DESENVOLUPAMENT EN NETWORK SIMULATOR 2. PROTOCOL DE MIGRACIÓ PER SIMULAR AGENTS MÒBILS

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria en Informàtica presentat per Cristian Stefan Tanas i dirigit per Rubén Serrano Núñez.

Bellaterra, Juny de 2010

El firmant, Rubén Serrano Núñez, 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 Cristian Stefan Tanas

Bellaterra, Juny de 2010

Firmat: Rubén Serrano Núñez

A la persona que, per sempre, ha marcat la meva vida.

Agraïments

Thomas Jefferson escrivia en la Declaració d'Independència que un dels drets fonamentals de les persones és *"the pursuit of happiness"*. Però per què *perseguir* la felicitat i no *ser* feliços? Perquè l'important no és la destinació final sinó el camí que hem seguit. Cada pas que donem en la vida determina el següent, i totes les persones que he conegut, totes les experiències que he viscut han deixat la seva particular marca en la meva personalitat. Per això, vull agrair a totes aquelles persones que d'alguna forma o altra han influenciat la meva vida.

En primer lloc vull agrair el suport i recolzament del meu director de projecte, Rubén Serrano. La seva constant ajuda i supervisió han fet possible que avui estigui escrivint aquestes línies.

També vull agrair als meus pares pel seu suport i consells que m'han donat, i els vull agrair tots els esforços i sacrificis que han fet per a que jo pugui disfrutar de la millor vida possible. Als meus avis, per compartir amb mi tota la seva experiència, per cuidar-me i per haver influenciat tant en la meva evolució com a persona. Gràcies a la meva avia per les nits mirant les estrelles! A la Eli, que sempre ha estat al meu costat i qui ha revolucionat la meva vida. Gràcies per la paciència i comprensió per tants dies dedicats exclusivament a aquest projecte!

I no per últim, David, Victor, Dani, Ivan, Marcos, Carlos, Leo, Tomás, Jesús, Julien, Bàrbara, Alba, Maija, Júdit, Elisa agrair-vos tants moments bons i no tan bons que hem viscut junts. Tots vosaltres heu estat presents en una nova etapa de la meva vida i mai us oblidaré!

Us demano disculpes si m'he oblidat de mencionar algú, seran coses de l'edat.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

Els agents mòbils són entitats software que disposen de l'habilitat d'aturar i reprendre la seva execució en diferents llocs de la xarxa per dur a terme una series de tasques. Avui en dia, el desplegament dels agents mòbils és limitat, però noves aplicacions s'estan obrint pas constantment. Aquesta tecnologia ofereix solucions prometedores a problemes com la pèrdua de connexió entre nodes o grans endarreriments en la connexió. La capacitat dels agents mòbils de residir i migrar entre diferents nodes de una xarxa els fa especialment interessants, perquè ens permet optimitzar els recursos de la xarxa i evitar congestionar-la.

Per exemple, Vieira-Marques et al. [VMRC+06] proposen un esquema per recopilar i integrar dades mèdiques d'un pacient, distribuïdes en varis hospitals, utilitzant els agents mòbils. Martí et al. [MR+08] proposen un esquema pel triatge de víctimes en situacions d'emergència basat en els agents mòbils. L'informació de les víctimes s'encapsula en un agent mòbil i es enviada al centre d'operacions per una ràpida i eficient distribució dels recursos mèdics. Aquest sistema es coneix com MAETTS (*Mobile Agent Electronic Triage Tag System*). Algunes de les característiques més importants d'aquests tipus d'entorns son la connectivitat intermitent, latència variable, patrons ambigus de moviment dels nodes. En conseqüència un agent pot residir en un node durant un llarg període de temps, mentre que la finestra de connexió pot ser molt petita.

Tot aquest ampli ventall d'aplicacions han de demostar la seva viabilitat, la

possiblitat de ser implantats en el món real. La falta d'un sistema de proves per poder demostrar la viabilitat d'una aplicació o simular el seu funcionament ens obliga a passar per una etapa de desenvolupament, de construcció de l'aplicació final. Això dona lloc a una gran varietat d'aplicacions que després poden resultar no útils, amb la conseqüent pèrdua de temps.

Hi ha multitud de simuladors de xarxa disponibles en el mercat, un dels quals és el NS2. És el simulador més utilitzat en l'àmbit acadèmic i permet simular la gran majoria de protocols i tipologies de xarxa existents avui en dia. És tracta d'un projecte de codi lliure i per tant, els usuaris poden aportar el seu coneixement pel desenvolupament de nous protocols i nous sistemes.

En aquest projecte pretenem afegir els agents mòbils al NS2 per poder fer simulacions i no haver de construir l'aplicació final.

1.1 Motivació

Una de les qüestions a la que s'enfrenten els agents mòbils és demostrar la seva viabilitat en sistemes reals. Necessitem demostradors o simuladors que ens ajudin a predir quin serà l'impacte de l'implantació d'un aplicació basada en agents mòbils, que no es produiran situacions de col·lapse o exhauriment dels recursos disponibles.

Demostrar la viabilitat de les aplicacions basades en agents mòbils en entorns heterogenis i complexos, com ara les xarxes MANET o les DTN, ens pot portar a molts entrebancs degut a la complexitat que suposa. El cost en temps o diners per construir sistemes de test (*test-bed*) o sistemes reals per dur a terme un anàlisi de rendiment en molts cassos pot resultar inviable. I més si tenim en consideració diferents aplicacions amb diferents paràmetres de configuració i diferents indicadors de rendiment. Es per això que necessitem construir un model de simulació del comportament general dels agents mòbils i estudiar-lo com un substitut del sistema real.

1.2 Objectius

L'objectiu base d'aquest projecte és construir un framework per la simulació del comportament general dels agents mòbils ampliant l'actual simulador de xarxa NS2. Per fer això, en primer lloc hem de fer un estudi de la tecnologia d'agents mòbils i el simulador de xarxa NS2. Hem de coneixer quins són els protocols de migració d'agents mòbils, com funciona el NS2 i com es pot afegir un nou protocol al simulador.

L'element principal dels agents mòbils és la seva capacitat per desplaçar-se entre els diferents nodes d'una xarxa. Per tant, en segon lloc hem de desenvolupar - o millor dit implementar - un protocol de migració per agents mòbils i afergir-lo al NS2.

L'objectiu que finalment ens interessa assolir es poder analitzar quin és el funcionament dels agents mòbils en una xarxa donada. Per fer això hem de dissenyar i desenvolupar un framework que ens permeti simular el seu comportament. Hem de poder veure si l'agent té la capacitat d'adaptar-se a una determinada topologia de xarxa i dur a terme amb èxit les seves tasques.

Finalment, per veure el funcionament del model implementat i tenir una prova de la seva correcta implementació, hem de simular el comportament d'un tipus d'agent concret en els escenaris definits, fent servir el protocol de transferència implementat al NS2.

1.3 Estructura de la memòria

En els següents paràgrafs donarem una visió general sobre el contingut dels capítols i apèndixs que composen aquesta memòria.

El capítol 2 presenta una breu introducció a l'estat de l'art d'aquest projecte. En primer lloc, farem una petita introducció a la tecnologia d'agents mòbils i presentarem les seves característiques més importants. Després farem una introducció al marc de desenvolupament d'aquest projecte: el simulador de xarxa NS2.

El capítol 3 exposa un estudi del projecte. Analitzarem els requisits funcionals i no funcionals, presentarem un breu estudi de viabilitat i finalment veurem la planificació temporal inicial.

