XARXES TOLERANTS A ENDARRERIMENTS I INTERRUPCIONS. DESENVOLUPAMENT D'UN SCHEDULER D'AGENTS MÒBILS.

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per David Sanchez Garcia i dirigit per Sergi Robles Martínez.

Bellaterra, 17 de juny de 2010

El firmant, Sergi Robles Martínez , 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 David Sanchez Garcia

Bellaterra, Juny de 2010

Firmat: Sergi Robles Martínez

A tots aquells que sou importants per mi.

Agraïments

En primer lloc voldria a agrair el suport i recolzament dels meus directors, Sergi Robles i Carlos Borrego. Gràcies per buscar sempre un lloc per mi en aquesta agenda tan ajustada. Gràcies també als membres del SeNDA per escoltar-me i donar-me consells.
Donar les gràcies a l'Adri per estar sempre al meu costat, en la realització del projecte i en la meva vida. Ja saps que ets molt important per mi.
Gràcies a la meva família per recolzar-me i ajudar-me en tot el que heu pogut.
També voldria donar les gràcies al Toni per animar-me en els moments difícils.
Finalment, no voldria oblidar-me de tots aquells que m'heu ajudat a arribar fins aquí i que estic segur m'ajudareu a seguir endavant. Gràcies de tot cor.

Índex

Índex de figures

Capítol 1

Introducció

La majoria de les xarxes de comunicacions que es fan servir avui en dia segueixen uns estàndards que es van pensar assumint que els diferents punts que les formen sempre estan interconnectats entre ells. Ja sigui fent servir connexions amb cables o fent servir tecnologies més actuals, com les connexions *wireless* o via satèl·lit, sempre se suposa que un punt dins de la xarxa és accessible per un altre punt. Quan la connexió no és possible, simplement s'assumeix que aquest punt ha deixat de funcionar.

Aquest model és aplicable a molts casos, però quan ens trobem en situacions en que aquestes assumpcions no són correctes, aquestes xarxes deixen de funcionar correctament. És per això que apareix un nou tipus de xarxes, anomenades xarxes DTN [2], que tenen la capacitat d'adaptar-se a aquestes situacions. Actualment, el seu ús en aplicacions reals és força escàs, doncs la seva naturalesa fa que de moment només es facin servir en casos molt específics. A més, encara encara s'estan estudiant i desenvolupant diferents formes d'aprofitar-les i implementar-les. Aquests motius fan que encara no estiguin gaire esteses.

D'altra banda, tot i que s'han plantejat convencions en diferents aspectes, encara hi ha problemes que no tenen una única interpretació. D'entre aquests podríem destacar el fet que només tenim una única manera de decidir quina ruta seguirà la informació des del punt de partida fins el seu destí o el fet que no podem aplicar priorització a la informació que ha de saltar d'un node a un altre.

Una forma d'implementar-les és fer servir agents mòbils per a transportar la informació a través de la xarxa. Els agents mòbils són programes amb la capacitat de "viure" i executar-se en un punt de la xarxa [6]. A més, també tenen la capacitat de moure's d'un punt a un altre i seguir executant-se allà. Per fer-ho, els diferents nodes necessiten tenir un entorn on els agents puguin viure. Aquest entorn s'anomena plataforma i encara que hi ha més d'un tipus de plata-

forma, la més estesa actualment és la plataforma anomenada JADE (*Java Agent DEvelopment Framework*) [5].

Una bona idea seria la d'aprofitar els avantatges dels agents mòbils per aconseguir millorar la manera en que la informació viatja a través de les xarxes DTN.

1.1 Objectius

Com a objectiu del nostre projecte volem crear xarxes DTN fent servir agents mòbils que transportin informació a través d'aquestes. Volem dissenyar i implementar millores per a que els agents mòbils puguin viatjar-hi de forma ràpida i efectiva.

Volem trobar una manera de permetre tenir informació amb prioritats, de manera que puguem saber quina informació s'haurà de transmetre abans. En una DTN és molt probable que un node no "vegi" un altre node durant molt de temps, per tant hem d'aprofitar al màxim els moments en que sí es veuen. La idea es donar més prioritat a les dades que puguin aprofitar millor el node que veiem en aquell moment.

També volem aconseguir que la informació transmesa viatgi per uns camins o per uns altres, depenent de les necessitats que tinguin les aplicacions. Podem trobar-nos, per exemple, en el cas d'una aplicació que necessiti informació de diferents punts de la xarxa abans d'arribar al seu destí. En aquest cas, no ens interessa només trobar el camí més ràpid per arribar al nostre destí, també ens interessa que sigui el més ràpid passant per aquests altres punts de la xarxa. Una altra aplicació seria la de trobar un camí que estigués menys congestionat i que, en conseqüència, fos més ràpid. Per tant, volem trobar una forma de que la informació pugui decidir per on vol viatjar en funció dels objectius de la seva aplicació.

1.1.1 Objectius concrets

Les fites concretes que ens hem marcat al nostre projecte són les següents:

1. Entendre què són i com funcionen les xarxes DTN.
2. Estudiar el funcionament de JADE. Concretament entendre l'Add-on IPMS que permet als agents migrar entre plataformes.
3. Dissenyar una gestió de cues que permeti als agents fer servir informació d'encaminament per decidir el camí a seguir fins al seu destí.

4. Dissenyar una manera senzilla de permetre als agents tenir prioritats i tenir-les en compte durant l'encaminament.

5. Implementar els dos dissenys i validar la implementació en un model d'escenari real.

1.2 Estructura de la memòria

A continuació veurem en quins apartats es divideix la resta de la memòria, així com el resum dels seus continguts.

En el capítol 2 explicarem més detalladament què són i com funcionen les xarxes DTN. També introduirem la plataforma JADE, els agents mòbils i el Add-on de JADE que permet migrar als agents, l'IPMS. Seguidament, en el capítol 3, analitzarem quins són els requisits del projecte i quines necessitats se'ns plantegen. En el capítol 4 veurem el disseny realitzat per a satisfer aquests requisits. Dins d'aquest capítol discutirem perquè s'han triat unes solucions i no unes altres. Després veurem quina ha estat la implementació de la nostra solució En el capítol 5 explicarem quines proves s'han realitzat per tal de validar la implementació i exposarem els resultats obtinguts. Finalment, al capítol 6 veurem les conclusions que hem extret després de realitzar el projecte i parlarem sobre possibles millores i treballs futurs.

Capítol 2

Les xarxes DTN i agents mòbils

En aquest capítol exposarem els mecanismes actuals que permeten implementar xarxes DTN, així com els avantatges i els inconvenients que presenten. Abans però, entendrem quina és la situació actual i quines són les situacions que ens porten a haver de fer servir aquest nou model de xarxa.

Un cop posats en situació, parlarem sobre els agents mòbils i la plataforma JADE, els quals ens proporcionaran mecanismes per assolir els objectius del nostre projecte.

2.1 El model TCP/IP i Internet

Avui en dia el model més estès en l'interconnexió en xarxes és el model TCP/IP [1]. Aquest model és el que es fa servir per interconnectar les diferents xarxes i sub-xarxes que formen Internet. Concretament, TCP/IP és un conjunt de protocols que s'encarreguen d'encaminar la informació que viatja a través de les xarxes, així com d'assegurar-se que aquesta informació arribarà correctament al seu destí. Aquests protocols estan pensats per a entorns com els que trobem actualment, on la connectivitat recau principalment en connexions per cable, com la xarxa telefònica, o sense cables, com ara les connexions amb telèfons mòbils o via satèl·lit. Aquestes connexions estan sempre connectades extrem-a-extrem amb un cert retard entre la font i el destí, tenen un tant per cent d'errors baix i una relació bidireccional de transmissió de dades relativament simètrica (Figura 2.1). És a dir, que quan parlem de xarxes que fan servir TCP/IP, parlem de xarxes on sempre hi ha connexió entre dos punts i on un tall o una desconnexió llarga significa que ja no podem transmetre informació.

Actualment, el model que proposa TCP/IP segueix funcionant. Com hem dit abans Internet es basa en aquest model, però TCP/IP no s'adapta a les noves necessitats. Ja s'està pensant en aplicacions on el més important no és transmetre la informació d'extrem a extrem correctament,

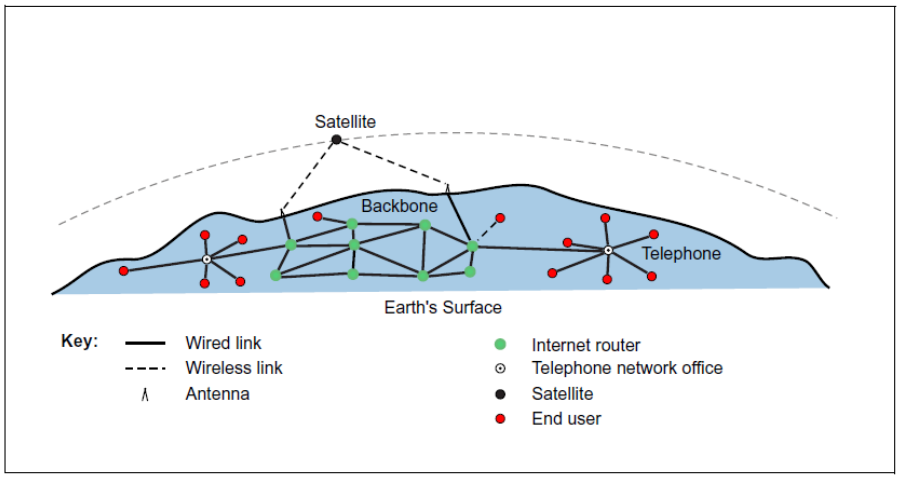

Figura 2.1: Xarxes TCP/IP. Model actual d'Internet.

si no que la informació arribi eventualment al seu destí. És en aquest entorn que neix un nou model de xarxes, les DTN (Delay-Tolerant Networking).

2.2 Delay-Tolerant Networking (DTN)

A partir dels anys 70 i 80, als inicis d'Internet, es va pensar en crear una xarxa Interplanetària anomenada IPN (InterPlaNetary Internet) (Figura 2.2). Aquesta xarxa hauria de permetre la transmissió d'informació entre els planetes que formen el nostre sistema solar, fent possible la comunicació entre els diferents planetes. Tot i que aquesta idea encara es lluny de poder-se aplicar a la vida real, ens planteja un problema important. Si intentem fer servir el protocol actual, el TCP/IP, per intentar crear aquesta comunicació, ens adonarem que ens serà impossible fer-l'ho servir.

En el moment en que s'envia informació d'un punt a un altre, fent servir TCP/IP, l'emissor activa un comptador (*timeout*). Si quan el comptador arriba a zero no s'ha rebut una confirmació de que les dades han arribat correctament, se suposa que hi ha hagut un problema en algun punt de la connexió i es tornen a enviar les dades. Aquest procés es repeteix un número concret de vegades i si finalment no es rep una confirmació, la comunicació es dóna per tallada i es tanca la connexió. En l'escenari de la IPN aquest problema podria donar-se encara que no hi hagués problemes amb la connexió. El temps que pot trigar a arribar la informació que viatja d'un planeta a un altre pot ser molt gran i per tant, és molt probable que el *timeout* s'acabés quan la informació encara estigués viatjant, donant lloc a les repeticions i desconnexions pertinents. Aquesta situació també pot donar-se en casos més propers a nosaltres. Sense anar més lluny, les connexions sense fils entre aparells mòbils (telèfons, ordinadors portàtils, etc) també poden patir llargues desconnexions si, per exemple, les ones que transmeten les dades troben obstacles

que no poden travessar.

És per aquests motius que ja aleshores es comencen a estudiar i proposar noves alternatives. Tot i així, no és fins l'any 2002 que apareix el terme que designa actualment aquest tipus de xarxes: Delay-Tolerant Networking (DTN). Més endavant, a l'any 2007, apareix el seu RFC [2].

Així doncs el que es busca és una arquitectura que permeti transmetre informació suportant talls constants en la connexió. D'aquesta manera seran possibles, tant les connexions sense fils que pateixin problemes, com les futures connexions interplanetàries, entre d'altres.

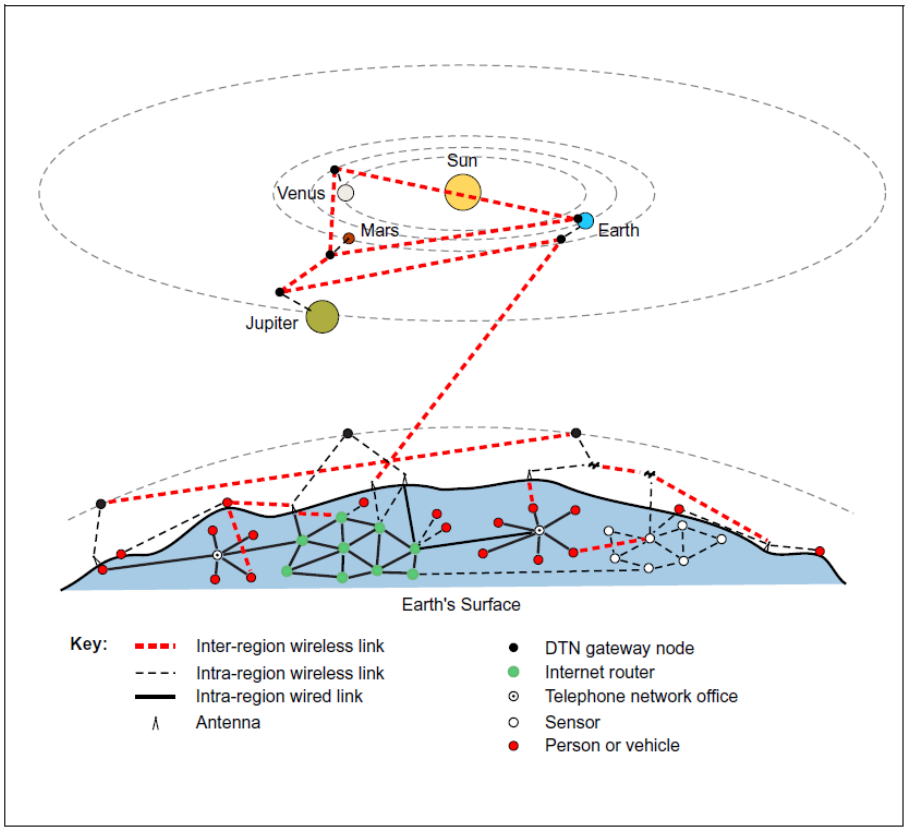

Figura 2.2: InterPlaNetary Network. Una de les idees que dóna lloc a les DTN.

