de la part de les dades (instància) com de la part del codi i diferents escenaris.

Finalment s'han comentat i raonat els resultats obtinguts en base als tests dissenyats i s'han comparat aquests resultats amb altres protocols de migració existents. A partir d'aquests resultats s'han pogut extreure conclusions que s'explicaran en el següent capítol.

Capítol 7

Conclusions

En aquest capítol es resumirà la feina feta i s'enumeraran els objectius assolits. Posteriorment s'explicaran les conclusions a les què s'ha arribat en base als resultats obtinguts i es presentaran una sèrie de línies de treball futures. Finalment es farà una valoració personal del projecte.

En aquest projecte s'ha dissenyat i desenvolupat un protocol de migració per a agents mòbils emmarcat en l'arquitectura multiprotocol definida per l'IPMA i integrat dins de l'*add-on* de mobilitat JIPMS de la plataforma JADE. Amb aquest projecte, doncs, s'ofereix un nou mecanisme de transferència per a agent mòbils flexible i personalitzable, que permet utilitzar-lo en diferents escenaris i situacions on hi participen agents.

La planificació que s'havia fet inicialment en el capítol d'anàlisi no s'ha complert en tots els casos amb el temps previst. La fase de codificació i tests ha necessitat més temps del planificat en un principi a causa d'alguns problemes apareguts en els test concurrents de rendiment del protocol que han permès descobrir errors en la implementació del protocol de migració fragmentada FTP que no s'havien detectat amb els tests de funcionament. Aquest fet ha obligat a recodificar certes parts del protocol per tal de resoldre problemes trobats en migracions simultànies

d'agents mòbils. D'altra banda, gràcies als tests de rendiment s'ha pogut desenvolupar un protocol de migració més robust i fiable, que únicament amb els tests de funcionament no s'hagués pogut fer. S'ha dissenyat i realitzat un conjunt més ampli i complet de tests que no només han permès avaluar el rendiment del protocol FTP amb més profunditat, sinó que a més a més han permès poder-lo comparar amb altres protocols existents en el projecte JIPMS en diferents escenaris.

7.1 Assoliment d'Objectius

En aquest apartat s'enumeraran els objectius assolits relacionant-los amb els objectius inicialment plantejats en el capítol d'introducció.

1. S'ha realitzat un estudi exhaustiu dels protocols de migració implementats dins del mòdul JIPMS. Especialment del protocol PCTP que és el protocol de migració que s'utilitza per defecte en la versió de desenvolupament del projecte JIPMS. També s'ha estudiat el protocol presentat en [MIP07] per tal d'analitzar-ne les mancances i poder encarar la fase d'anàlisi amb solidesa.

2. Un cop acabada la fase de documentació i una vegada recopilada tota la informació necessària sobre el què es volia fer, s'ha realitzat un anàlisi dels requisits funcionals i no funcionals que ha de tenir el protocol de migració que s'ha dissenyat.

3. S'ha proposat un disseny i una implementació d'un protocol de migració fragmentada (FTP) per a agents mòbils que respecta els estàndards definits per FIPA. L'agent és capaç de parar la seva execució, migrar d'una plataforma a una altra dividit en petits fragments i un cop arribat a la plataforma destí reconstruir-se i continuar la seva execució.

4. S'ha desenvolupat un protocol de migració flexible. El protocol permet configurar les condicions de migració pel propi agent en cada migració, tals com la mida dels fragments amb què es vol migrar l'agent, oferint fins i tot la possibilitat de fer una migració sense fragmentar. A més a més s'ha dotat al protocol d'un sistema de recuperació de fragments que fa viable la seva execució en escenaris que no ofereixen fiabilitat.

5. S'ha dissenyat un joc de proves que ha permès avaluar el funcionament i rendiment del protocol implementat. A més a més s'han extès aquests tests a altres protocols del servei de migració JIPMS, fet que ha permès poder comparar el rendiment de diversos protocols en diferents escenaris.