El capítol 4 mostra la proposta de disseny que compleix amb els requisits i objectius del projecte. Partirem d'una visió general del sistema per anar centrant l'atenció en els elements principals.

El capítol 5 explica quina ha estat l'implementació del disseny presentat en el capítol anterior. En primer lloc, presentarem els detalls d'implementació de cada un dels elements principals del disseny, i després explicarem les diverses proves que s'han realitzat per comprovar el correcte funcionament de l'implementació. Presentarem diferents escenaris de simulació.

El capítol 6 serà un resum de les conclusions obtingudes a partir dels objectius presentats anteriorment, i exposarà algunes possibles línies de treball futures.

Finalment, l'apèndix A contindrà els detalls tècnics derivats de l'integració en el NS2 del sistema desenvolupat, i l'apèndix B contindrà dos exemples dels resultats obtinguts en les simulacions.

Capítol 2

Els agents mòbils i el Network Simulator 2

En aquest capítol descriurem el context en el qual es troba aquest projecte per tal de comprendre millor el seu desenvolupament. Explicarem en primer lloc que són els agents mòbils, aspectes relacionats amb el funcionament d'aquests i finalment explicarem el marc de desenvolupament d'aquest projecte: el simulador de xarxa Network Simulator 2.

2.1 Agents mòbils

S'entén per agent un component software autònom, que actua en nom d'algú, s'executa en entorns controlats anomenats plataformes, és capaç d'interactuar amb el seu entorn, percebre canvis i l'estat d'aquests, i reaccionar davant d'aquests. Si aquest agent té la capacitat de desplaçar-se, aleshores s'ha de parlar d'agents mòbils. Aquesta tecnologia va donar els seus primers passos amb la publicació de l'artícle de Jim White [WMA96] el 1996. La definició exacta, però varia d'un autor a l'altre, tot i que hi ha certes característiques que són comunes per tots els autors:

- **Persistència**: el codi que s'executa en cada moment és decidit pel propi

agent en funció de l'objectiu concret que tingui. L'agent presenta un comportament dirigit a complir l'objectiu.

- **Autonomia**: l'agent és capaç de prendre decisions sobre les seves accions o estat sense la intervenció directa d'una persona.

- **Habilitat social**: els agents són capaços d'interactuar amb altres agents per tal d'assolir els seus objectius.

- **Reactivitat**: els agents perceben l'entorn o context en el que es troben reaccionant en conseqüència als canvis que es produeixen.

- **Mobilitat**: l'agent és capaç de suspendre la seva execució, moure's entre diferents plataformes i reprendre la seva execució.

Com hem esmentat anteriorment, una de les propietats dels agents mòbils és la mobilitat. Això s'aconsegueix parant l'execució de l'agent en la plataforma origen, copiant el codi font juntament amb l'estat d'execució d'aquest en la plataforma destí i reprenent l'execució del codi de l'agent, tot comprovant que sigui correcte. Una vegada fet això es destrueix la còpia que ha quedat en la plataforma origen. Tots aquests passos es coneixen sota el nom de migració. Els passos presentats aquí es el que s'anomena també migració inter-plataforma. També podem distingir entre:

- **Migració forta**: en aquest tipus de migració es copien tant el codi i les dades com l'estat de l'agent, facilitant la implementació al programador, ja que s'assegura que el codi se seguirà executant des de la línia de codi següent a la ordre de migració en arribar a la plataforma destí.

- **Migració dèbil**: en aquest tipus de migració només es còpia la part de codi i dades de l'agent, provocant que el codi s'executi des de l'inici un cop s'hagi realitzat la migració. Per tant, el programador s'ha d'encarregar d'emmagatzemar variables de control específiques per saber en quin punt del codi es troba l'execució de l'agent.

També s'ha parlat de plataformes. Aquestes, també anomenades agències són els entorns d'execució dels agents que els proporcionen tot un conjunt de serveis com ara comunicació o mobilitat.

2.2 Especificacions i estàndards

La FIPA (*Foundation for Intelligent Physical Agents*) és un comitè de l'IEEE que des del 2005 promou la tecnologia basada en agents y la utilització dels seus estàndards per la interoperabilitat amb d'altres tecnologies.

Hi ha molts estàndards definits per la FIPA per al desenvolupament d'agents i els seus sistemes, tal i com es pot veure a [FIPASpec]. Una plataforma basada en els estàndards FIPA inclou, entre d'altres:

- **Agents**: les entitats software que actuen en nom d'algú.
- **Agent Management Service (AMS)**: és un servei que s'encarrega de registrar i mantenir el control de tots els agents que hi ha a la plataforma. Es pot veure com un servei de pàgines blanques de la plataforma.
- **Directory Facilitator (DF)**: és un servei que s'encarrega de registrar tots els serveis oferts pels agents d'una plataforma. Es pot veure com un servei de pàgines grogues i és opcional (pot ser-hi o no).

La comunicació entre els agents es duu a terme mitjançant missatges ACL, també definits per FIPA, que estandarditza què han de contenir i com han de ser. Aquests missatges es transmetran mitjançant el MTS (*Message Transport Service*), un servei que proporciona la plataforma per l'intercanvi de missatges entre els agents. Si els agents es troben en plataformes diferents s'utilitza també el MTP (*Message Transport Protocol*), que encapsula els missatges ACL en un protocol de la capa d'aplicació. Dit amb altres paraules, afegeix un sobre indicant la informació de transport (figura 2.1).

Des del grup de recerca SeNDA del dEIC (Departament d'Enginyeria de la Informació i de les Comunicacions) de la UAB (Universitat Autònoma de Barcelona) es va iniciar un projecte per a la creació d'estàndards de mobilitat seguint les

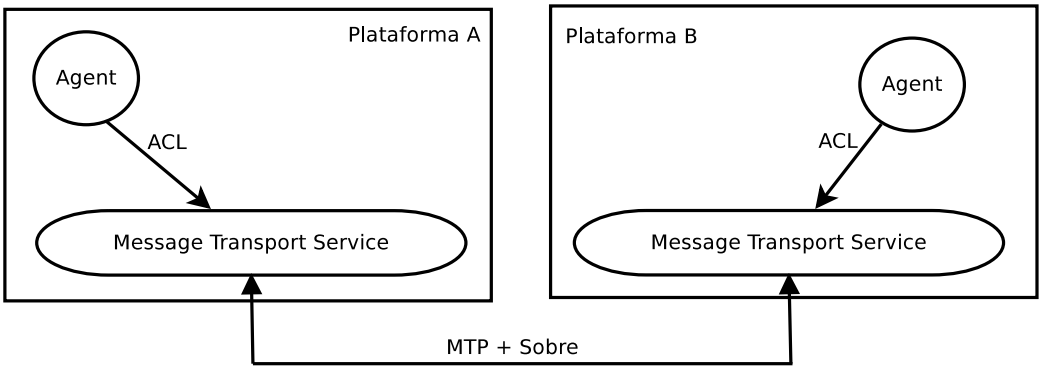

Figura 2.1: Message Transport Protocol

especificacions FIPA que va finalitzar amb la proposta de l'IPMA (*Inter-Platform Mobility Architecture*). Aquesta proposta defineix una nova arquitectura per a la migració d'agents que pretén crear un framework de migració flexible capaç de suportar qualsevol conjunt de protocols de migració escollits lliurement per cada agent.

El procés de migració es divideix en 5 passos contingut dintre del MMP (*Main Migration Protocol*) (figura 2.2): Pre-Transfer, Transfer, Post-Transfer, Agent Registration i Agent Power Up. En els primers dos passos s'utilitzen un conjunt obert de protocols especificats en el primer missatge mentres que els últims dos utilitzen protocols prefixats.