2.3 Bundle Protocol (BP)

Com hem vist en la secció anterior, els protocols TCP/IP no poden fer-se servir en els entorns on faríem servir una arquitectura DTN. Per a poder solucionar aquest problema necessitem un nou protocol. S'han presentat bàsicament dos protocols: el Licklider Transmission Protocol i el Bundle Protocol (BP) [7]. Sembla ser que de moment el protocol que s'està imposant és el BP.

Aquest serà el protocol sobre el que nosaltres treballarem.

Aquest nou protocol es pensa com un afegit que ajudarà a solucionar els problemes en aquelles transmissions on poden haver-hi talls. La idea principal consisteix a transmetre la informació de punt en punt dins de la xarxa DTN. És a dir, volem que la informació vagi avançant pels diferents punts de la xarxa fins que arribi al seu destí. Per fer-ho, el BP s'afegirà com una capa més en el model que es fa servir per Internet, encarregant-se de que la transmissió en aquests casos es faci correctament. Aquesta nova capa s'anomena capa bundle (Figura 2.3).

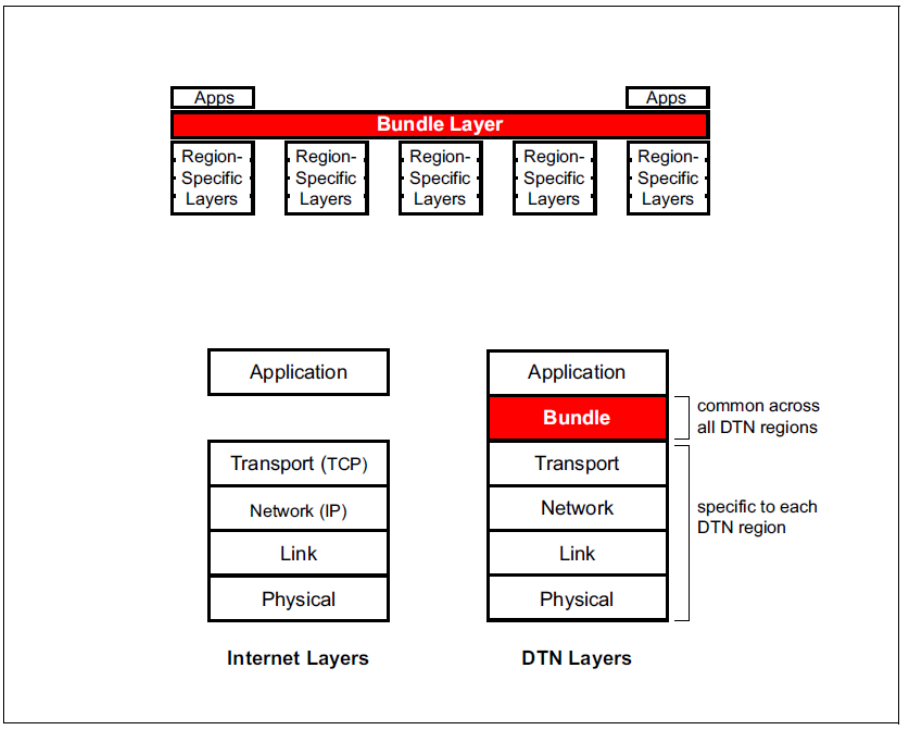

Figura 2.3: Capes en DTN.

Cal adonar-se, que una xarxa DTN s'entén com una xarxa de xarxes (cadascuna d'aquestes xarxes internes s'anomena regió) i que el BP serveix per a lligar-les (Figura 2.3). D'aquesta manera la informació pot viatjar tant per xarxes que són tolerants a talls com per xarxes que no ho són (com ara les que fan servir TCP/IP). Com que tenim una capa nova, també tenim una nova manera d'encapsular la informació. En aquest cas la informació encapsulada pel BP s'anomena bundle o missatge. A la figura 2.4 podem veure com s'encapsularien els bundles quan la informació passa a través d'una xarxa TCP/IP.

Els bundles es transmeten entre els diferents punts que implementen aquesta nova capa bundle. Com que volem que la nostra xarxa DTN sigui tolerant a talls també és necessari que els punts intermedis tinguin la capacitat d'emmagatzemar la informació fins que la connectivitat

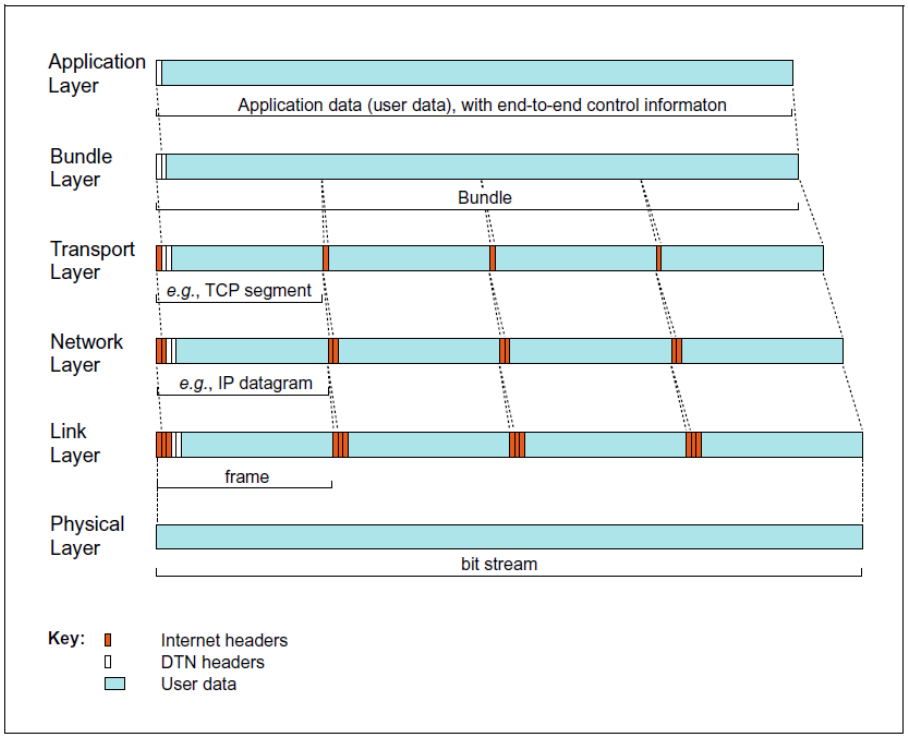

Figura 2.4: Encapsulament dels bundles en DTN.

amb el següent punt sigui viable. Els punts de la xarxa que poden emmagatzemar informació s'anomenen recipients i poden ser de tres tipus (Figura 2.5):

- Host: Actua com a emissor o receptor de bundle, és a dir, envia o rep missatges de la capa bundle.
- Router: S'encarrega de passar missatges dins d'una regió, però no té capacitat per a passar la informació d'una regió a una altra.
- Gateway: S'encarrega de passar missatges d'una regió a una altra.

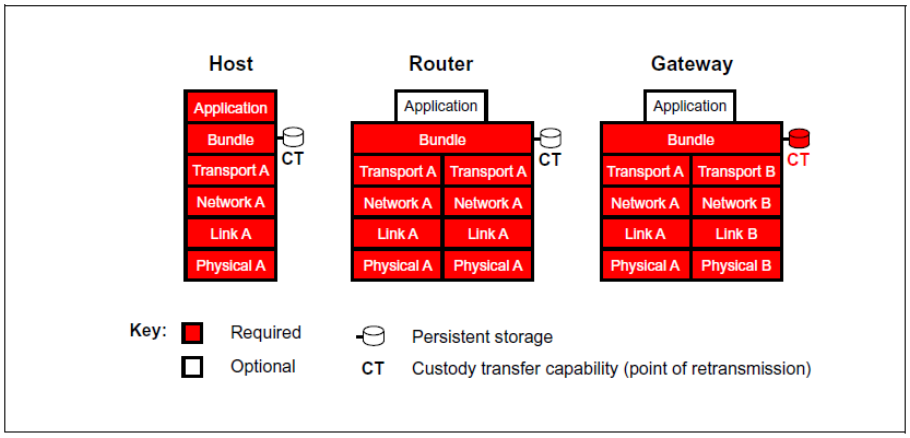

Figura 2.5: Recipients de la capa bundle.

Així doncs la informació que surt d'un host va passant pels routers i gateways. Aquests emmagatzemen la informació per si en algun punt del camí no tenim connectivitat. Quan n'hi ha passen la informació i aquesta arriba eventualment al seu destí.

Un exemple força clar seria el d'una transmissió entre la Terra i Mart:

1. Des d' un punt de la Terra s'envia informació cap a Mart.
2. El missatge s'encapsula en un missatge (bundle) i es passa a la capa de sota (com que segurament ho fem a través d'Internet, suposem TCP/IP).
3. TCP/IP s'encarrega de passar la informació fins el següent router i aquest el passa al següent. Al final el missatge arriba a un gateway des d'on surten els missatges que van de la Terra fins a Mart.
4. El gateway passa la informació al protocol que s'encarrega de les transmissions espacials.
5. La informació arriba al gateway de Mart el qual passa la informació a una suposada Internet marciana.
6. Finalment la informació travessa els routers i gateways que puguin haver-hi a Mart i arriba al seu destí.

Com podem veure, el BP no tan sols ens ajuda a solucionar el problema dels talls, sinó que també ens ajuda a simplificar la manera de comunicar xarxes heterogènies (i.e. que fan servir protocols diferents). Ara bé, tot i que tenim un mecanisme per transmetre informació a través de les DTN, encara hi ha molts aspectes, com ara la priorització d'informació o l'encaminament dels bundles a nivell d'aplicació, per als quals el BP no proporciona solucions adequades. Hi ha propostes amb protocols alternatius, com ara la proposada a [8] que proporcionen solucions a aquests i d'altres aspectes. Nosaltres ens basarem en [8], la qual es basa en fer servir agents mòbils, per donar una solució als problemes plantejats en el nostre projecte.

2.4 JADE i els Agents

En aquest apartat parlarem sobre els agents software [4] i sobre la plataforma JADE [5]. Aquests dos punts són molt importants ja que formen l'entorn amb el que treballarem i necessitem entendre exactament com funcionen per aconseguir assolir els nostres objectius. També explicarem de quina manera farem servir els agents i parlarem sobre l'IPMS [11], un Add-on de JADE que, com veurem, serà el centre del nostre treball.

2.4.1 Agents software

Un agent software pot ser descrit com una entitat software que funciona de forma contínua i autònoma en un entorn, habitualment habitat per altres agents i processos [4]. Aquesta és una de les descripcions més acceptades pels investigadors de la matèria, però no existeix una definició concreta per aquest terme. Tot i això, sí que hi ha tot un seguit de punts on els experts estan d'acord i que poden servir per definir que és un agent software. Són els següents[4]:

- **Persistència:** El codi de l'agent no s'executa per demanda, si no que s'executa contínuament i és el propi agent qui decideix quan s'ha d'executar depenent del context en que es troba.

- **Autonomia:** Els agents tenen capacitats de selecció de tasques, priorització, comportament dirigit per objectius i de presa de decisions sense intervenció humana.

- **Habilitats socials:** Els agents tenen la capacitat d'interactuar amb altres agents, fent servir algun tipus preestablert de comunicació i coordinació. Han de poder col·laborar en una tasca.

- **Reactivitat:** Els agents perceben l'entorn on es troben i reaccionen adequadament als seus possibles canvis

2.4.2 La plataforma JADE

Com hem vist a l'apartat anterior, els agents funcionen (i.e. s'executen) en un entorn. Podríem dir que els agents "viuenï interactuen dins d'aquest entorn. JADE (*Java Agent DEvelopment Framework*) és un d'aquests entorns. Implementat en Java i iniciat pel departament de R&D de Telecom Italia Group, JADE és un projecte de codi obert destinat al desenvolupament d'aplicacions basades en sistemes multi-agent. Els agents viuen en aquest entorn, anomenat generalment plataforma, on tenen la capacitat d'executar-se i comunicar-se amb la resta d'agents.