6. S'ha implementat el protocol proposat al projecte [JIPMS]. A més a més, també s'han integrat el conjunt de tests dissenyats per al protocol FTP a la *suite* de tests de rendiment del projecte [JIPMS]. Aquesta contribució també permetrà que futurs protocols que es desenvolupin dins del projecte [JIPMS] puguin utilitzar els tests dissenyats per tal d'avaluar-ne el rendiment.

7.2 Conclusions dels Resultats Obtinguts

En aquest apartat es descriuran les conclusions a les quals s'ha arribat després d'analitzar els resultats obtinguts en els tests de rendiment explicats en el capítol anterior.

La primera conclusió a la que s'ha arribat és que la migració fragmentada en diversos missatges de la instància i del codi d'un agent mòbil resulta més òptima que migrar tot l'agent en un sol missatge per a agents grans. En [MIP07] es va poder comprovar que l'enviament fragmentat del codi era més eficient i es va suggerir que l'enviament de la instància fragmentada possiblement també milloraria

els temps de migració. En el present projecte s'ha demostrat que aquesta hipòtesis presentada en [MIP07] era encertada.

La segona conclusió és que el protocol FTP dissenyat millora els resultats de migració dels protocols existents (PCTP i [MIP07]) per a agents grans. Per tant, la decisió de disseny de no confirmar la recepció de fragments va ser adequada.

Una altra conclusió que s'extreu dels resultats és que no existeix a priori una única mida de fragment òptima que garanteixi el mínim temps de migració, sinó que aquesta és variable i ve determinada per les condicions de migració. Si bé algunes d'aquestes condicions les podem conèixer prèviament, com per exemple la mida de l'agent mòbil a transferir, d'altres s'escapen al control del programador, com per exemple el nombre de migracions simultànies que s'estan duent a terme en una plataforma en un moment donat. Tot i això, si deixem de banda les condicions de migració que no es poden preveure i ens centrem en els resultats de migracions no concurrents d'instància i codi presentades en el capítol anterior, es pot concloure que fins i tot coneixent les condicions de migració a priori tampoc és possible determinar una mida de fragment que garanteixi la migració en el mínim temps possible.

Una conclusió que s'ha extret de l'anàlisi dels resultats és que tot i dividir l'agent en missatges més petits i millorar notablement els resultats en migracions per agents grans, el coll d'ampolla en les migracions creiem que està en el tractament dels missatges ACL que implementa JADE i en l'interpretació d'aquests missatges que es reben en base a les ontologies definides. Per cada missatge rebut JADE comprova l'ontologia associada i en *parseja* el contingut i posteriorment assigna un sentit semàntic a cada part segons l'ontologia associada. Aquest procés s'ha comprovat que resulta molt més lent i tediós realitzar-lo per a missatges grans que per a missatges petits. S'ha volgut deixar constància d'aquest fet, perquè es creu que pot ser útil i cal tenir-lo en compte de cara a la millora de la plataforma JADE.

Com a darrera conclusió deixar constància d'un fet que ja s'ha comentat en el capítol anterior. El protocol de migració fragmentada proposat (FTP) presenta temps de migració lleugerament superior que el protocol PCTP per a agents petits. La causa d'això és que en agents petits on hi ha poca o nul·la fragmentació, el fet d'enviar un missatge petit extra per transferir la instància suposa un percentatge elevat respecte el total de missatges enviats. Si bé les diferències de temps per a agents petits no són gaire importants comparades amb les diferències de temps en migracions d'agents grans, pensem que aquesta deficiència es podria solucionar fàcilment i ho comentarem com a via d'ampliació en el següent apartat.

7.3 Línies de Millora i Treball Futur

En aquest apartat comentarem aquelles línies en les que es podria aprofundir per tal de millorar el protocol de migració presentat en aquest projecte.

- Incloure la instància de l'agent en el primer missatge (REQUEST) del procés de migració. Aquesta inclusió, però, només estaria justificada per les migracions d'agents petits. Per tant, a partir dels resultats exposats en el capítol anterior, caldria fer un estudi exhaustiu dels temps de migració d'agents amb mides d'instància entre 25 KB i 50 KB. Seria necessària la construcció d'un joc de proves que permetés establir un llindar que definís a partir de quina mida d'instància el protocol FTP enviaria la instància a priori i a partir de quina mida d'instància enviaria tot l'agent fragmentat.