El **MMP** s'encarrega d'iniciar i gestionar el protocol de migració així com tota la resta de protocols seguint l'estàndard *IEEE-FIPA Request Interaction Protocol*. El primer missatge conté, entre d'altres elements, el conjunt de protocols a utilitzar en els tres primers passos, així com altres requisits de l'agent.

Les fases **Pre-Transfer, Transfer, Post-Transfer** tenen com a funcionalitat transferir el codi, dades i/o estat de l'agent i altres operacions opcionals a dur a terme abans i després de la transferència. Aquesta funcionalitat es pot implementar per diferents protocols seleccionats en temps d'execució per part de l'agent.

Les fases **Agent Registration, Agent Power Up** tenen com a funcionalitat registrar i re-emprendre l'execució de l'agent en la plataforma destí. Quan l'agent es registra en la plataforma destí, s'elimina immediatament la còpia que hi ha a la plataforma origen.

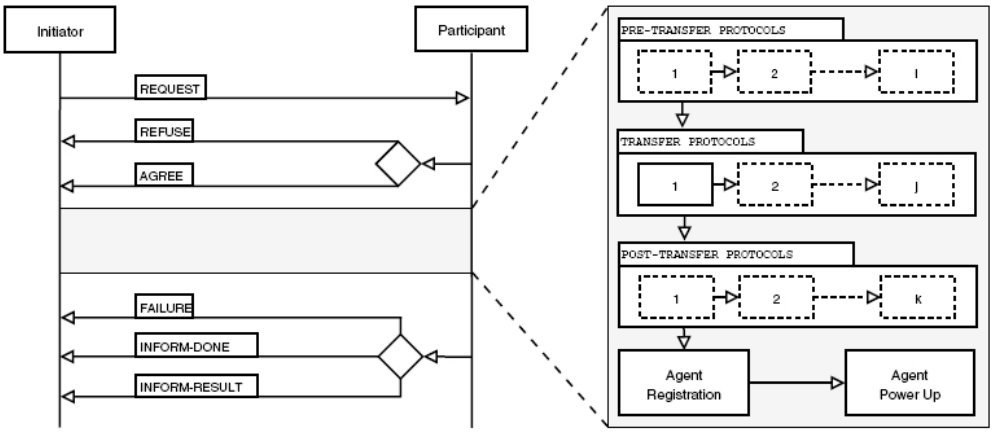

Figura 2.2: Main Migration Protocol

Font: [FIPA-IPMA]

En aquest projecte es pretén implementar un protocol inclòs en la fase de *Transfer* i afegir-lo dins del simulador de xarxa NS2.

2.3 Network Simulator 2

El Network Simulator 2 és possiblement el simulador de xarxa més utilitzat dintre del món acadèmic. El seu desenvolupament va començar en el *Information Science Institute* de la *University of Southern California* i avui en dia forma part del projecte VINT, una col·laboració entre investigadors de la *UC Berkeley, LBL, USC/ISI* i *Xerox PARC*.

És un simulador orientat a objectes escrit en C++ amb un intèrpret oTCL com a interfície d'entrada i és un simulador orientat a esdeveniments discrets. TCL o Tool Command Language és un llenguatge de scripting desenvolupat en els anys 80 per John Ousterhout treballant en la *UC Berkeley* i oTCL no és més que una extensió del TCL orientat a objectes. L'ús de dos paradigmes de programació es justifica per dues característiques importants que ha de proporcionar el simulador. Per una banda, la simulació detallada de protocols necessita un llenguatge de programació que sigui capaç de manipular de manera eficient bytes i paquets, implementar algoritmes que funcionin sobre grans volums de dades, i *velocitat* en

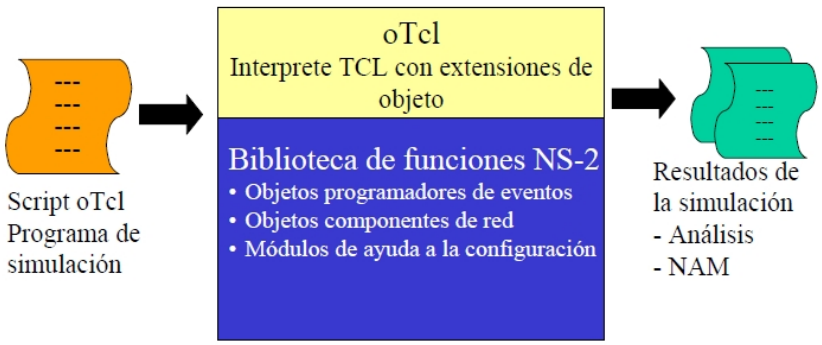

Figura 2.3: Arquitectura del NS2

Font: [MUN]

quant a temps d'execució. Per altra banda, l'anàlisi de xarxes, molts cops implica haver de canviar paràmetres de configuració o avaluar multitud d'escenaris on és important el temps d'*iteració* (canviar el model i executar de nou). El llenguatge C++ cobreix la primera necessitat, sent un llenguatge adequat per la implementació de protocols, mentre que oTCL cobreix la necessitat de canviar fàcilment la configuració d'una simulació. A la figura 2.3 es pot veure una vista bastant simplificada de l'arquitectura del NS2.

L'entrada de l'aplicació és un script en oTCL que especifica el que l'usuari vol simular. És l'únic *input* que l'usuari necessita proporcionar al simulador, sent la resta d'accions processament intern del NS2. Al finalitzar la simulació, els resultats es guarden en un fitxer d'extensió *.nam* i en un fitxer d'extensió *.tr*, que conté una completa descripció de la simulació, on cada línia descriu els paquets enviats, rebuts, encolats, extrets de la cola, etc. Malgrat que els arxius amb extensió *.tr* presenten la informació d'una forma llegible per l'usuari, es fa molt incòmode llegir, analitzar alguns d'aquests arxius o tenir una imatge clara de què és el que està passant en la simulació per la gran quantitat de dades. Es per aquest motiu que la visualització dels resultats de la simulació es realitza mitjançant un mòdul extern al NS2 anomenat NAM (Network AniMator) que és una interfície que mostra de manera gràfica els resultats obtinguts.

Com hem dit, la dualitat C++ i oTCL és una de les principals característiques

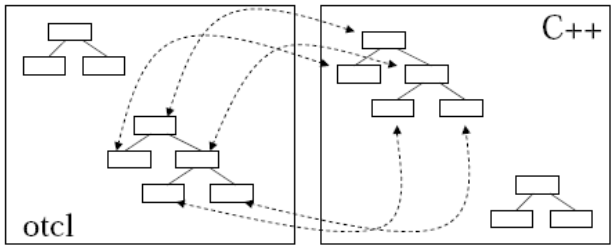

Figura 2.4: Dualitat C++ - oTCL

Font: [FV09]

del NS2. El simulador soporta una jerarquia de classes en C++ (també coneguda com jerarquia compilada) i una altra jerarquia de classes similar en oTCL (també coneguda com jerarquia interpretada). Des de la perspectiva de l'usuari, hi ha una correspondència un-a-un entre una classe en la jerarquia interpretada i una classe en la jerarquia compilada (figura 2.4).

Les principals classes responsables de mantenir aquest vincle entre C++ i oTCL són les següents:

- **Classe Tcl**: encapsula l'actual instància de l'intèrpret oTCL i proporciona mètodes per accedir i comunicar-se amb l'intèrpret. Proporciona mètodes per invocar procediments oTCL mitjançant l'intèrpret, rebre o enviar resultats a l'intèrpret, guardar i buscar objectes, etc.