A l'any 2005 la FIPA (*Foundations of Intelligent Physical Agents*) [9] s'adhereix com a comitè de l'IEEE, amb l'objectiu de promoure la tecnologia dels agents i la utilització d'uns estàndards que millorin la interoperabilitat amb altres tecnologies. El projecte JADE proporciona una manera d'implementar sistemes basats en agents que segueixin aquests estàndards.

El fet de que JADE utilitzi Java fa que aquest entorn on viuen els agents pugui executar-se en qualsevol màquina, independentment del seu hardware, sempre que aquesta tingui la capacitat d'executar una màquina virtual de Java.

2.4.3 Els agents com a medi de transport

Fins ara hem vist com funcionen els agents i com és l'entorn on s'executen, però com volem fer-los servir? Com hem pogut veure en els primers apartats, una xarxa DTN no té una forma estàtica. El fet que puguin haver-hi talls o que alguns punts deixin d'estar operatius fa que aquesta canviï constantment i que, per tant, ens hàgim d'adaptar a aquests canvis. Els agents ens proporcionen una solució a aquest problema, ja que són entitats que tenen la capacitat d'adaptar-se al seu entorn i de reaccionar-hi conseqüentment (veure apartat 2.4.1).

Per tant, podem dir als nostres agents que agafin la informació de la capa bundle, i.e., els bundles o missatges, se'ls guardin, observin el seu entorn, decideixin quin camí seguir per arribar fins al seu destí i un cop allà, entreguin la informació. Ara bé, tot i que els agents que podem crear mitjançant JADE tenen la capacitat de viure en una plataforma, no tenen la capacitat de viatjar d'una plataforma a una altra. És per aquest propòsit que farem servir l'IPMS, del qual parlarem al següent apartat.

2.4.4 IPMS (*Inter-Platform Mobility Service*)

L'IPMS (*Inter-Platform Mobility Service*) és un Add-on creat per a la plataforma JADE que ens dóna la possibilitat de fer que els agents puguin viatjar d'una plataforma a una altra. Una de les habilitats més importants dels agents és la de comunicar-se amb altres agents. En JADE aquesta comunicació es duu a terme mitjançant missatges en un llenguatge especificat per la FIPA anomenat ACL [10].

Els agents poden comunicar-se tant amb agents que es trobin a la seva mateixa plataforma, com amb agents que es trobin en altres plataformes. Aquesta propietat de JADE és la que utilitza l'IPMS per als agents viatjar entre plataformes. Generalment, anomenem migració al fet que un agent viatgi d'una plataforma a una altra i anomenem agents mòbils a aquells agents que tenen la possibilitat de migrar.

Quan un agent vol migrar li comunica a la plataforma, concretament a l'IPMS, aquest para l'execució de l'agent i el deixa "adormit". Aleshores l'IPMS es posa en contacte amb la plataforma de destí i li envia tant el codi de l'agent com l'estat en que aquest es troba. Aquesta transmissió es fa mitjançant missatges ACL que contenen les diferents parts de l'agent. Quan la plataforma de destí rep l'agent per complet es comprova que no hi ha hagut cap error, s'esborra l'antic agent de la plataforma emissora i es desperta l'agent per que continuï la seva execució.

És important adonar-se que tot aquest procés és transparent per l'agent, és a dir, que un cop

s'ha fet la migració l'agent continua la seva execució com si no hagués passat res. Serà ell mateix qui comprovi si ha arribat al seu destí i qui continuï la seva execució d'una manera o una altre depenent de si la migració s'ha realitzat amb èxit o no.

Com podem veure, fent servir agents mòbils tenim una solució per a fer viatjar la informació a través d'una DTN. Tot i així l'aplicació d'aquesta idea no és tan senzilla, ja que cal trobar maneres de que els agents puguin rebre informació del seu entorn i que la facin servir de manera eficient. En el nostre cas, ens hem plantejat les següents preguntes:

- Si tenim més d'un agent a migrar en una plataforma i sabem que la comunicació amb el següent punt no durarà gaire, qui migra primer?
- I lligat amb la pregunta anterior. Com podem fer que l'agent no només arribi al seu destí, si no que també hi arribi el més ràpidament possible i passant per certs punts de la xarxa (per recollir informació, per exemple)? I si a més volem fer-ho sense malgastar energia?

Aquestes preguntes no tenen una única resposta, però als següents apartats intentarem trobar-hi una solució i implementar-la per demostrar que els agents mòbils poden ser d'utilitat a l'hora de transportar informació en les DTN.

Capítol 3

Anàlisi de requisits

En aquest capítol analitzarem els requisits necessaris per a la correcta realització del nostre projecte. Primer tindrem en compte els requisits funcionals, on explicarem què volem fer de forma general, i després els no funcionals, on analitzant quines són les nostres necessitats des del punt de vista tècnic. També estudiarem la viabilitat del projecte en totes les seves vessants.

En un segon apartat veurem quins han siguts els passos seguits en el disseny de la nostra implementació comentant perquè s'han pres unes decisions i no unes altres.

3.1 Requisits funcionals

Amb la realització del nostre projecte volem trobar, dissenyar i implementar una solució per a que els agents que viatgin a través d'una DTN tinguin llibertat per triar quin és el millor camí a seguir per arribar al seu destí. És possible que un agent hagi de passar per alguns punts en concret de la xarxa per a complir els seus objectius. Per això necessitem una forma de deixar decidir a l'agent el camí a seguir fins al seu destí.

També volem trobar una manera de decidir quin agent ha de migrar quan n'hi ha més d'un amb aquesta necessitat en una plataforma. Actualment l'agent que arriba primer és el que surt primer, però aquesta política no sempre és la millor.

Volem que les nostres solucions s'adaptin a l'actual disseny de l'IPMS i que el que implementem pugui ser usat en qualsevol agent que es programi per a la plataforma JADE.

Com a últim requisit volem que estalviïn el màxim possible d'energia en aquests processos. Molt probablement els dispositius que facin servir les nostres solucions faran servir bateries o dispositius similars, que poden esgotar-se fàcilment si no ho tenim en compte.

3.2 Requisits no funcionals

En aquest apartat veurem els requisits tècnics que hem de tenir en compte a l'hora de planificar el nostre projecte.

Requisits hardware

Com que la implementació que realitzarem es basarà en fer canvis o afegir nous elements a l'IPMS, necessitarem una màquina que pugui executar l'entorn de JADE. Per tant, com que aquest està programat en Java necessitarem que la màquina on s'executi JADE tingui disponible una màquina virtual de Java. Aquesta màquina virtual haurà de ser com a mínim la versió 1.4.

Requisits software

El fet que tot el que farem haurà d'executar-se finalment sobre la plataforma JADE significa que necessitarem un sistema operatiu que tingui disponible una màquina virtual de Java. A més, també necessitarem instal.lar l'IPMS com a Add-on de JADE, ja que els Add-ons no venen inclosos quan en realitzem la instal·lació. Un altre requisit important serà el de tenir un compilador per al llenguatge Java.

Requisits de comunicacions

L'aplicació que nosaltres volem fer dels agents implica la migració d'aquests a altres plataformes. Una màquina només pot arrencar una única plataforma JADE i per tant necessitarem tenir al menys dues màquines per poder fer les proves pertinents. Tot i així, en un entorn real, és d'esperar que es tingui més d'una màquina o dispositiu ja que sense més d'un dispositiu el que estem fent no té sentit. A més a més aquests dispositius hauran d'estar connectats en xarxa amb els corresponents ports oberts de manera que les plataformes es puguin detectar entre elles i puguin realitzar la migració dels agents correctament.

Requisits de memòria

En principi els requisits de memòria per a JADE i l'IPMS no són gaire grans. Només s'ha de tenir prou memòria per poder executar la màquina virtual de Java i que JADE pugui executar-s'hi. De tota manera, s'ha de tenir en compte que en un entorn real on es fessin servir els agents com a medi de transport, es podria necessitar una quantitat important de memòria per emmagatzemar tots aquests agents i la informació que transportessin. En un entorn DTN real, la informació pot anar arribant a un punt de la xarxa i no sortir-ne fins al cap de molt de temps. En aquests casos seria important tenir suficient memòria per no perdre informació.

Requisits de seguretat

És evident que quan parlem d'informació que ha de travessar una xarxa la seguretat sobre aquestes dades ha de ser un punt important a tenir en compte. Tot i així, el nostre projecte està centrat en encaminar els agents per les xarxes DTN i no en implementar seguretat en aquestes. Centrar-nos en la seguretat a més dels nostres objectius seria impossible ja que és un aspecte que no tindríem temps de tractar correctament. Ara bé, hem de tenir en compte que un dels avantatges d'un disseny basat en agents és que la protecció de la informació pot estar en els mateixos agents [12]. Per tant, la seguretat en el nostre cas no serà un aspecte crític.

3.3 Viabilitat del projecte

En aquest apartat analitzarem si el nostre projecte es viable des del punt de vista tècnic, econòmic i legal. D'aquesta manera podrem afirmar que és viable i que podem seguir endavant.

Viabilitat tècnica

En el nostre cas pretenem adquirir tot una sèrie de coneixements per aconseguir, finalment, crear un codi que ens serà útil en la gestió de les xarxes DTN. El nostre projecte està emmarcat dins d'altres projectes de proporcions molt més grans i per tant, la feina que realitzem no serà interessant només per a nosaltres, sinó per a tots aquests projectes. Tècnicament, el nostre projecte fa servir un protocol (Bundle protocol) que ja està estandarditzat en un RFC on s'hi descriu com ha de ser. També hem de tenir en compte que ja s'han realitzat implementacions similars. Els routers IP que es fan servir actualment per Internet, fan servir protocols semblants i que funcionen. A més, la plataforma JADE sobre la que farem el nostre projecte, és de codi lliure i per tant, no tindrem restriccions a l'hora de fer-hi les modificacions adients.

D'altra banda, la universitat ens proporciona un laboratori per projectistes que ens permetrà implementar i provar el nostre disseny. El laboratori conté varis ordinadors connectats ens xarxa que ens seran de gran utilitat a l'hora de fer proves amb agents mòbils.

És per tot això que el nostre projecte és viable tècnicament.

Viabilitat econòmica

Econòmicament, el nostre projecte fa servir un estàndards oberts (RFC), de manera que no hi haurà cap inversió en aquest aspecte. El software de desenvolupament és de codi obert, és a dir, que per aquest costat tampoc invertirem diners. El projecte té una durada d'unes 300 hores (15 crèdits), de les quals unes 30 s'han invertit en el curs de formació. Això vol dir que les hores

de treball en el projecte serien unes 270. Si suposem un preu de 20€/hora, que correspon al d'un Enginyer Tècnic Superior de Suport a la Recerca segons conveni amb la UAB i una única persona treballant en el projecte, el preu total seria de 5400€. Donats els resultats que es volen obtenir és un preu raonable. Per aquests motius, el projecte és viable econòmicament.

Viabilitat legal

Si ens centrem en l'aspecte legal del projecte, veiem que en cap moment es tracten dades de caràcter personal. Si més no, no es tracta amb aquest tipus de dades durant el desenvolupament del projecte. Per tant, això no suposa un problema legalment. D'altra banda tampoc infringeix cap llei espanyola ni cap patent. Es farà servir software lliure i la resta de recursos que puguin ser necessaris els proporciona la pròpia universitat. Per tant, legalment, el projecte també és viable.

Viabilitat general

Com hem comprovat en els apartats anteriors el nostre projecte és viable tècnicament, econòmicament i legalment. Donat que aquests punts són els que determinen la viabilitat general del projecte i com que els tres es compleixen, podem afirmar que el nostre projecte és viable.

3.4 Planificació

En aquest apartat exposarem la planificació temporal que es va proposar a l'inici del projecte.

A la taula 3.1 podem veure les tasques que es van planificar i les dates previstes d'inici i de final de cadascuna d'elles. Cal remarcar que aproximadament un 20% del temps total, un mes aproximadament, es va deixar com a marge per si trobàvem imprevistos. En cas que tot anés bé, aprofitaríem aquest temps per a realitzar ampliacions.

A la figura 3.1 podem veure el diagrama de Gantt, realitzat mitjançant el programa *Gantt Project* [13] on hi trobem tant les tasques previstes com les fites més importants. Aquestes últimes es van marcar com a *deadlines* per que s'haguessin de seguir més estrictament. Com podem veure tenim tres grans blocs. Un primer bloc que arriba fins a la finalització de l'anàlisi de requisits per entendre la situació en que ens trobàvem i experimentar amb JADE i els agents. Un segon bloc on faríem el disseny, la implementació i el testeig. I un tercer bloc que inclouria la realització de la memòria i la presentació final.

Al final de la memòria es presentarà la planificació real que s'ha dut a terme per la realització del projecte.