- Una limitació que presenta la implementació del protocol FTP proposat és que només permet fer migracions d'agents mòbils que es troben ubicats en el contenidor principal de la plataforma JADE. Si bé aquest fet no presenta un greu problema, ja que totes les migracions es gestionen des del contenidor principal i en última instància es duen a terme en aquest, sí que seria

interessant oferir la possibilitat de realitzar migracions entre diferents plataformes independentment del contenidor on es trobés situat l'agent mòbil.

Com a treball futur es proposa la creació d'un nou protocol de l'etapa de *post-Transfer* de l'arquitectura IPMA que complementi el protocol de migració fragmentada proposat. Aquest protocol s'executaria un cop l'agent s'hagués transferit completament a la plataforma de destí i serviria per transmetre volums de dades grans. Si un agent ha de recopilar una informació i aquesta acaba creixent a una mida considerable, la idea és que la informació no formi part de la instància de l'agent sinó que vagi associada a aquest i es transfereixi amb un protocol *mot-xiller*. Després del què s'ha apuntat en la darrera conclusió, pensem que aquest protocol de migració hauria d'aprofitar algun dels protocols basats en HTTP i evitar la utilització de missatges ACL per tal d'aprofitar al màxim la velocitat que ofereix la xarxa. Aquest protocol podria ser utilitzat per exemple en aplicacions com les descrites en [Med07] i [Med08] per enviar proves mèdiques associades a un pacient (representat per un agent mòbil) com poden ser vídeos d'ecografies, ecocardiogrames, imatges d'alta resolució de ressonàncies magnètiques o de Tomografia Axial Computaritzada (TAC)s, etc.

7.4 Valoració Personal del Projecte

A nivell personal aquest projecte m'ha ajudat a formar-me com a futur enginyer, permetent-me desenvolupar un projecte de principi a fi. Amb aquest projecte he pogut satisfer la inquietud que em despartava el paradigma dels agents mòbils i m'ha permès aprofundir en els meus coneixements sobre aquesta tecnologia des de l'interior, dissenyant un protocol de migració d'agents mòbils que m'ha permès veure aquest món a baix nivell.

Gràcies a aquest projecte he pogut formar part d'un projecte de codi obert

[JIPMS] i desenvolupar-lo en un entorn de treball molt enriquidor, ja que el grup de recerca SeNDA fa un seguiment acurat de tots els projectes fent reunions periòdiques on tots els projectistes presentem els nostres dubtes i podem discutir i proposar solucions.

El fet que el resultat d'aquest projecte estigui inclòs dins dels projecte JIPMS i que, per tant, pugui ser utilitzat per altres persones en futurs projectes per desenvolupar noves aplicacions per agents mòbils, suposa una motiu de satisfacció afegit.

Finalment comentar que els resultats d'aquest treball contribuiran en l'elaboració d'un article d'investigació d'àmbit internacional que s'està preparant conjuntament amb membres del grup de recerca SeNDA del dEIC.

Bibliografia

[AIPUML] James J. Odell, H. Van Dyke Parunak i Bernhard Bauer. Representing Agent Interaction Protocols in UML. Agent-Oriented Software Engineering: First International Workshop, Volum 1957, AOSE 2000, Limerick, Ireland. 2001. Pag: 201-218.

[AgentUML] Bernhard Bauer, Jörg P. Müller i James Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, Limerick, Ireland. 2001. Pag: 109-120.

[CMJC06] J. Cucurull. Contribució a la mobilitat i seguretat dels agents software, maig 2006. Tesina del programa de doctorat en informàtica, opció Combinatòria i Comunicació Digital.

[FIPAIP] FIPA Interaction Protocols Specifications, març 2008. <http://www.fipa.org/repository/ips.php3>

[FIPA-IPMA] J. Cucurull, R. Martí, S. Robles, J. Borrell i G. Navarro. FIPA-based Interoperable Agent Mobility. Multi-Agent Systems and Applications V, Volum 4696, Leipzig, Alemania. Setembre 2007. Pag: 319-321.