- **Classe TclObject**: és la classe base per la majoria de les classes que podem trobar tant en la jerarquia compilada com en la jerarquia interpretada. Qualsevol objecte en la classe TclObject es creat per l'usuari des de l'intèrpret i acte seguit es crea un objecte *ombra* en la jerarquia compilada. La classe TclClass és la que ens proporciona aquesta correspondència.

- **Classe TclClass**: és una classe compilada virtual pura i proporciona dues funcions: construir la jerarquia interpretada per reflectir la jerarquia compilada i proporcionar mètodes per instànciar nous objectes TclObject.

2.4 Agents mòbils en NS2

Network Simulator 2 és un simulador molt comú, accessible de manera pública i que soporta la simulació de una gran varietat de protocols existents. Ens proporciona una molt bona infraestructura per afegir nous protocols i també ens ofereix la possibilitat d'estudiar l'interacció de protocols a gran escala en un entorn completament controlat. L'abstracció fonamental que l'arquitectura software del NS2 ens proporciona és la *composició programable*, es a dir, la configuració de la simulació es veu més com un programa que una configuració estàtica.

NS2 també té cert inconvenients. És un sistema gran en el qual la curva d'aprenentatge té una pendent molt pronunciada. La dualitat C++ - oTCL utilitzada a l'hora de implementar el NS2 es converteix, molts cops, en una barrera per als desenvolupadors. Però, degut a que els investigadors cada vegada prenen més consciència d'aquest fet, juntament amb una varietat d'eines, com ara, tutorials, manuals i llistes d'usuaris estan millorant aquesta situació.

El relació als agents mòbils, el NS2 ens proporciona una estructura per capes, seguint el model TCP/IP, i implementa tots els protocols necessaris. D'aquesta forma, simplement afegint una capa en aquesta estructura podem simular fàcilment el comportament dels agents mòbils. A més a més, el llenguatge de programació C++ és força conegut i el llenguatge oTCL és un llenguatge molt descriptiu i de fàcil utilització.

2.5 Conclusions

En aquest capítol hem vist que un agent mòbil era una entitat software independent capaç de desplaçar-se entre varies plataformes. Hem vist també quines eren les seves propietats més interessants i els tipus de migració. Així mateix, hem vist com els agents utilitzen *Message Transport Protocols* dintre de l'arquitectura IPMA per migrar. Finalment, hem vist les característiques més importants del Network Simulator 2, i com la seva estructura i protocols ja implementats ens seran de gran ajuda per tal d'implementar el nostre sistema.

Capítol 3

Anàlisi de Requeriments

En aquest capítol veurem un anàlisi dels requisits necessaris pel nostre projecte, realitzat seguint les especificacions IEEE indicades al document STD 830-1998 [IEEE-SRS]. Anirem veient tant requisits funcionals com no funcionals a través dels diferents apartats, així com també un estudi de viabilitat. Finalment, presentarem la planificació temporal prevista per les diferents etapes del projecte.

3.1 Descripció global

Volem un sistema que ens permeti simular la migració d'agents mòbils dins del NS2. Els agents han de poder saltar entre diverses plataformes d'acord amb el seu comportament. El sistema ens ha de permetre crear agents i simular el seu comportament en un escenari ben definit. Per fer això necessitarem definir tres elements: agent, plataforma i un protocol per la migració dels agents.

El protocol que utilitzarem és el RESTTP (*REST Transfer Protocol*), que com el seu nom indica, combina la tecnologia REST (*Representational State Transfer*) [FT02] amb missatges ACL. REST és un conjunt coordinat de restriccions arquitecturals basat en el protocol HTTP. Aquest protocol és el més adequat en el nostre cas per què és un protocol pensat per agents que són totalment independents de la seva plataforma origen ja que sempre transporten tots els seus recursos (codi, dades i estat). També ens permet aprofitar la eficiència en el transport de dades

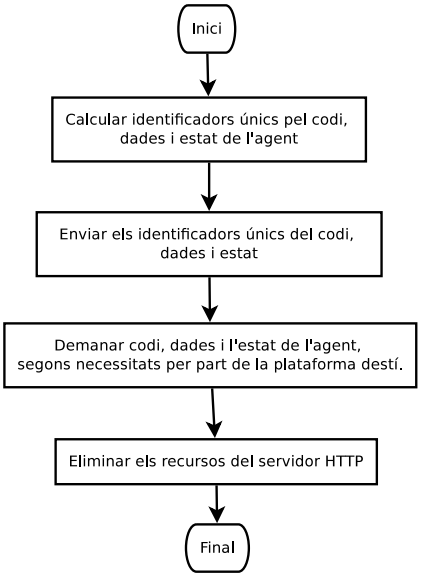

Figura 3.1: REST Transport Protocol

que ens proporciona el protocol HTTP.

El procés de migració de l'agent està compost per dues parts, una part en la qual s'intercanvien missatges ACL per establir varis paràmetres per la migració i una altra part on s'utilitzen peticions HTTP per tal de transferir els components de l'agent. Els recursos (codi, dades i estat) són enviats per un servidor HTTP sota noms únics i aleatoris només vàlids durant la transacció, per tal d'evitar l'intervenció d'elements externs.

El protocol comença enviant un missatge ACL amb els identificadors únics assignats a cada recurs i l'adreça del servidor HTTP on es troben (figura 3.1). Quan la plataforma destí rep aquest missatge demana, segons les necessitats, el codi, les dades i l'estat de l'agent utilitzant varies peticions HTTP (una per cada recurs).

Finalment, una vegada finalitzada la transferència, la plataforma destí envia un missatge ACL per informar de l'èxit de la transferència o bé un missatge d'error en cas contrari. En rebre aquest missatge, la plataforma origen elimina els recursos del servidor HTTP.

En la figura 3.2 podem veure un exemple de la seqüència de missatges ACL i

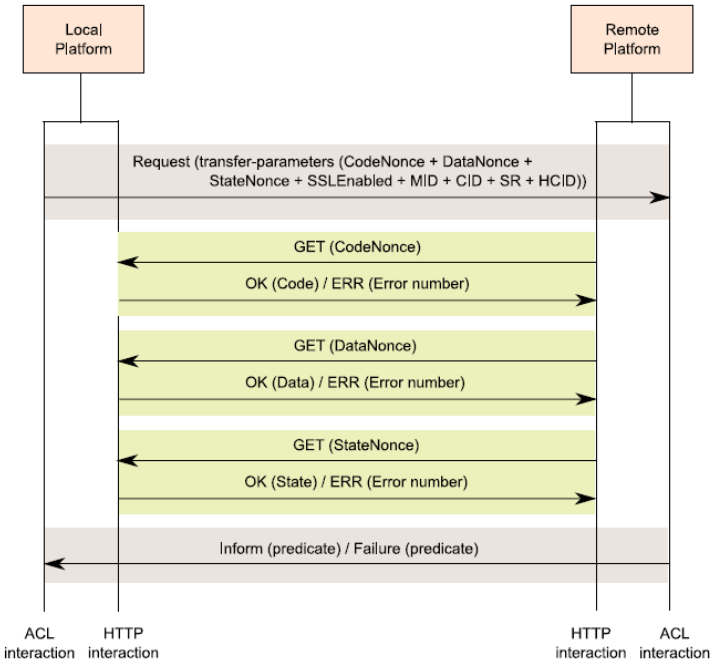

Figura 3.2: Diagrama de flux del RESTTP

Font: [CEM08]

HTTP que es dona en una migració típica.