Tasca	**Data d'inici**	**Data de finalització**
Entendre DTNs i Protocol Bundle	11/01/10	14/01/10
Lectura de papers sobre cues a les DTNs	14/01/10	23/01/10
Entendre la gestió de cues dins les DTNs	25/01/10	3/02/10
Documentar-se sobre el projecte MAETTS	3/02/10	9/02/10
Entendre el funcionament dels agents mòbils i la plataforma JADE	9/02/10	23/02/10
Entendre i experimentar amb el Add-on	23/02/10	2/03/10
Anàlisi de requisits	2/03/10	5/03/10
Disseny de la nostra gestió de cues	5/03/10	16/03/10
Implementació del codi	16/03/10	13/04/10
Testeig del codi	13/04/10	27/04/10
Integració en un entorn real	27/04/10	11/05/10
Realització de la memòria	11/05/10	20/05/10
Preparació de l'exposició oral	20/05/10	29/05/10

Taula 3.1: Tasques planificades

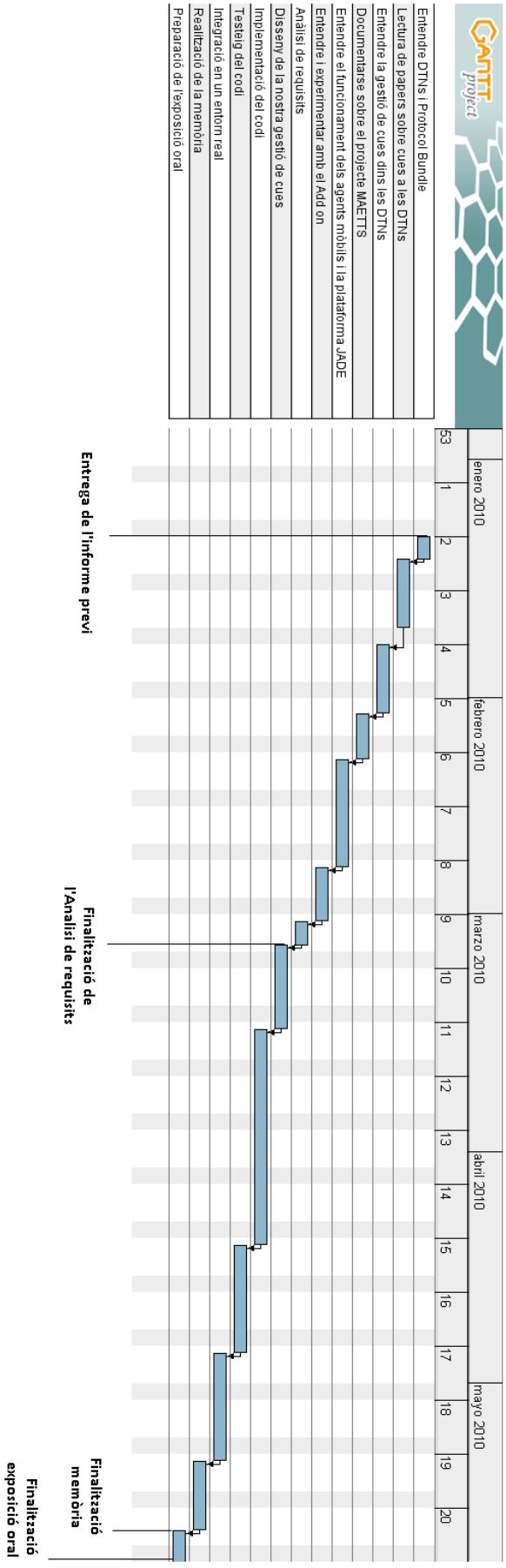

Figura 3.1: Diagrama de Gantt. Planificació temporal.

Capítol 4

Disseny i implementació

En aquest capítol exposarem el disseny pensat per la nostra solució. En un segon apartat exposarem com s'ha dut a terme la implementació basada en aquest disseny.

4.1 Disseny

En aquest apartat exposarem les decisions preses durant la fase de disseny del projecte. Primer explicarem quina era la situació en que ens trobàvem inicialment. D'aquesta manera tindrem una visió general del problema. Després exposarem les diferents parts del disseny, explicant perquè s'han pres unes decisions i no unes altres. Finalment, veurem el resultat global del nostre disseny i el compararem amb el de la situació inicial.

Durant la realització del projecte, la metodologia que es va dur a terme va ser la de dissenyar i implementar cadascuna de les parts independentment. Cada vegada que volíem afegir una nova millora pensàvem el seu disseny i fèiem la implementació. Tot i així, aquí presentarem primer tota la part de disseny i després tota la part de implementació, ja que durant les diferents fases van sortir molts dubtes i problemes que no podríem encabir en aquesta memòria si s'expliquessin part per part. Per això s'ha decidit fer-ho d'aquesta manera.

4.1.1 Situació inicial

Com s'ha explicat en apartats anteriors, l'IPMS és un Add-on de JADE que ens permet migrar agents d'una plataforma a una altra. Les millores que volem aportar estan centrades en decidir qui ha de saltar i cap a on vol fer-ho. D'aquestes decisions se n'encarrega l'IPMS i per aquest motiu el nostre disseny haurà d'inserir-se en el de l'Add-on. A la figura 4.1 podem veure la situació en que ens trobem quan l'IPMS posa a la cua els agents que vulguin sortir de la plataforma i tria quin serà el següent a migrar.

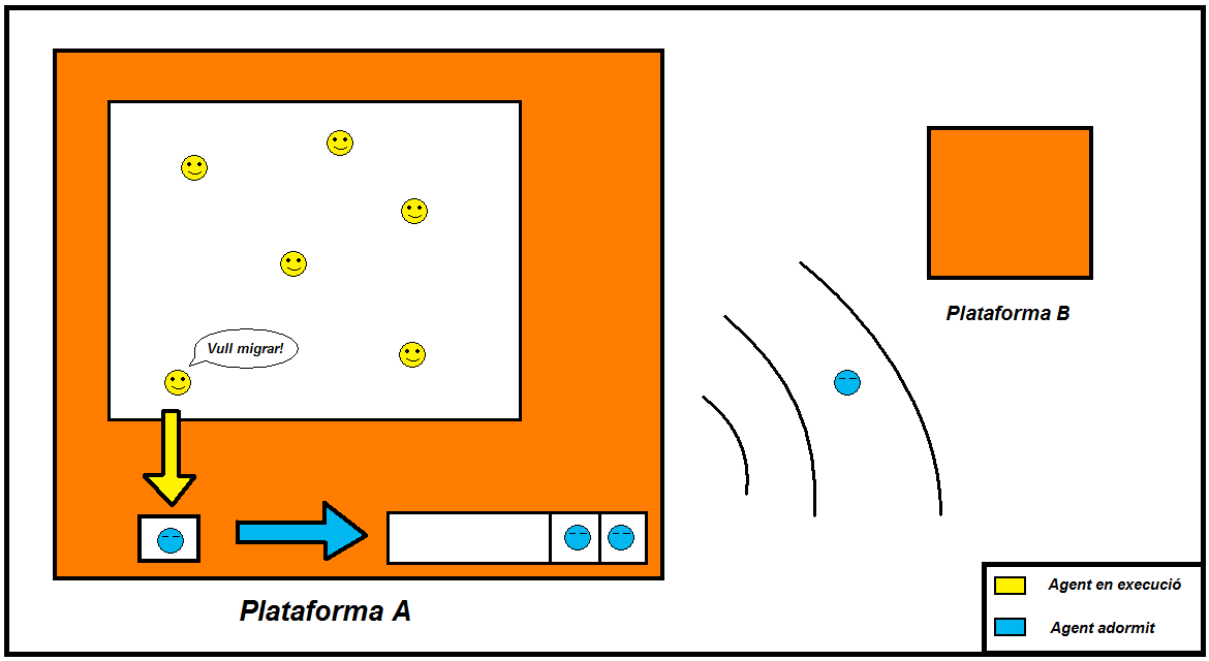

Figura 4.1: Situació inicial. Plataforma tal com l'implementa l'IPMS.

Quan un agent vol migrar li comunica a l'IPMS, aquest "l'adorm" i el posa a una cua FIFO (*First In First Out*) on s'espera. Quan és el seu torn, l'Add-on el treu de la llista i l'envia cap a l'altra plataforma, que el rep i se n'encarrega.

Reflexió: Els agents es desperten quan arriben a la següent plataforma

Com hem dit, quan els agents arriben a la nova plataforma, aquesta se n'encarrega. El que fa realment és despertar-los i fer que segueixin la seva execució. Per tant, els agents no s'adormen durant tot el camí fins al seu destí, si no que es van despertant a cada plataforma per la que passen. Sota el nostre punt de vista això no hauria de ser així. La DTN és un medi pel qual l'agent és transportat des del punt on es troba fins a un altre al qual vol arribar. No ens interessa que es desperti als nodes intermedis.

Per tant es va acordar que s'hauria de modificar la forma en que l'agent arriba a la plataforma per que no es desperti si no és la seva plataforma de destí. Tot i així, finalment, la implementació no es va dur a terme per que ens trobàvem en una fase força avançada del projecte i fer totes aquestes modificacions no era viable.

4.1.2 Disseny de l'encaminament d'agents

Un dels objectius plantejats al projecte és el d'aconseguir que els agents puguin decidir el camí que volen seguir. Aquesta habilitat podria implementar-se directament en els agents, doncs

aquests tenen la capacitat de prendre decisions basant-se en el seu entorn, però això implicaria dues coses:

1. Estaríem deixant tot l'encaminament dels agents a la capa d'aplicació i no a les capes inferiors.

2. Cada cop que l'agent hagués de prendre una decisió s'hauria de despertar i això pot consumir molts recursos.

Per tant, ens interessa trobar una solució que permeti a l'agent decidir el seu camí basant-se en uns criteris definits per l'aplicació per la qual treballa. A més, haurà de fer-ho de manera que s'estalvii el màxim d'energia possible.

Per solucionar aquests problemes, hem decidit que els agents que vulguin tenir aquesta habilitat hauran d'implementar un nou mètode. La plataforma executarà el mètode passant-li informació de l'entorn i aquest retornarà quina és la següent plataforma a la que l'agent vol saltar. D'aquesta manera, el nou mètode pot ser programat específicament per a cada agent com a criteri per decidir quin camí agafar, però tota la lògica es duu a terme a un nivell més baix.

Tot i que això és una bona solució, ens planteja un problema de disseny: només els agents que vulguin decidir el seu camí hauran de tenir aquest mètode, per tant, aquest no podrà ser un mètode de la classe que representa als agents. Per solucionar aquest problema, hem decidit que aquells agents que vulguin tenir aquesta nova habilitat hauran d'implementar una interfície.

Fer servir una interfície té dos avantatges. El primer és que una classe pot escollir si la vol implementar i per tant, a l'hora de programar un agent, només haurem de decidir si aquest vol aprofitar les noves funcions proporcionades. És com una peça que s'adhereix a la classe i que no interfereix en la resta del codi.

El segon avantatge és que una interfície és una classe on es declaren els mètodes, però no s'implementen. Serà la classe que implementa la interfície la que s'encarregarà d'implementar els mètodes com més li convingui. D'aquesta manera, podem definir quins seran els arguments i el valor de retorn del mètode independentment de com sigui per dins. En el nostre cas, l'agent podrà implementar aquest nou mètode de la forma que més s'adapti a la seva manera de viatjar per la xarxa, només preocupant-se de retornar un valor que identifiqui la següent plataforma a la que vol saltar.

Fent servir aquesta solució resolem també el segon problema. Com que és la plataforma i no pas l'agent qui crida al mètode, no és necessari que l'agent estigui en execució per prendre

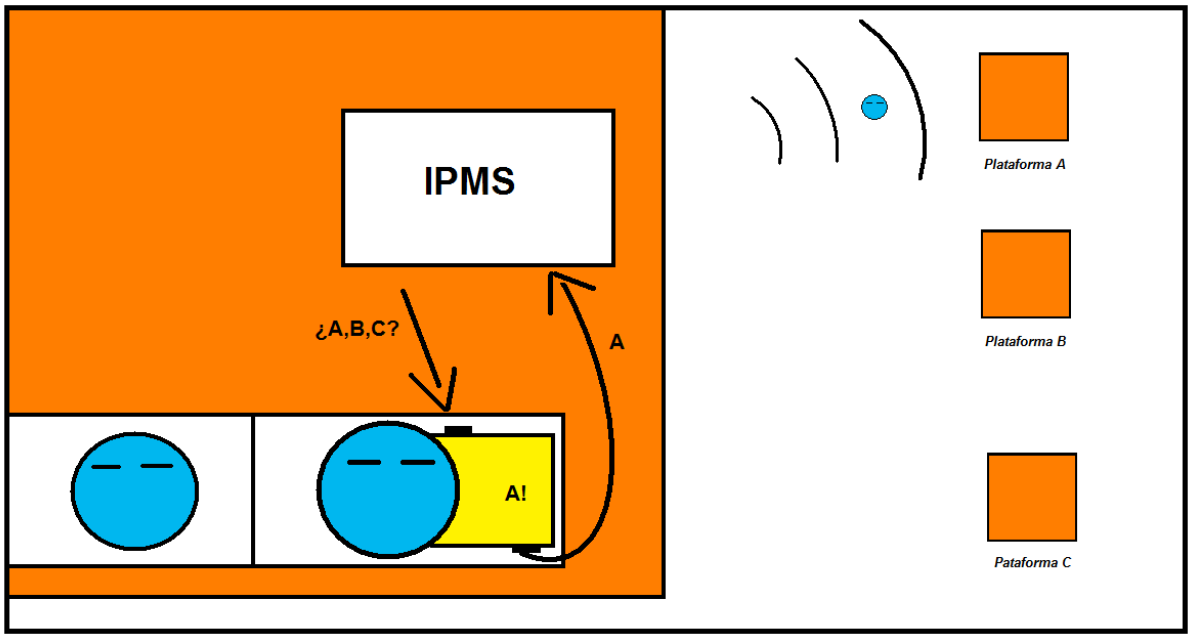

Figura 4.2: Crida al mètode d'encaminament.

les decisions. Quan la plataforma necessiti saber cap a on vol saltar l'agent, simplement buscarà el mètode i l'executarà (figura 4.2). No serà necessari "despertar" l'agent. Això ens permetrà estalviar energia, doncs despertar un agent que es troba a la cua és més costós que buscar el mètode de l'agent i que la plataforma l'executi.