[FIPARIP] FIPA Request Interaction Protocol Specification, març 2008. <http://www.fipa.org/specs/fipa00026/SC00026H.html>

[FIPASL] FIPA Semantic Language Specification, abril 2008. <http://www.fipa.org/specs/fipa00008/SC00008I.html>

[JIPMS] JADE Inter-Platform Mobility Service. Source Project, juny 2008. <http://sourceforge.net/projects/jipms>

[GCJADE] G. Caire. JADE Tutorial - JADE Programming for Beginners, setembre 2007.

[JADEWEB] Java Agent DEvelopment Frameword, maig 2007. <http://jade.tilab.com>

[JADEBOOK] Fabio L. Bellifemine, Giovanni Caire i Dominic Greenwood. Developing Multi-Agent Systems with JADE. febrer 2007.

[JMT04] J. Cucurull. JADE MTP-TFTP: Desenvolupament d'un mòdul de comunicacions per a la plataforma d'agents Jade basat en el protocol de xarxa TFTP, juny 2004. Projecte Fi de Carrera corresponent als estudis d'Enginyeria Superior en Informàtica.

[IPMA] J. Cucurull, B. J. Overeinder, M. A. Oey, J. Borrell i F. M. T. Brazier. Abstract software migration architecture towards agent middleware interoperability. Proceedings of the International Multiconference on Computer Science and Information Technology, Wisla, Polònia. octubre 2007. Pag: 27-37.

[Med07] A. Martin. MedIGs, un sistema segur basat en agents per al descobriment i obtenció de dades mèdiques distribuïdes. Estudi i desenvo-

lupament, juny 2007. Projecte Fi de Carrera corresponent als estudis d'Enginyeria Superior en Informàtica a la UAB.

[Med08] D. Peña. Aplicació dels agents mòbils en els sistemes d'informació mèdica. Desenvolupament d'un sistema per a la obtenció segura de dades distribuïdes, juny 2008. Projecte Fi de Carrera corresponent als estudis d'Enginyeria Superior en Informàtica a la UAB.

[MedIGS] Vieira-Marques, P.M.; Robles, S.; Cucurull, J.; Cruz-Correia, R.J.; Navarro,G.; Marti, R.; "Secure Integration of Distributed Medical Data Using Mobile Agents", Intelligent Systems, IEEE, Volum 21, Issue 6, Nov.-Des. 2006. Pag: 47-54

[MFBBR07] Book (mfbbr07) Meier, J.; Farre, C.; Bansode, P.; Barber, S. and Rea, D. Performance Testing Guidance for Web Applications Microsoft Press, 2007

[MIP07] V. Sales. Mobilitat Inter-Plataforma d'Agents Software: Protocol de migració fragmentada, juny 2007. Projecte Fi de Carrera corresponent als estudis d'Enginyeria Superior en Informàtica a la UAB.

Firmat: Gerard Suades Méndez

Bellaterra, setembre de 2008

Resum

En aquest projecte s'ha dissenyat un protocol de migració d'agents mòbils per a l'arquitectura IPMA basat en l'enviament dels agents fragmentats en diversos missatges FIPA ACL. Aquest s'ha implementat dins el servei de migració JIPMS per a la plataforma JADE. Finalment s'ha dut a terme un conjunt exhaustiu de tests per avaluar-ne el rendiment i comparar-lo amb altres protocols de migració existents.

Resumen

En este proyecto se ha diseñado un protocolo de migración de agentes móviles para la arquitectura IPMA basado en el envío de los agentes fragmentados en varios mensajes FIPA ACL. Éste se ha implementado dentro del servicio de migración JIPMS para la plataforma JADE. Finalmente se han llevado a cabo un conjunto exhaustivo de tests para evaluar su rendimiento y compararlo con otros protocolos de migración existentes.

Abstract

In this project a mobile agents migration protocol for the IPMA architecture based on the transference of agents fragmented in several FIPA ACL messages has been designed. This protocol has been implemented in the migration service JIPMS for the JADE platform. Finally a comprehensive set of tests has been carried out in order to evaluate the migration performance and compare it with other existing migration protocols.