3.2 Requisits Funcionals

A continuació veurem un recull dels requisits funcionals que el nostre model haurà de complir en quant a interfícies del sistema i d'usuari.

3.2.1 Interfícies del sistema

El sistema haurà de complir amb els estàndards FIPA per la migració d'agents entre sistemes compatibles: comunicació mitjançant missatges ACL, utilitzar protocols estandarditzats a FIPA, etc. Així mateix haurà de seguir l'arquitectura IPMA descrita en [FIPA-IPMA]. També ens de poder permetre migrar agents tant en xarxes cablejades com en xarxes sense fills, simular el comportament de l'agent en aquests entorns i controlar possibles errors en la migració (si la transferència

d'algun recurs falla hem d'anular el procés). Com hem dit, el sistema s'haurà d'integrar en el entorn de desenvolupament del NS2 i per tant, els llenguatges de programació seran C++ i oTCL. Haurem de seguir les normes d'estil i els estàndards de desenvolupament imposats pel NS2.

3.2.2 Interfícies d'usuari

L'interacció amb l'usuari es realitzarà mitjançant un script escrit en llenguatge oTCL. En aquest script s'hauran d'especificar tots els paràmetres de la simulació: topologia de la xarxa que es pretén simular, moviments específics dels nodes, plataformes, nombre d'agents, tamany i plataforma a la que pertanyen, instant de temps en el que començarà la migració, i altres paràmetres de configuració.

3.3 Requisits No Funcionals

A continuació veurem el software i hardware necessari per la realització d'aquest projecte.

3.3.1 Interfícies Software

Com que el sistema s'haurà d'afegir al NS2, necessitarem un Sistema Operatiu que tingui instalat un compilador del llenguatge C++ i el propi NS2. Tot i que està pensat per la seva utilització en plataformes GNU/Linux, es poden trobar emuladors de l'entorn per plataformes Windows.

3.3.2 Interfícies Hardware

Com que el NS2 no especifica cap restricció en quant al hardware, el programa podrà ser executat en qualsevol màquina que compleixi els requisits software indicats en la secció anterior.

3.4 Estudi de viabilitat

En aquest apartat farem un estudi de viabilitat en quatre camps: tècnic, operatiu, econòmic i legal i veurem els pros i contres del nostre sistema.

3.4.1 Viabilitat Tècnica

En aquest projecte es treballarà dintre de l'entorn de desenvolupament del NS2. Per tant, es desenvoluparà amb els llenguatges de programació C++ i oTCL. Mentre que el coneixement del llenguatge de programació C++ s'ha adquirit al llarg de la carrera, serà necessari un treball previ de documentació i formació en relació al llenguatge oTCL i les eines i entorn de desenvolupament del NS2. Per aquest projecte amb un equip de prestacions estàndard actuals ja en tindríem prou, ja que s'ha comprovat que els temps de processament són acceptables. El que considerem com a estàndard actual és un PC amb un processador AMD Turion 64 Single Core a 1.6Ghz i 1Gbyte de memòria RAM. Tot i així hi ha la possibilitat de treballar amb màquines més potents en el cas que es necessiti més potència de càlcul, ja que se'ns posa a disposició un laboratori d'ordinadors amb processadors Core 2 Duo a 3GHz i amb tot el programari i eines necessàries per al desenvolupament del projecte. En quant a la memòria que ha de tenir el maquinari, el NS2 no imposa cap requisit específic, però en la simulació de grans escenaris es important tenir una memòria el més gran possible. També s'ha comprovat que amb 1Gbyte de memòria RAM les prestacions del simulador són acceptables.

3.4.2 Viabilitat Operativa

Com ja hem esmentat anteriorment, es tracta d'afegir al NS2 un protocol amb una funcionalitat ben definida: la migració d'agents mòbils entre dos nodes. La implementació i integració dintre de l'entorn de desenvolupament del NS2 és efectivament viable ja que NS2 contempla la possibilitat d'afegir protocols de la capa d'aplicació, i de fet ja n'incorpora d'altres com el FTP o el HTTP. A més a més ja s'han realitzat proves utilitzant el protocol FTP i s'ha analitzat el seu codi i el que es pretén desenvolupar serà molt similar. Tot i així, la part de simulació

és una part que suposarà un cost computacional més elevat ja que la complexitat dels escenaris en els que s'haurà de comprovar el funcionament del protocol pot ser considerable. Però, com hem comentat en l'apartat anterior, es disposa de un laboratori d'ordinadors on es podran realitzar totes les proves necessàries per comprovar el correcte funcionament del protocol.

3.4.3 Viabilitat Econòmica

Tot el programari que s'utilitzarà per la realització del projecte és programari lliure. Tots els mòduls del NS2 estan sota llicència GPL i per tant, es poden utilitzar i modificar de forma gratuïta. No cal destinar pressupost per la compra de maquinaria nova ja que es podran utilitzar els recursos esmentats en la part de viabilitat tècnica i totes les proves es poden dur a terme amb els recursos hardware i software que hi ha disponibles, no suposant cap cost addicional en el projecte.

3.4.4 Viabilitat Legal

Com s'ha comentat en l'apartat anterior, l'entorn de desenvolupament del NS2 està sota llicència GPL. Això implica que s'ha de respectar l'autoria del codi i s'han d'especificar les modificacions respecte a l'original i també s'impedeix la venta del software. Totes les restriccions d'aquesta llicència no ens afecten, ja que l'objectiu no és comercialitzar el producte sinó poder avaluar viabilitat i rendiment de diferents projectes que treballen en entorns on s'utilitzen els agents mòbils, des d'un punt de vista acadèmic. A més, l'objectiu principal d'aquest projecte és la simulació de la migració dels agents, amb lo qual no ens haurem de preocupar de les dades que els agents puguin transportar i que poden estar protegides per lleis com per exemple la Llei Orgànica de Protecció de Dades de Caràcter Personal (LOPD).

Per tant, podem afirmar que és viable realitzar aquest projecte des de tots els punts de vista considerats.

3.5 Planificació temporal

El projecte s'ha dividit en 4 etapes. En una primera etapa de formació es preveu l'estudi de l'entorn i eines de desenvolupament que s'hauran d'utilitzar per la realització del projecte, així com l'estudi del protocol de migració dels agents mòbils. Un cop acabada aquesta fase ja es tindran els coneixements necessaris per realitzar l'anàlisi de requeriments que ens permetrà fer el disseny i codificació del protocol de migració en NS2. Posteriorment, s'iniciarà la fase de simulació i finalment la redacció de la memòria.

S'estima una dedicació temporal de 4 hores diàries durant un semestre més una part prèvia de formació i documentació sobre l'entorn i les eines de desenvolupament. Sense tenir en compte la fase de formació, es preveu una dedicació en hores de:

20h Anàlisi + 20h Disseny + 144h Implementació + 16h Simulació + 80h Memòria + 20h Presentació = **300 hores**

En la figura 3.3 es pot veure el diagrama de Gantt corresponent a la planificació inicial del projecte.

Tot i així, la planificació temporal resultant ha estat diferent a la inicial degut a les dificultats derivades del desenvolupament en l'entorn NS2. En el capítol 6.2 veurem com aquestes dificultats han afectat a la etapa de desenvolupament i quina ha estat la planificació temporal final.

3.6 Conclusions

En aquest capítol hem vist quins són els requisits, tant funcionals com no funcionals, que ha de complir el nostre sistema. Hem vist també una breu descripció del protocol RESTTP que ens permetrà fer una primera descripció de com serà la fase de disseny. Hem vist també que el projecte és viable des dels punts de vista tècnic, operatiu, econòmic i legal. Finalment, hem vist la planificació inicial del projecte. Estimem 300 hores per dur a terme aquest projecte.