Reflexió: Perquè és l'agent qui decideix la següent plataforma?

Durant el disseny d'aquesta part es va pensar en que fos directament la plataforma i no els agents qui decidís cap a on migraven els agents. Donat que la plataforma coneix el seu entorn i va rebent informació de com canvia aquesta, sembla lògic pensar que és més fàcil deixar la decisió en aquest nivell. D'aquesta manera no hauríem de buscar específicament el mètode de l'agent ni hauríem de fer que aquest implementés cap interfície.

Ara bé, el fet de tenir el disseny amb la interfície ens aporta una manera de poder encaminar els agents independentment de l'aplicació per la que treballi. Si no fos així, possiblement la plataforma hauria de saber com funcionen totes les aplicacions per saber com s'encamina cada agent. Òbviament això és impossible, doncs hi ha infinites aplicacions i codificar la plataforma de forma que en conegués només unes quantes no seria correcte.

A més, hi ha un altre motiu pel qual és interessant fer-ho d'aquesta manera. Els agents són els que viatgen per la xarxa i són ells els que coneixen millor la situació en que es troben. Si ho fem correctament, un agent pot anar recollint informació dels diferents punts de la xarxa i fer-la

servir per encaminar-se d'una forma més efectiva.

Aquests motius van fer que ens decidíssim pel disseny basat en la interfície.

4.1.3 Disseny de l'intercanvi d'informació

A l'apartat anterior hem vist quin ha sigut el disseny escollit per donar l'habilitat de decidir el seu camí als agents. Hem esmentat que fent servir aquest nou mètode l'agent podrà decidir la següent plataforma a la que vol saltar fent servir informació de l'entorn. Aquesta idea és bona sempre que l'agent obtingui informació de l'entorn que l'ajudi a decidir correctament. Si li donem, per exemple, informació que no s'ha actualitzat en molt de temps, l'agent podria prendre decisions equivocades. Per tant, necessitem una manera de proporcionar-li informació i de fer que aquesta informació sigui el més útil possible.

Per fer-ho hem decidit afegir un nou element a la plataforma, on es guardarà informació sobre l'entorn que l'envolta. Podríem dir que és una taula on es guarden diferents tipus d'informació. Aquesta taula es passarà com argument al mètode que pren les decisions, el qual podrà consultar-la i basar-se en la informació que li sembli més rellevant per decidir el següent salt de l'agent.

Com hem dit, ens interessa que la informació que hi ha en aquesta taula estigui el més actualitzada possible, de manera que l'agent pugui decidir de manera més efectiva. Aquesta informació hauria de ser intercanviada per les plataformes i ja que els agents han de viatjar-hi, perquè no fer-los servir per transportar aquesta informació? Donat que els fem servir per transportar informació de l'aplicació són un candidat perfecte.

Així doncs, el que farem serà posar una "motxilla" als agents on puguin emmagatzemar la informació. Quan l'agent arribi a una nova plataforma aquesta s'encarregarà de fer l'intercanvi i després el posarà amb els altres agents (figura 4.3). Per fer-ho, hem triat triat la mateixa opció que en l'apartat anterior. Crearem una nova interfície que contindrà tant els mètodes per fer l'intercanvi d'informació, com la motxilla on la ficarem. Aquesta motxilla podria ser de nou una taula com la que tenen les plataformes. Igual que en l'apartat anterior, fer servir una interfície ens permetrà que els agents decideixin si volen fer aquest intercanvi d'informació sense haver d'afegir nous elements a la classe dels agents. A més també permetrà que l'agent pugui fer l'intercanvi con més convingui, ja que depenent del tipus d'informació que transporti serà convenient fer-ho d'una manera o d'una altra.

En aquest cas hem de tenir en compte que, com que serà l'agent qui s'encarregui d'actu-

alitzar la taula, hem procurar que ningú pugui afegir informació incorrecta ni esborrar-ne de correcta. Per fer-ho ens basarem en el patró de disseny *facade*. Aquest patró implementa uns mètodes que serviran per accedir a la informació, tant per afegir-ne com per treure'n. No hi ha cap altre manera d'accedir-hi i per tant, a partir d'aquests mètodes podem controlar i validar les modificacions aplicades sobre la taula. Nosaltres implementarem una nova classe que farà servir aquest patró i substituirem la taula de la plataforma per un objecte d'aquesta classe. Al següent capítol explicarem amb més detall aquesta nova classe.

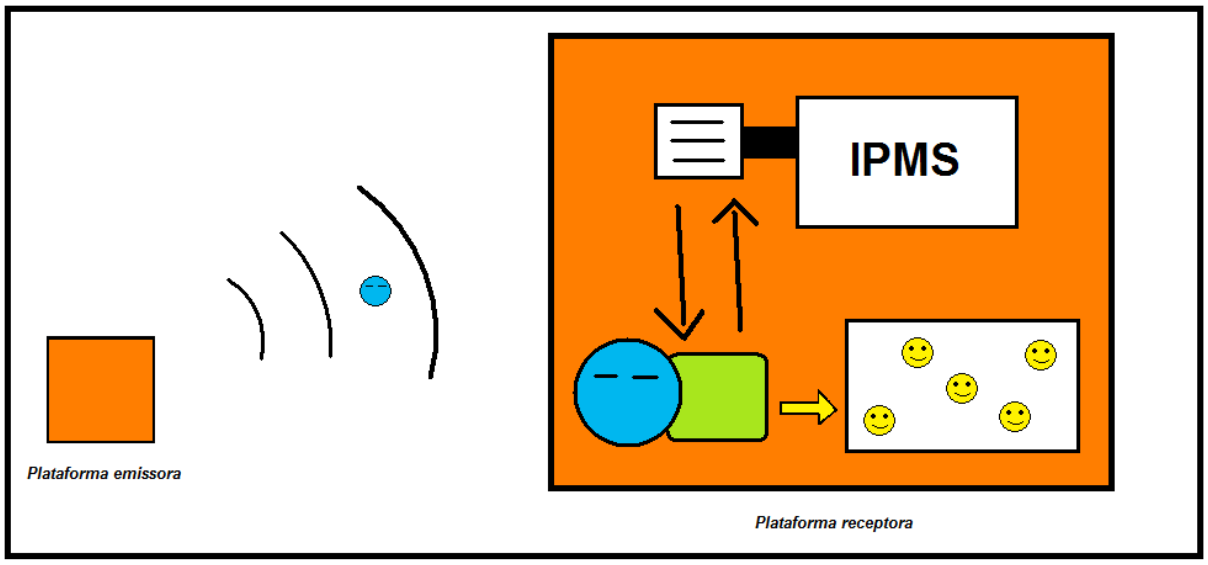

Figura 4.3: Intercanvi d'informació.

4.1.4 Resum d'encaminament amb intercanvi d'informació

Com a resum, a la figura 4.4 mostrem un diagrama de seqüència on es poden veure els passos que seguiria un agent que tria el seu camí i que intercanvia informació. El diagrama comença assumint que s'ha enviat un agent cap a la plataforma i que l'estem rebent:

4.1.5 Prioritats

Un altre dels nostres objectius era aconseguir que alguns agents fossin més prioritaris que a altres a l'hora de sortir de la plataforma. Al disseny actual de l'IPMS els agents es posen en una cua FIFO (*First In First Out*), això vol dir que el primer que es fica a la cua és també el primer que surt. Aquesta solució no sempre és la millor, doncs en les DTN un node pot estar molt de temps sense veure un altre node i la finestra de temps en que es pot transmetre informació pot ser molt petita. Per tant, hem d'aprofitar aquesta finestra al màxim i transmetre els agents que més interessi que viatgin a aquest node, no només els que estan primers a la cua.

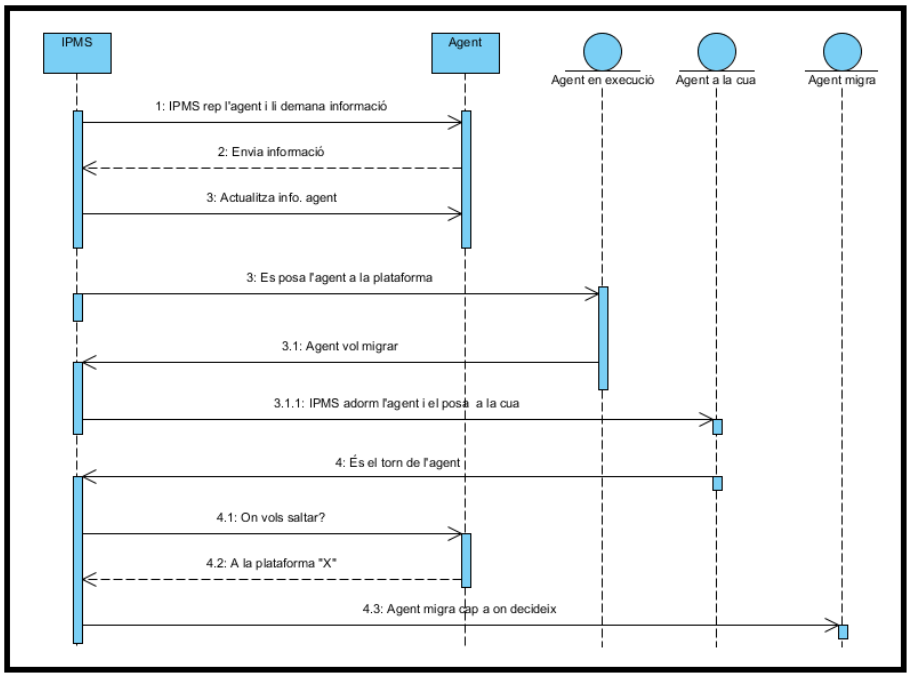

Figura 4.4: Encaminament amb intercanvi d'informació.

La nostra idea, basant-nos en [8], és que s'executi un mètode a la plataforma que doni una prioritat a cada agent basant-se en diferents criteris. En el moment de transmetre informació la plataforma triarà primer aquells agents que tinguin més prioritat. Tot i així, implementar un disseny com aquest pot resultar molt complicat, així que nosaltres només canviarem la cua FIFO de la plataforma per una cua amb prioritats. Per fer-ho, modificarem la classe actual de la cua de manera que hi hagi prioritats, suposant que ja existeix un mètode per assignar-les.

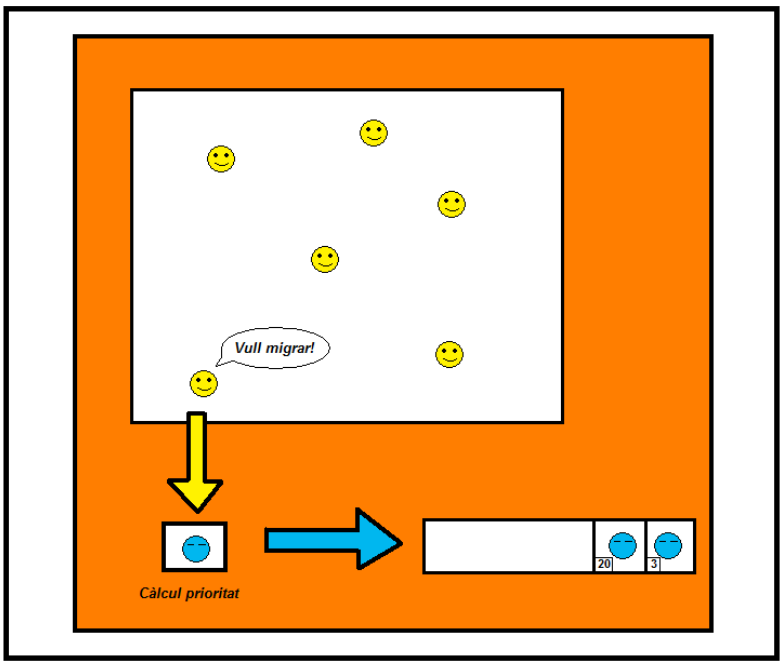

Figura 4.5: Cua amb prioritats.

4.1.6 Situació final

A la figura 4.6 podem veure quina seria la situació afegint els canvis explicats en els apartats anteriors. Podem comprovar que respecte la figura 4.1 s'ha afegit l'intercanvi d'informació quan un agent arriba a la plataforma i que després aquest es posa a executar amb la resta d'agents. També veiem com s'ha afegit un pas intermedi que posa prioritats als agents (tot i que com ja hem dit, aquesta part no s'implementarà) i com finalment se li pregunta a l'agent cap a on vol migrar.