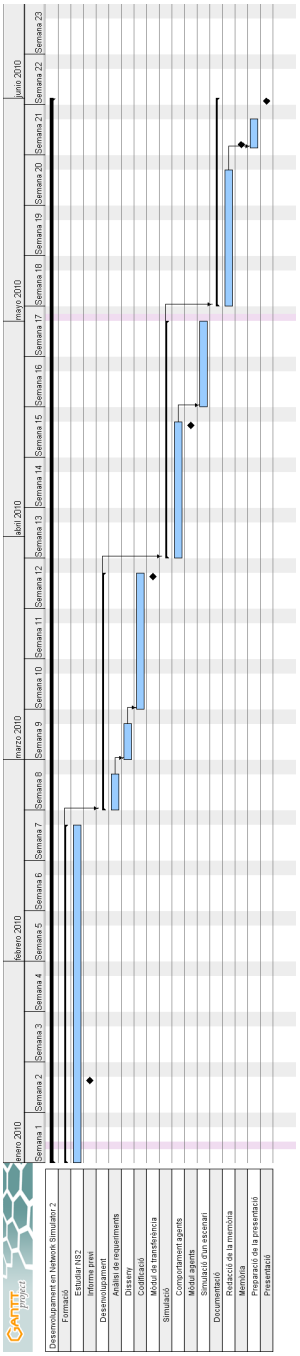

Figura 3.3: Planificació temporal inicial

Capítol 4

Disseny

En aquest capítol començarem definint l'esquema general del sistema que hem vist en el capítol d'anàlisi, i com s'integrarà dintre del NS2. Veurem com integrem el protocol RESTTP dintre del nostre sistema i dissenyem la plataforma i els agents, i també veurem tots el detalls tècnics derivats de l'utilització del sistema: com interacciona la plataforma amb els agents i la tolerància a fallades.

4.1 Disseny global

Tenim dos elements principals en el nostre disseny: l'agent i la plataforma. La plataforma serà responsable de crear l'agent i proporcionar tots els serveis que aquest necessiti, com ara migrar a un altre node, processar els agents que arriben, eliminar, etc. També utilitzarà el protocol RESTTP per la migració dels agents.

NS2 permet afegir nous protocols en diferents nivells d'abstracció seguint el model TCP/IP. En aquest cas podem afegir el protocol de migració en la capa d'aplicació o bé en la capa de transport. Per facilitar la posterior implementació i per què la plataforma ha de proporcionar un servei de transport dels recursos de l'agent hem decidit afegir el protocol de migració en la capa de transport. En NS2, els extrems finals d'una connexió punt a punt es defineixen com *Agents*. Utilitzarem aquesta notació durant la resta de la memòria per diferenciar-los dels agents mòbils com entitats software.

En el nostre cas, ens interessen més els passos del protocol de migració que les dades que s'han d'intercanviar els *Agents* per què el que volem és simular, i per tant podem assumir varies simplificacions:

1. Representarem la mida del codi, dades i estat d'execució de l'agent com paràmetres d'aquest, i assumirem que són coneguts. D'altra banda, la mida dels diferents recursos de l'agent no és constant, i per tant, podrem modificar els paràmetres definits per reflectir aquest fet.

2. No considerem la plataforma com una aplicació a sobre d'un *Agent* de la capa de transport sinó que la plataforma implementarà uns serveis mínims necessaris per la comunicació amb altres plataformes. En conseqüència, haurem d'implementar l'interfície *Agent* proporcionada en NS2.

3. Els agents mòbils es poden implementar de forma totalment independent al model TCP/IP. No tenim perquè afegir-los en una capa en concret d'aquest model.

En la figura 4.1 podem veure la jerarquia de classes escollida per implementar el nostre model. La classe que representa la plataforma (que en diem RESTTP_PlatformAgent) deriva de la classe Agent i per tant, el seu nom en la jerarquia de classes oTCL és *Agent/RESTTP_PlatformAgent*. En canvi, la classe que representa els agents mòbils (que en diem MobileAgent) no estarà accessible directament en la jerarquia.

4.2 Plataforma

La plataforma és la part central del nostre disseny i es pot dividir de manera lògica en dues parts: una part externa que s'encarregaria de gestionar la comunicació amb altres plataformes i la transferència dels agents i una part interna que s'encarregaria de gestionar als agents mòbils.

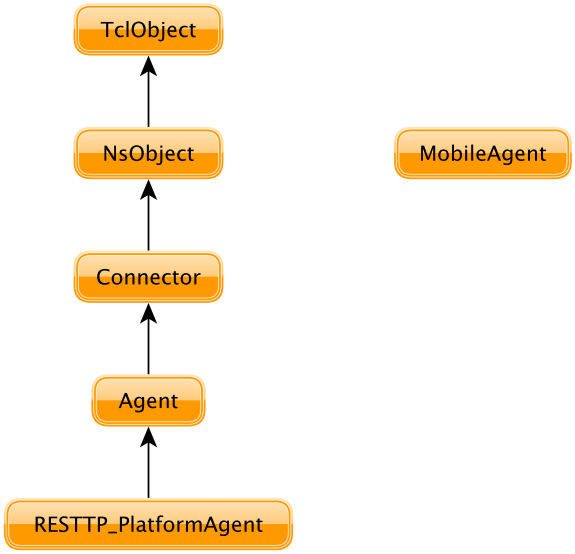

Figura 4.1: Jerarquia de classes pel model

4.2.1 Comunicació externa

Com hem vist en el capítol anterior, el protocol de migració necessita l'intercanvi de dos tipus de missatges: missatges ACL i peticions/respostes HTTP. En primer lloc haurem de definir quins són els camps necessaris per les seves capçaleres. En la taula 4.1 podem veure els camps continguts en la capçalera dels missatges ACL.

En primera instància vam pensar un esquema on s'utilitzi un temporitzador per tal d'enviar les peticions HTTP cada cert interval i buscar un model que ens caracteritzi els temps d'enviament i processament, però ens hem adonat que aquest plantejament no era correcte ja que en NS2, una característica dels enllaços és el *delay*, donat per les pròpies característiques de l'enllaç. Aquest delay fa que es produeixi un retard en l'enviament dels paquets per l'enllaç, i per tant no ens cal un temporitzador. A més a més, el temps de processament de les peticions, al tractar-se simplement d'una simulació el podem considerar despreciable. D'acord amb això, hem implementat un esquema que utilitza números de seqüència per l'enviament dels missatges HTTP. En la taula 4.2 podem veure la descripció dels

Camp	Descripció
com_type	Ens indica quin tipus d'acte comunicatiu s'està portant a terme entre dues plataformes (p.e. TRANSFER, INFORM, FAILURE, etc.)
info	Estructura amb dos camps: *type* i *id*. Ens indica el tipus d'agent a transferir i el seu identificador únic.
cid	Identificador únic assignat al codi de l'agent d'acord amb les especificacions del protocol RESTTP.
did	Identificador únic assignat a l'estat de l'agent d'acord amb les especificacions del protocol RESTTP.
sid	Identificador únic assignat a les dades de l'agent d'acord amb les especificacions del protocol RESTTP.

Taula 4.1: Capçalera dels missatges ACL

Camp	Descripció
req	Un descriptor que ens indica si es tracta d'una petició HTTP (1) o la resposta a una petició (0).
rcode	Si es tracta d'una resposta, necessitem saber si la petició s'ha processat correctament (rcode = 200) o si s'ha produït un error (rcode = codi de l'error).
rid	Identificador únic del recurs demanat.
seq	Número de seqüència del paquet HTTP.