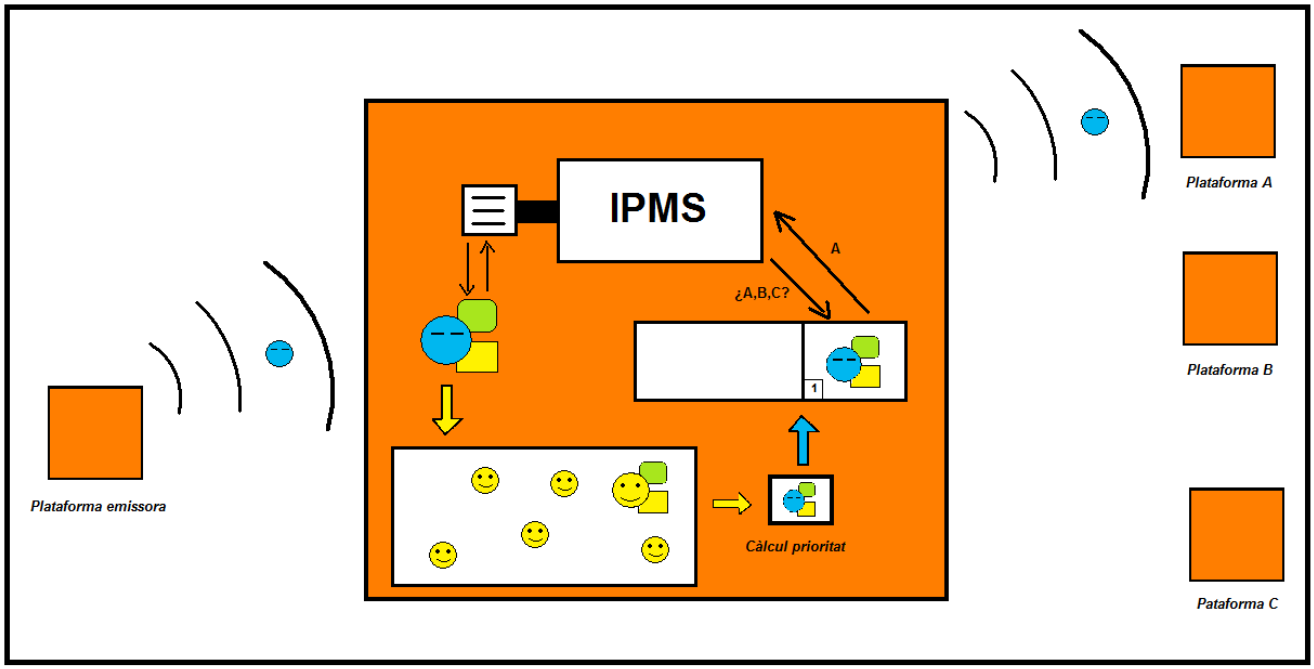

Figura 4.6: Situació final. Situació amb els canvis afegits.

4.2 Implementació

En aquest apartat explicarem quin ha estat el llenguatge de programació triat per dur a terme la implementació i perquè l'hem escollit. Seguidament, explicarem com s'ha implementat finalment la solució.

4.2.1 Llenguatge de programació escollit

Com ja es va explicar als primers apartats, per dur a terme la implementació del nostre projecte farem servir el llenguatge de programació JAVA. L'elecció d'aquest llenguatge ve donada pel fet que els nostres agents hauran de viure en una plataforma JADE, la qual està implementada i està adaptada per crear agents programats en JAVA. Intentar crear agents en un altre llenguatge i adaptar-los a JADE no ens aportaria masses avantatges i ens suposaria perdre temps. D'altra banda, aquest projecte no esta centrat en obtenir una solució amb el millor rendiment possible, si no en obtenir una solució que funcioni i que demostri que la nostra idea pot ser implementada en un entorn real. Per tant, donats aquests motius i el fet que volem fer servir JADE per ser la plataforma més estesa, el llenguatge de programació que farem servir serà JAVA.

4.2.2 Implementació de l'encaminament d'agents

Al capítol anterior hem decidit que per aconseguir que els agents puguin decidir quin camí volen seguir farem servir una interfície JAVA. Hem anomenat aquesta interfície *Routable* (veure figura 4.7) pel fet que ajudarà a l'agent a poder encaminar-se cap al seu destí. Quan programem un agent que vulguem que tingui aquesta habilitat, només haurem de fer que la classe del nostre agent implementi la interfície.

Com hem comentat al capítol anterior, l'avantatge de la interfície és que els seus mètodes estan declarats però no estan implementats. Serà el programador de l'agent el que s'encarregui d'implementar-los. En el nostre cas només necessitem tenir un mètode, el que farà servir l'agent per decidir. Hem anomenat aquest mètode *nextHop*, perquè ens donarà el destí del següent salt que faci l'agent. Aquest rep com a paràmetre els veïns de la plataforma, és a dir les plataformes que té a l'abast, el seu destí final i la taula de la plataforma per que l'ajudi en la seva decisió. El valor que retorna el mètodes és l'identificador de la plataforma a la que vol saltar. En el nostre cas, per fer-ho més simple, fem servir com a identificador de plataforma la IP de la màquina on aquesta s'executa.

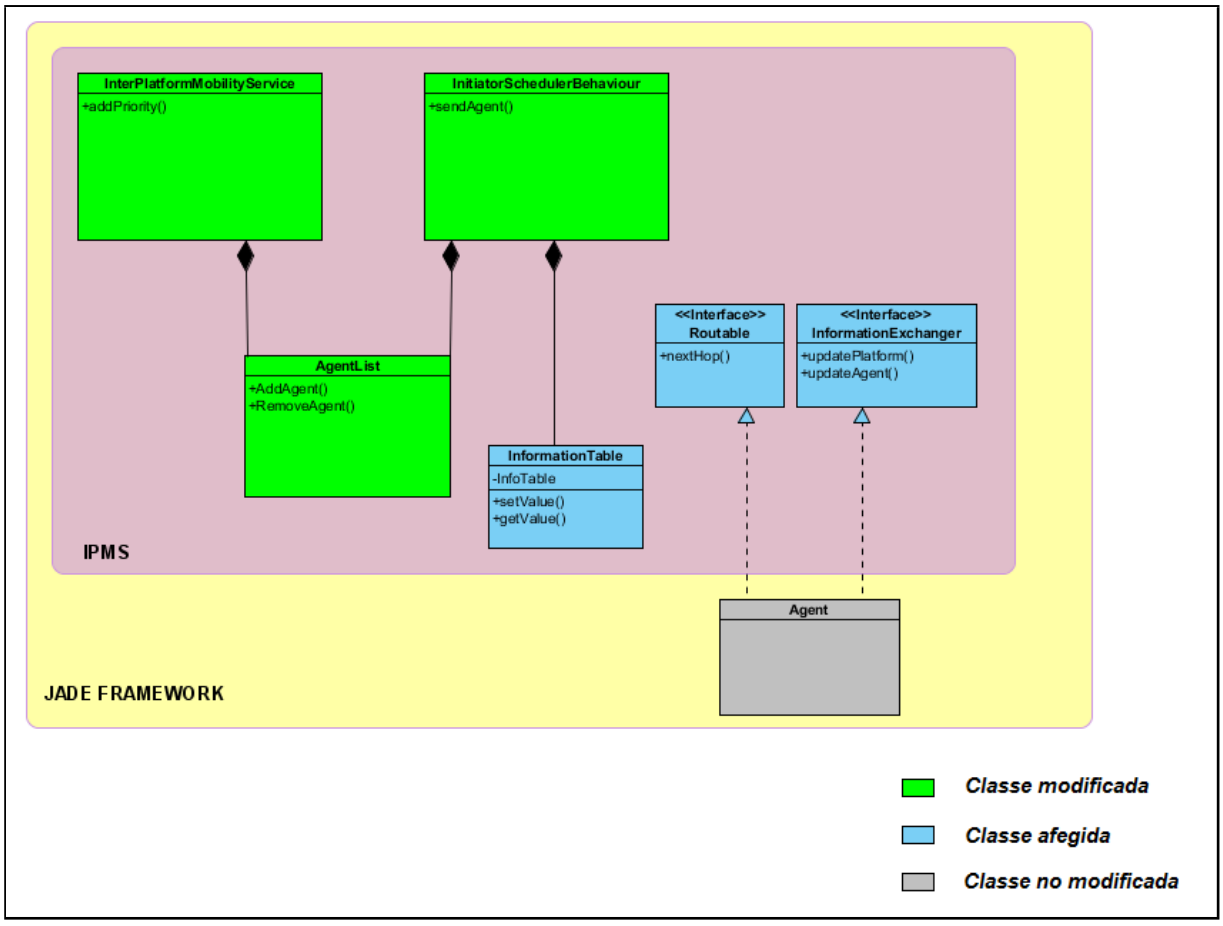

Figura 4.7: Diagrama de classes de la solució.

Finalment l'estructura d'aquesta interfície seria la següent:

```
public interface Routable{

    public String nextHop(InformationTable infoTable, String[] neighbours, String finalDestiny);

}
```

4.2.3 Implementació de l'intercanvi d'informació

Com en el cas anterior, aquí també fem servir una interfície per solucionar el nostre problema. En aquest cas la interfície s'anomena *InformationExchanger* (figura 4.7). D'altra banda, també necessitem un lloc on guardar la informació. Per simplificar-ho fem servir una taula *hash* per guardar-la. La plataforma també tindrà una taula com la dels agents, tot i que, com comentarem més endavant, serà una mica diferent. Aquesta taula relacionarà les IPs de les plataformes amb un número enter. Com més baix sigui aquest número, millor serà la plataforma.

En aquest cas necessitarem dos mètodes, que hem anomenat *updatePlatformInformation* i *updateAgentInfoTable*.El primer mètode serveix per actualitzar la taula de la plataforma i el segon per actualitzar la taula de l'agent. Els dos mètodes reben la taula de la plataforma com a paràmetre i retornen un valor booleà. Com que en JAVA al passar un objecte a un mètode aquest es passa sempre per referència no cal retornar la taula, doncs els canvis que s'hi apliquin ja quedaran guardats.

Com veiem, és el propi codi de l'agent, escrit pel programador, el que s'encarrega d'actualitzar la taula. Fer-ho d'aquesta manera té l'avantatge de que l'agent pot fer servir informació recollida d'altres plataformes per actualitzar la taula, però té el desavantatge de que un codi mal escrit pot manipular incorrectament la informació de la taula. Per evitar-ho, com comentàvem al disseny, fem servir una nova classe. Hem anomenat aquesta classe InformationTable i està dissenyada seguint el patró *facade*, el qual serveix per controlar els accessos a un objecte. Aquest patró manté l'objecte a protegir com a variable privada de la classe i implementa mètodes que serveixen per accedir-hi. D'aquesta manera, cada cop que l'agent vulgui tenir accés a la taula de la plataforma ho haurà de fer a partir dels mètodes d'aquesta classe, que s'encarregaran de que no s'hi facin canvis il·legals.

L'estructura de la interfície serà doncs:

```
public interface InformationExchanger{

        public boolean updateAgentInfoTable(InformationTable p_it);

        public Hashtable<String, Integer> getAgentInfoTable();

        public boolean updatePlatformInformation(InformationTable p_it);

        Hashtable<String,Integer> agentInfoTable = null;

}
```

4.2.4 Prioritats

La implementació de les prioritats en la cua d'agents s'ha implementat d'una forma molt senzilla. Inicialment l'IPMS tenia una classe anomenada *AgentList*, que mantenia un vector d'identificadors d'agents i que implementava uns mètodes per afegir i extreure agents d'aquesta cua seguint el model FIFO. Nosaltres hem afegit un segon vector, el qual guarda els valors de les prioritats de cada agent. Per saber la prioritat d'un agent, només hem de mirar la mateixa posició al vector de prioritats.A més també hem modificat el mètode que extreu agents per que retorni l'agent amb la prioritat més alta.
Com veiem, la implementació s'ha fet de forma senzilla però funcional.

4.2.5 Programa principal

A l'IPMS la classe que s'encarrega d'agafar els agents de la cua i preparar-los per migrar s'anomena *InitiatorScheduleBehaviour*. És en aquesta classe que hem afegit modificacions per poder aprofitar els avantatges de les nostres millores.

Cada cop que aquesta classe agafa un agent per migrar (agafarà el que tingui més prioritat pels canvis afegits a la classe *AgentList*) haurem d'executar el seu mètode de decisió. Ara bé, això no sempre serà possible, doncs també podem tenir agents que no implementin *Routable* i que, simplement, no tinguin aquest mètode. Per solucionar aquest problema hem aprofitat el que en JAVA s'anomena reflexió. Aquesta propietat ens permet obtenir informació a baix nivell d'un objecte i ens ajuda a saber si un objecte implementa una interfície. D'aquesta manera, cada cop que trèiem un agent de la cua comprovem si implementa la nostra interfície i, si ho fa, executem el seu mètode *nextHop*.

En el cas de l'intercanvi d'informació aquests mateixos canvis s'apliquen a la classe *InterPlatformMobilityService* però fent les comprovacions abans de posar l'agent a la cua. També és

en aquest punt quant es posen les prioritats als agents.

Finalment, per il·lustrar millor quins són els passos que es segueixen, mostrem el diagrama de flux de la nostra implementació (figura 4.8).

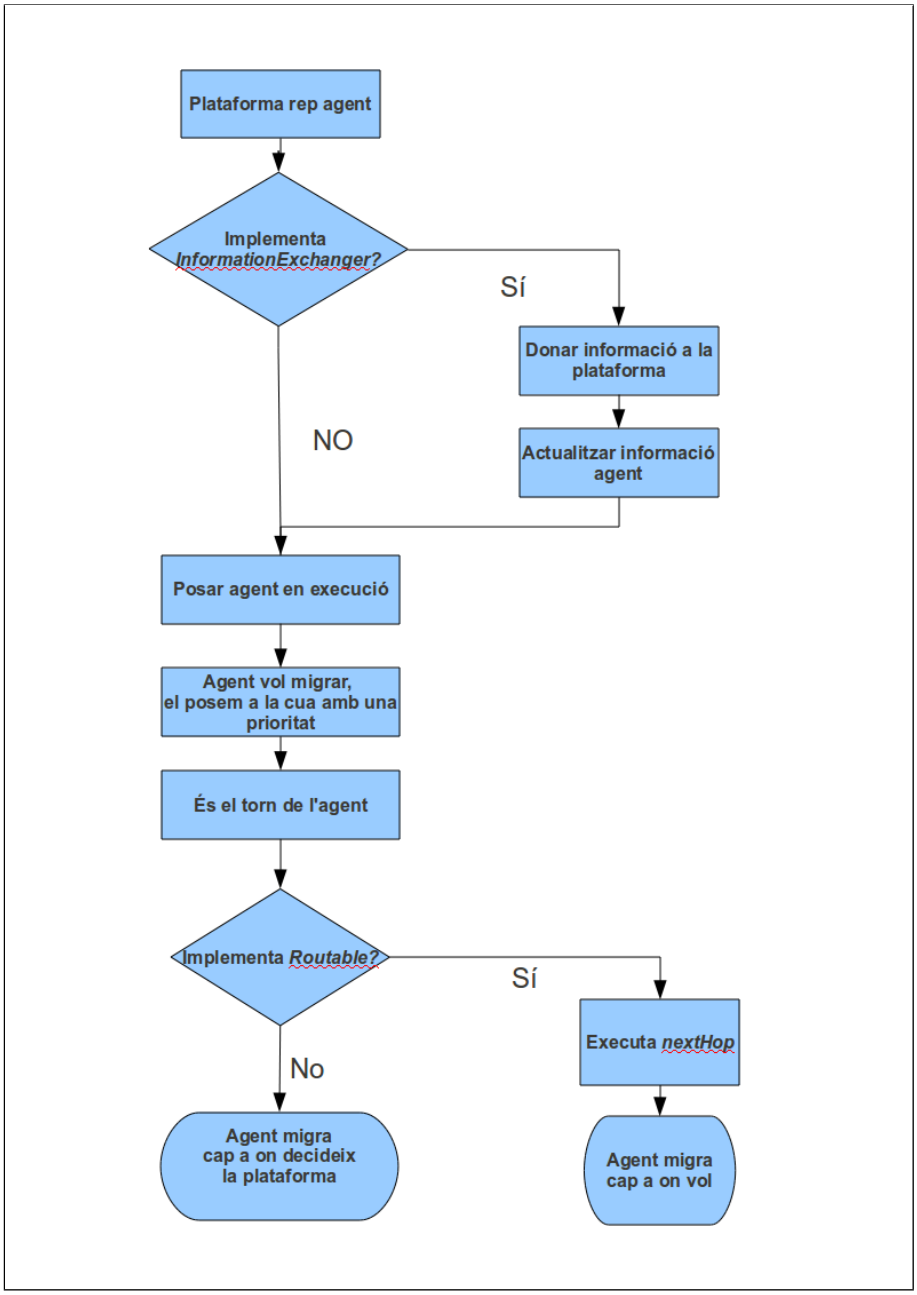

Figura 4.8: Diagrama de flux de la implementació.

Capítol 5

Proves i resultats

En el primer apartat d'aquest capítol explicarem quines proves s'han dut a terme per tal de validar la nostra solució. En un segon apartat valorarem els resultats obtinguts en aquestes proves. Per últim, exposarem quina ha estat la planificació final del projecte per tal de comprovar si hem aconseguit adaptar-nos a planificació que teníem inicialment.

5.1 Proves

En aquesta secció explicarem de quina manera hem comprovat si la nostra implementació funciona com esperàvem i quines han estat les diferents proves realitzades per tal de comprovar-ho.

5.1.1 Entorn de proves

Per tal d'avaluar la nostra solució hem muntat un petit entorn de proves. Aquest entorn consisteix en quatre ordinadors de la sala de projectistes connectats en xarxa. Aquests ordinadors formen una xarxa interna del tipus 192.168.10/24. A més, també formen un sistema centralitzat mitjançant un disc NFS. Això ens permet, per exemple, fer servir un mateix fitxer per configurar tots els ordinadors a la vegada.

La topologia de la xarxa que formen els quatre ordinadors podríem dir que té forma de rombe (figura 5.1). Per fer les proves, els agents sortiran de la plataforma número 1 i tindran dues plataformes cap a les que saltar. La plataforma 2 es considera millor, és a dir, a la taula d'informació de la plataforma 1 la IP de la plataforma 2 estarà associada amb un número més petit que el de la plataforma 3. Un cop hagin fet el primer salt, veuran la plataforma número 4, que és la plataforma de destí, i saltaran cap a aquesta última. Tot i que és una situació molt senzilla, és suficient per comprovar si la implementació funciona.

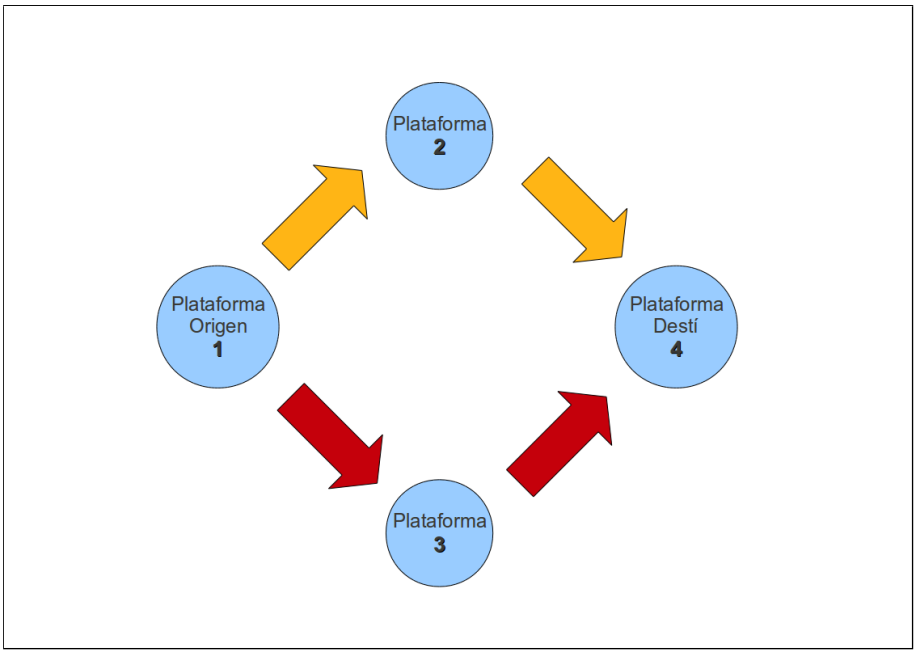

Figura 5.1: Representació de l'entorn de proves.

5.1.2 Aplicació de proves

A més de tenir l'entorn de proves també hem implementat una petita aplicació que ens permet fer proves més fàcilment. Aquesta aplicació executarà n agents a la primera plataforma que tindran una prioritat donada per l'aplicació i tindran com a destí la plataforma número 4. A més, els agents implementaran les dues interfícies, però implementant els mètodes de la interfície *Routable* de forma diferent. Els mètodes de l'altre interfície són iguals, doncs en aquest entorn no té gaire sentit realitzar l'intercanvi d'informació de formes diferents. Els agents no acumulen prou informació com per haver de fer-ho de diferent manera.

5.1.3 Proves realitzades

En aquesta apartat explicarem quines han estat les proves realitzades, dividint-les en proves de caixa negra i de caixa blanca.

Caixa negra

Aquestes proves les hem realitzat per comprovar si el comportament de la implementació és el que s'esperava. Per fer-ho hem fet servir la nostra aplicació per a llençar agents dins l'entorn de proves. Més concretament les proves realitzades són:

P1 Llencem 300 agents des de la plataforma número 1. La meitat tenen prioritat alta i l'altre meitat baixa. Tots implementen *Routable* de la mateixa manera, intentant fer el següent salt cap a la millor plataforma. Mentre els agents viatgen es canvia com tracten les prioritats les plataformes de manera que aquestes agafin primer els de prioritats baixes. Amb aquesta prova pretenem que durant la primera fase arribin primer els agents amb més prioritat i durant la segona fase es vegi com van arribant primer els que tenen una prioritat més baixa.

P2 Llencem 300 agents tots amb la mateixa prioritat. Ara la meitat implementen *Routable* seguint sempre la millor plataforma i l'altre meitat seguint sempre la pitjor. Òbviament el comportament dels segons agents no és massa coherent, però ens serveix per comprovar que segueixen correctament els criteris imposats. Durant la realització de la prova farem desaparèixer una plataforma simulant que aquesta queda fora de l'abast de les altres o que simplement ha caigut. D'aquesta manera també podrem comprovar que els agents tenen la capacitat de triar un camí alternatiu per arribar al seu destí, encara que la xarxa canviï mentre ells hi viatgen.

P3 Aquest cop llencem només 10 agents que implementen *Routable* igual que en la prova anterior. En aquest cas també implementen *InformationExchanger*. Mitjançant aquesta prova volem comprovar que els agents poden transmetre la informació de la primera plataforma cap a les següents i que uns no desfan el que han fet els altres.

P4 Finalment, l'última prova consisteix a llençar els agents amb tot el que hem posat a les altres proves. Els agents implementen *Routable* de les dues maneres, implement *InformationExchanger*, tenen prioritats diferents, a la meitat de la prova es fa el canvi a les plataformes i també en fem desaparèixer una. Aquesta prova pretén demostrar que tots els elements poden conviure correctament.

Caixa blanca

Aquestes proves pretenen validar la implementació des d'un punt de vista més tècnic. Com que en el nostre projecte la implementació només era el punt final, no hem realitzat proves molt exhaustives en aquest apartat. Tot i així si que hem fet proves que executaven tot el codi afegit per poder verificar que cap part del codi falla.

5.2 Resultats de les proves

En aquest apartat analitzarem els resultats de les proves realitzades al nostre escenari de validació.

5.2.1 Resultats de la priorització

Com hem esmentat a l'apartat de disseny, la implementació que hem realitzat per obtenir prioritats a la nostra cua d'agents no inclou la part en que es donen les prioritats als agents. Tot i així, sí que és interessant veure si, finalment, aplicar prioritats als agents pot ser beneficiós en la gestió de les DTN.

Els resultats obtinguts a les proves confirmen que les prioritats poden ajudar-nos a que un grup d'agents no es quedin col·lapsats en un node. En un primer moment veiem que si no fem cap modificació a les plataformes de proves, els agents amb més prioritat arriben, generalment, abans que els altres al seu destí. Això ja és un resultat positiu doncs vol dir que la priorització funciona correctament.

A més, si en un moment donat canviem la forma en que les plataformes processen les prioritats, veiem que els agents que abans trigaven més a arribar comencen a arribar abans. Per tant, hem aconseguit que els agents que es quedaven aturats als nodes intermedis, saltin abans que els altres i segueixin endavant. Cal tenir en compte que en un escenari real els agents amb menys prioritat podrien haver-se quedat molt de temps aturats en els nodes intermedis.

Tot i que seria més interessant veure aquests resultats amb agents que tenen una prioritat assignada per la plataforma (recordem que la complexitat d'aquesta part no ens ha permès implementar-la), veure que la cua amb prioritats funciona també és positiu, doncs d'aquesta manera hem pogut validar el seu funcionament. Si les proves funcionen amb prioritats posades per nosaltres també funcionaran amb prioritats posades per la plataforma (sempre que aquesta les posi coherentment).

5.2.2 Resultats de l'encaminament dels agents

Els resultats que esperàvem obtenir en la part d'encaminament dels agents eren bàsicament tres:

1. Aconseguir que els agents arribessin al seu destí encara que algunes parts de la xarxa quedessin fora de servei.
2. Aconseguir que seguissin uns criteris, donats per la interfície *Routable*, a l'hora de trobar el camí cap als seus destins encara que la xarxa canviï durant el seu viatge.
3. Aconseguir que poguessin transportar informació per tenir informades les plataformes dels canvis en el seu entorn.

El primer punt podem dir que s'ha acomplert. Tot i que la xarxa de proves no és gaire complexa i els agents flueixen sempre en la mateixa direcció, són capaços d'arribar al seu destí

encara que en un moment donat es faci caure un node de la xarxa. Com que el camí que segueixen no està marcat pel punt emissor, els agents van avançant per la xarxa buscant sempre un node que estigui actiu i que serveixi per anar cap al seu destí. Això es un resultat positiu, doncs els agents estan seguint exactament el comportament que s'espera d'una xarxa DTN. Encara que durant la transmissió d'informació alguns nodes no estiguin disponibles, aquesta es capaç d'arribar seguint un altre camí sense necessitat de tornar a realitzar una connexió, com hagués passat amb TCP/IP per exemple.

El segon punt també podem dir que s'ha acomplert, tot i que les proves realitzades no han estat molt exhaustives. Els criteris que s'han aplicat als agents eren força simples, però han demostrat que aquests poden arribar al seu destí per diferents camins fent servir uns criteris imposats. Quan tenien dos possibles camins a seguir, els agents que havien de seguir el camí que tingués una valoració més positiva anaven per aquest i els que havien de fer el contrari anaven per l'altre. Tot i així, tots arribaven finalment al seu destí. Com que en el nostre entorn de proves no hi ha camins millors que altres, tots els agents trigaven aproximadament el mateix a arribar al destí, però ho feien seguint camins diferents. Aquest resultat el podem considerar molt positiu, doncs era la part que més "novetats" aportava a la gestió de DTN i era la que més ens interessava que funcionés correctament.