Taula 4.2: Capçalera dels missatges HTTP

camps continguts en la capçalera HTTP.

4.2.2 Gestió dels agents

Per la gestió dels agents, per una banda hem de saber quins agents resideixen en la plataforma. La resposta ens la proporcionarà un registre dels agents mòbils disponibles, que podem veure com el servei de pàgines blanques. Aquest registre s'actualitzarà amb la creació d'un nou agent o amb la migració d'un agent ja existent.

Per altra banda, hem de saber quan un agent vol migrar. En aquest cas, serà la pròpia plataforma qui interrogarà als agents per saber si aquests volen migrar

o volen realitzar alguna altra operació. D'aquesta forma, serà la plataforma qui assumirà el rol principal i controlarà tot el procés de migració. Els agents contestaran basant-se en el seu comportament i en base a certs paràmetres que els proporcionarà la plataforma. Hem decidit fer-ho d'aquesta manera per què simplifiquem l'inserció del nostre model en la jerarquia de classes del NS2. Així, els agents mòbils es poden implementar de forma totalment independent, sent accessibles només des de la plataforma.

4.3 Agents i comportaments

Una de les característiques més interessants i que considerem la més important per la simulació de la migració dels agents, és el comportament de l'agent. Quina és la reacció de l'agent davant de certs estímuls. El comportament de l'agent base serà simple i consistirà en migrar sempre. Aquest comportament ens servirà per provar el funcionament bàsic del sistema.

4.3.1 Electronic Triage Tag Mobile Agent

Per tenir una prova de concepte i per demostrar que el nostre sistema realment funciona, hem decidit simular el comportament d'un agent mòbil per aplicacions mèdiques en entorns MANET[1], el Triage Tag mòbil (ETTMA) [MR+08]. Els Triage Tag són una eina que els serveis mèdics utilitzen en escenaris de catàstrofes per distribuir de manera efectiva els recursos limitats i proporcionar l'ajuda immediata a les víctimes que ho necessitin.

Hem decidit utilitzar aquest agent concret ja que és un agent que ha estat desenvolupat també dintre del grup SeNDA, ja que ens és més familiar i té un comportament molt ben definit, suficientment complex per proporcionar-nos una bona prova de concepte.

El comportament d'aquests agent mòbil es basa principalment en el TTR (*Time*

[1]MANET, Mobile Ad-hoc NETwork és una xarxa ad hoc de nodes que estan (o poden estar) en moviment, connectats per enllaços sense fils, la unió dels quals forma una topologia arbitraria que es configura a si mateixa

Columna	Descripció
Destinació	Adreça de la plataforma destí
Port	Port de la plataforma destí
TTR	TTR de la plataforma
Veí	Flag que ens indica si la plataforma està disponible en entrega directa o no

Taula 4.3: Taula de decisió del ETTMA

To Return) de les plataformes. El TTR representa el temps que li resta a la plataforma per arribar a la base de control. En un escenari d'emergència podem considerar les ambulàncies i els metges com plataformes i l'hospital de campanya com la base de control. Per tant, l'agent migrarà a una altra plataforma només si el TTR de la plataforma destí és menor. En cas contrari, l'agent no realitzarà cap acció.

Podem distingir dues parts en el comportament d'aquest agent: una part en la que el agent ha de prendre una decisió en base al TTR de les plataformes i una altra part de descobriment de les plataformes que estan al seu abast. Per la primera part utilitzarem una taula en la que guardarem informació sobre les plataformes que estan a l'abast amb el seu TTR. En la taula 4.3 podem veure el contingut d'aquesta taula.

Per la part de descobriment de plataformes hem definit el següent protocol:

1. **Fase d'anunciació d'existència**
   Cada X temps la plataforma A enviarà en BROADCAST i amb TTL = 1 el seu TTR. Només les plataformes que estiguin en entrega directa rebran el paquet. Aquestes al seu torn enviaran a la plataforma A el contingut de les seves taules de decisió. La plataforma A envia per UNICAST el seu TTR a totes les plataformes conegudes que no estan en entrega directa. En la figura 4.2 podem veure de forma gràfica aquests passos.

2. **Fase d'anunciació de TTR**
   Cada vegada que es canvia el TTR, la plataforma A envia per UNICAST a totes les plataformes de la seva taula de decisió el nou TTR (figura 4.3).

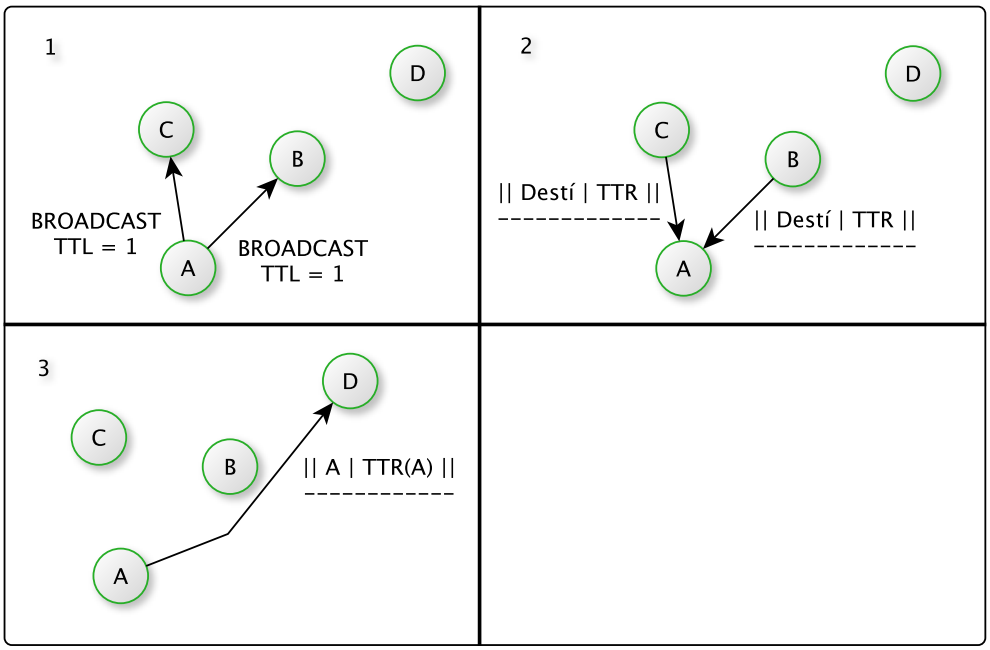

Figura 4.2: Fase de descobriment de plataformes

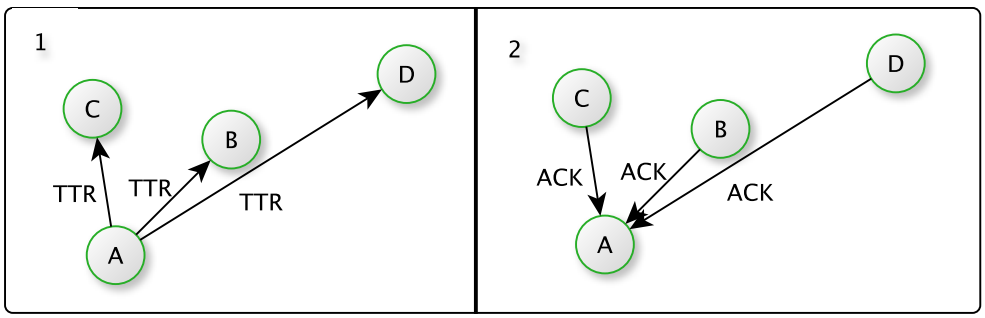

Figura 4.3: Fase d'anunciació de TTR

Camp	Descripció
type_of	Indica si es tracta d'un enviament en BROADCAST, un enviament en UNICAST o bé es tracta d'una resposta a un missatge d'anunciació d'existència.
ttr	En la fase d'anunciació d'existència, la plataforma origen emplena aquest camp amb el valor del seu TTR.
dtable	Contingut de la taula de decisió retornat en resposta a l'anunciament BROADCAST.