Finalment, el tercer punt també podem dir que s'ha acomplert, però de forma senzilla. El nostre entorn de proves no té una complexitat molt gran i per tant la informació que les plataformes puguin intercanviar no juga un paper prou significatiu. Tot i així hem pogut veure com la informació es transmetia entre les plataformes, que era l'objectiu més bàsic que ens havíem plantejat.

5.3 Planificació final

A continuació exposem la taula de tasques i el diagrama de gantt de la planificació final (taula 5.1 i figura 5.2). Com es pot comprovar hem fet algunes modificacions a la planificació.

Hem tret la tasca de documentar-se en MAETTS ja que finalment no ha tingut molta importància en el nostre projecte. A més també hem hagut de fer servir temps extra del que havíem deixat al final perquè vam afegir algun objectiu més i la part d'implementació i disseny es va allargar més del que esperàvem. També podem apreciar que les tasques de disseny i implementació estan superposades. Com es va comentar al capítol 4, cada cop que es dissenyava una nova part, també s'implementava i per tant aquestes tasques s'han dut a terme a la vegada.

Tot i els canvis en la planificació finalment hem aconseguit, en part gràcies al temps extra que es va deixar a la planificació inicial, adaptar-nos als canvis i realitzar el projecte a temps.

Tasca	**Data d'inici**	**Data de finalització**
Entendre DTNs i Protocol Bundle	11/01/10	14/01/10
Lectura de papers sobre cues a les DTNs	14/01/10	23/01/10
Entendre la gestió de cues dins les DTNs	25/01/10	3/02/10
Entendre el funcionament dels agents mòbils i la plataforma JADE	3/02/10	17/02/10
Entendre i experimentar amb el Add-on	17/02/10	5/03/10
Anàlisi de requisits	5/03/10	17/03/10
Disseny de la nostra gestió de cues	17/03/10	10/04/10
Implementació del codi	25/03/10	17/04/10
Testeig del codi	17/04/10	6/05/10
Integració en un entorn real	6/05/10	20/05/10
Realització de la memòria	14/05/10	14/06/10
Preparació de l'exposició oral	14/06/10	23/06/10

Taula 5.1: Tasques finals

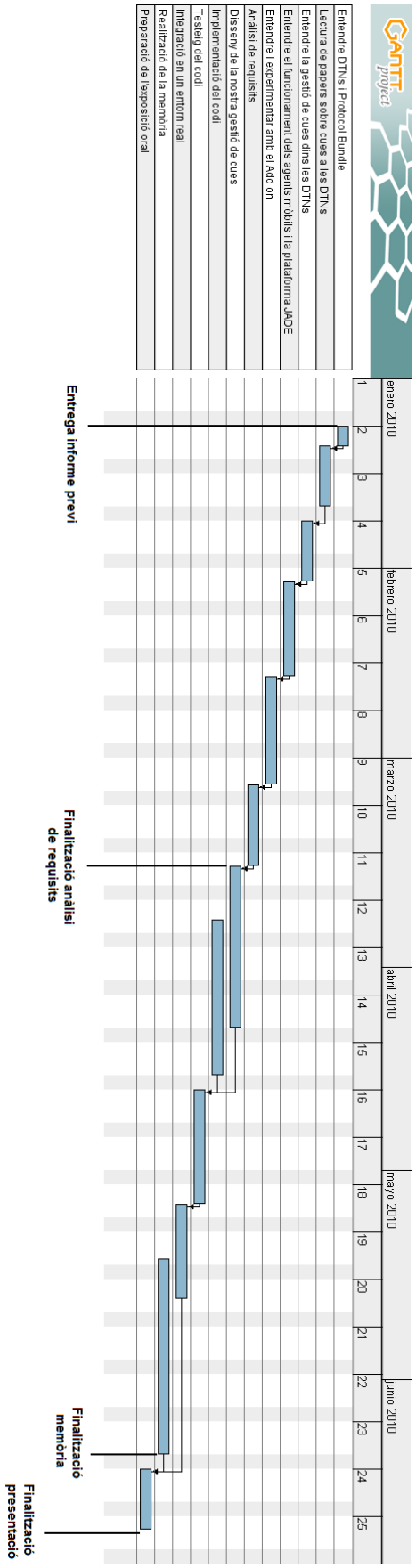

Figura 5.2: Diagrama de Gantt. Planificació final.

Capítol 6

Conclusions

En aquest capítol repassarem els objectius que ens havíem plantejat a l'inici del projecte i valorarem en quina mesura hem aconseguit assolir-los. Seguidament exposarem algunes conclusions extretes de la realització del projecte i finalment valorarem les possibles línies futures que continuarien el nostre treball.

6.1 Valoració dels objectius

A continuació valorem un per un els objectius que ens vam plantejar:

Entendre les DTN

Aquest era l'objectiu més prioritari del nostre projecte, ja que si no el complíem era impossible seguir endavant. El vam va assolir abans de començar amb la fase d'anàlisi de requisits i vam fer servir els coneixements adquirits per dissenyar i implementar les millores que hem dut a terme.

Estudi de JADE i de l'Add-on IPMS

Com passa amb el primer objectiu, entendre JADE i l'IPMS era un requisit indispensable. De nou, el fet que hàgim pogut dur a terme el nostre projecte demostra que hem estat capaços d'assolir aquest objectiu.

Disseny de l'encaminament d'agents independent de l'aplicació

Durant la realització del projecte hem dissenyat millores per l'IPMS que permeten als agents decidir el camí que volen seguir des del seu punt d'origen fins al seu destí. Aquest disseny s'ha implementat i provat amb èxit en un entorn real Per tant, podem dir que hem aconseguit assolir aquest objectiu.

Disseny de prioritats pels agents

Aquest objectiu es va plantejar inicialment com a principal del projecte, però finalment

vam decidir centrar-nos en l'encaminament dels agents. Tot i així hem dissenyat i implementat de manera senzilla una cua amb prioritats, substituint-la per la cua FIFO que hi havia inicialment a l'IPMS. Per tant, podem dir que, en la mesura que es va establir finalment, hem assolit aquest objectiu.

Implementar i validar el nostre treball en un model d'escenari real

Després de tota la fase d'anàlisi i disseny hem aconseguit implementar les millores, afegint-les a l'IPMS. A més hem creat un petit entorn de proves on hem vist que els resultats dels nostres experiments eren positius i que el que havíem fet funcionava. Per aquests motius, podem afirmar que l'objectiu ha estat assolit.

Nous objectius

A part dels objectius anteriors durant el projecte també ens vam plantejar els següents punts:

- Després de dissenyar la part de l'encaminament dels agents vam decidir posar-hi un afegit més, fent que els agents poguessin transportar informació rellevant per decidir quin camí seguir fins al seu destí. Aquesta part també ha estat provada i s'ha comprovat que funciona com s'esperava. Tot i així, hi ha alguns aspectes que no s'han tingut en compte i sense els quals l'intercanvi d'informació no podria arribar a funcionar correctament en un entorn real. Parlarem d'aquests aspectes a l'apartat de línies de continuïtat.

- En un moment donat del projecte ens vam plantejar l'objectiu d'intentar estalviar el màxim possible d'energia i recursos en l'encaminament dels agents. Com a solució a aquesta proposta vam afegir al disseny un nou mètode pels agents que ens permetia saber quin camí volien seguir sense despertar-los. El fet que els agents no es despertin fa que gastem menys recursos i per tant també podem afirmar que hem assolit aquest últim objectiu.

6.2 Conclusions generals

De manera més general podem afirmar que els agents mòbils poden servir per a transportar informació a través de xarxes DTN. El fet que puguin fer servir informació tant de l'aplicació per la que treballen, com de capes inferiors per viatjar per les DTN pot ajudar a que la informació viatgi de manera més ràpida i efectiva a través d'aquestes xarxes.

A més, com que tenen la capacitat d'adaptar-se a entorns molt diferents, només necessitem un entorn on poder executar JADE, poden ser utilitzats en situacions de molts tipus amb resultats igualment satisfactoris.

Com a última conclusió podríem remarcar que, encara que no puguem afirmar que el futur de les DTN passarà pels agents mòbils, la seva naturalesa i les seves grans possibilitats els presenten com a una bona solució per gestionar aquest tipus de xarxes.

6.3 Línies de continuïtat

En els apartats anteriors ja hem anat comentant alguns aspectes del nostre treball que es podrien millorar. Aquí les exposarem i en proposarem possibles solucions.

La primera línia de continuïtat estaria centrada en afegir una forma de donar prioritats automàticament als nostres agents. Nosaltres només hem afegit una solució per fer servir les prioritats dels agents un cop aquestes ja han estat assignades, però el punt realment interessant és com assignar-les. Donat que hi ha infinites característiques que poden definir un agent, caldria trobar una manera de classificar-los i assignar-los propietats per poder assignar prioritats coherentment. Probablement també ens basaríem en la situació al nostre voltant, com per exemple el domini de l'aplicació per la que treballa l'agent, per prendre aquesta decisió.

La segona línia de continuïtat implicaria millorar el sistema d'intercanvi d'informació mitjançant agents. En el nostre projecte anem afegint i modificant informació de les taules de les plataformes, però no tenim en compte, per exemple, que aquesta pot caducar. Si cap agent porta informació nova durant un període de temps llarg, la que es quedi obsoleta s'hauria d'eliminar o actualitzar d'alguna manera. D'altra banda tampoc es té en compte que els dispositius on es guarda la informació poden quedar-se sense memòria si anem guardant tota la que ens arriba. De nou s'hauria de buscar una forma de saber quina informació és important guardar i quina no.

Per últim, s'hauria d'evitar que els agents es despertessin a cada plataforma per la que passen. Actualment ha de ser el programador qui comprovi que l'agent ha arribat al seu destí final i si no hi ha arribat, fer que segueixi saltant. Això implica una pèrdua de temps i d'energia que es contraproduent per la xarxa. S'hauria de modificar el codi de la plataforma per que els agents només es despertessin quan es trobessin al seu destí final.

Bibliografia

[1] RFC 1180, TCP/IP Tutorial, T. Socolofsky, C. Kale, Gener 1991
<http://tools.ietf.org/html/rfc1180>

[2] RFC 4838, Delay-Tolerant Networking Architecture, V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, Abril 2007
<http://tools.ietf.org/html/rfc4838>

[3] DTN: A tutorial, Forrest Warthman, Maig 2003
<http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf>

[4] Shoham, Y. 1997. An Overview of Agent-oriented Programming. In Software Agents, ed J. M. Bradshaw. Menlo Park, Calif.: AAAI Press.

[5] TELECOM ITALIA, Java Agent DEvelopment Framework (JADE)
<http://jade.tilab.com/>

[6] J. Cucurull, Efficient Mobility and Interoperability of Software Agents, Phdthesis, Universitat Autònoma de Barcelona, 2008

[7] RFC 5050, Bundle Protocol Specification (Experimental), K. Scott, S. Burleigh, Novembre 2007
<http://tools.ietf.org/html/rfc5050>

[8] Seguridad en la planifiación de agentes móviles en redes DTN, C. Borrego, S. Robles, Actes de la XI Reunión Española de Criptologia y Seguridad de la Información. In press. 2010

[9] Foundations of Intelligent Physical Agents (FIPA)
<http://www.fipa.org>

[10] Agent Communication Language Specifications (ACL)
<http://www.fipa.org/repository/aclspecs.html>

[11] J. Cucurull, Inter-Platform Mobility Service (IPMS)
<http://sourceforge.net/projects/jipms/files/>

[12] Self-protected Mobile Agents; Ametller, Robles, Ortega; ACM, 2004

[13] Gantt Project
<http://www.ganttproject.biz/>

Firmat: David Sanchez Garcia

Bellaterra, Juny de 2010

Resum

En aquest projecte es proposa i s'implementa una nova forma de crear xarxes DTN (*Delay-Tolerant Networks*) mitjançant agents mòbils. Aquestes xarxes tenen la peculiaritat de ser tolerants a endarreriments i interrupcions, podent ser utilitzades en entorns on les xarxes actuals no es poden aplicar. Hem dissenyat mecanismes que permeten prendre decisions d'encaminament a nivell d'aplicació i mecanismes de priorització d'agents mitjançant informació d'alt nivell. Aquests mecanismes milloren les DTN fent-les més flexibles i efectives.

Resumen

En este proyecto se propone y se implementa una nueva forma de crear redes DTN (*Delay-Tolerant Networks*) usando agentes móviles. Estas redes tienen la peculiaridad de ser tolerantes a interrupciones y retrasos, pudiendo ser usadas en entornos donde las redes actuales no son aplicables. Hemos diseñado mecanismos que permiten tomar decisiones de encaminamiento a nivel de aplicación y mecanismos de priorización de agentes usando información de alto nivel. Estos mecanismos mejoran las DTN haciéndolas más flexibles y efectivas.

Abstract

In this project we propose and implement a new way to create Delay-Tolerant Networks (DTN) using mobile agents. These networks have the singularity of being tolerant to delays and disruptions, making them usable on environments where the current networks are not applicable. We have designed a routing mechanism able to make decisions at the application level, and an agent prioritization mechanism which uses high level information. These mechanisms improve Delay-Tolerant Networks by making them more flexible and efficient.