Taula 4.4: Capçalera dels missatges MTP_HELLO

3. **Fase d'eliminació d'entrada de la taula**
   Les plataformes que reben els missatges UNICAST han de respondre amb un ACK. Si una plataforma no rep un ACK d'alguna de les plataformes, l'esborrarà de la seva taula. Si s'intenta enviar un agent però el destí no és accessible també s'esborrarà la plataforma de la taula.

En aquest protocol ens apareixen nous tipus de missatges que les plataformes han d'intercanviar: missatges d'actualització de les taules de decisió (tant BROADCAST com UNICAST) i missatges de confirmació (ACK). En conseqüència, hem de crear dos tipus de paquets més i especificar la informació continguda en les seves capçaleres. Els missatges d'actualització els anomenarem MTP_HELLO i podem veure els camps de la capçalera en la taula 4.4. En el cas dels ACKs no necessitem afegir cap informació en la capçalera ja que en aquest cas no hi ha una transferència d'informació. Amb el camp genèric que identifica el tipus de paquet ja en tenim suficient.

En la figura 4.4 podem veure el diagrama de classes bàsic dels elements que hem vist fins al moment. En el capítol d'implementació veurem aquest diagrama amb més detalls.

4.4 Control d'errors

Com hem vist en l'anterior capítol, un dels requisits del nostre sistema era la tolerància a fallades. Es a dir, el que no podem permetre és que la transferència

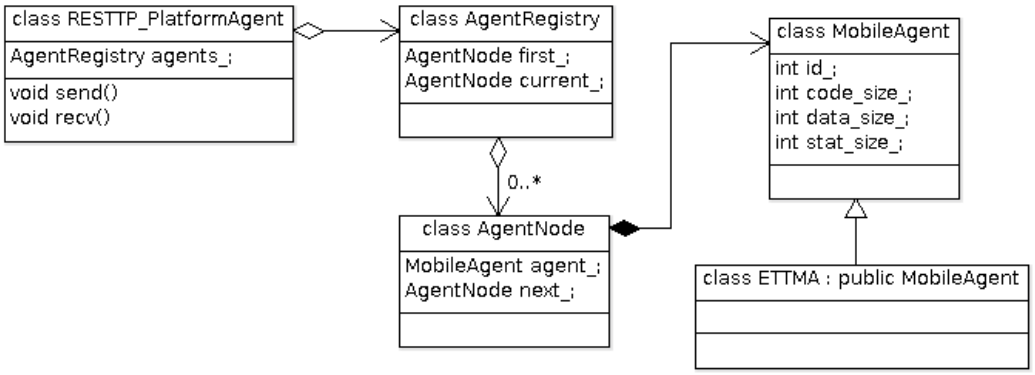

Figura 4.4: Diagrama de classes

d'algun dels recursos de l'agent falli. Per tant, hem dissenyat un sistema molt simple en el qual si falla la transferència d'algun dels recursos, informem a la plataforma d'origen, cancelem la migració i eliminen els recursos ja transferits.

Per altra banda, utilitzarem un esquema basat en ACKs en la fase de descobriment de plataformes de l'agent ETTMA per tal d'actualitzar les taules de decisió. En concret, per cada missatge que s'envia esperarem rebre un ACK. Aquest ACK ens informarà que, efectivament, la plataforma destí està al nostre abast.

4.5 Conclusions

En aquest capítol hem vist quin és el disseny global del nostre model de simulació per la migració d'agents. Hem vist que els principals elements són la plataforma, que utilitza el protocol RESTTP, i els agents mòbils. També hem vist com els agents deixen de tenir el control sobre la migració, sent la plataforma qui s'encarrega de controlar tot el procés i preguntar als agents si volen migrar o no. També hem vist la necessitat de simular el comportament d'un tipus d'agent concret com a prova de concepte. Hem escollit l'ETTMA com un possible candidat i hem definit un protocol pel descobriment de plataformes en un entorn MANET. Finalment, hem vist com un esquema molt senzill de control d'errors ens pot proporcionar la tolerància a fallades.

Capítol 5

Implementació i prova

En aquest capítol veurem com s'ha dut a terme la fase d'implementació del disseny presentat en el capítol anterior. Començarem veient l'implementació des d'un punt de vista global per anar centrant l'atenció en els detalls. Finalment, veurem quines han estat les proves realitzades per comprovar el correcte funcionament del sistema i unes breus conclusions sobre els resultats obtinguts.

5.1 Implementació

Els elements centrals són la plataforma i els agents mòbils, però també hem vist que necessitem paquets, temporitzadors, un registre d'agents i una taula de decisió. Per això, hem decidit crear els següents fitxers font:

- **mtp_http_packets.h**: conté la descripció de totes les estructures necessàries per implementar les capçaleres dels paquets HTTP, ACL i MTP_HELLO.
- **resttp_platform.{h,cc}**: contenen les classes corresponents a l'implementació de la plataforma (RESTTP_PlatformAgent), els temporitzadors i una descripció de la taula de decisió utilitzada per l'ETTMA.
- **mobile_agent.{h,cc}**: contenen les classes necessàries per l'implementació dels agents mòbils. Concretament, podem trobar l'implementació de la classe base amb el comportament bàsic dels agents i l'implementació de

l'ETTMA. També trobem la classe AgentRegistry, una llista de tots els agents disponibles en la plataforma.

A continuació veurem com s'implementen i quines són les particularitats de cada un dels elements presentats anteriorment. El diagrama de classes global es pot veure en la figura 5.1.

5.1.1 RESTTP_PlatformAgent

Com hem comentat anteriorment, inserirem la classe RESTTP_PlatformAgent a la capa de transport del NS2 i per tant, hem de derivar de la classe *Agent*. Aquesta classe ens oferirà els serveis bàsics de transport. En conseqüència, necessitem els següents mètodes per enviar i rebre paquets:

- *send_req(ns_addr_t, int, int)*: mètode que rep com a paràmetres l'adreça de la plataforma destí, que en NS2 es representa mitjançant l'estructura *ns_addr_t*, un identificador del recurs que es vol demanar i el número de seqüència de la petició HTTP a enviar. S'encarrega de construir la capçalera del missatge HTTP amb la informació corresponent i enviar la petició a la plataforma destí.

- *send_acl(ns_addr_t, int)*: mètode que rep com a paràmetres l'adreça de la plataforma destí i un identificador del tipus de missatge ACL a enviar (petició de transferència, informe d'error, etc.). S'encarrega de construir la capçalera del missatge ACL i enviar-lo a la destinació.

- *recv(Packet*, Handler*)*: és el mètode que s'invoca des del NS2 quan un paquet arriba a un agent de la capa de transport i se li passa una referència al paquet rebut. S'encarrega de realitzar el corresponent processament en funció del tipus de paquet rebut.

Hi ha un altre mètode que es necessari implementar i aquest és *command(int argc, const char*const* argv)*. Aquest mètode ens permetrà interaccionar amb l'script oTCL i és on hem d'implementar totes les comandes que des d'aquest es

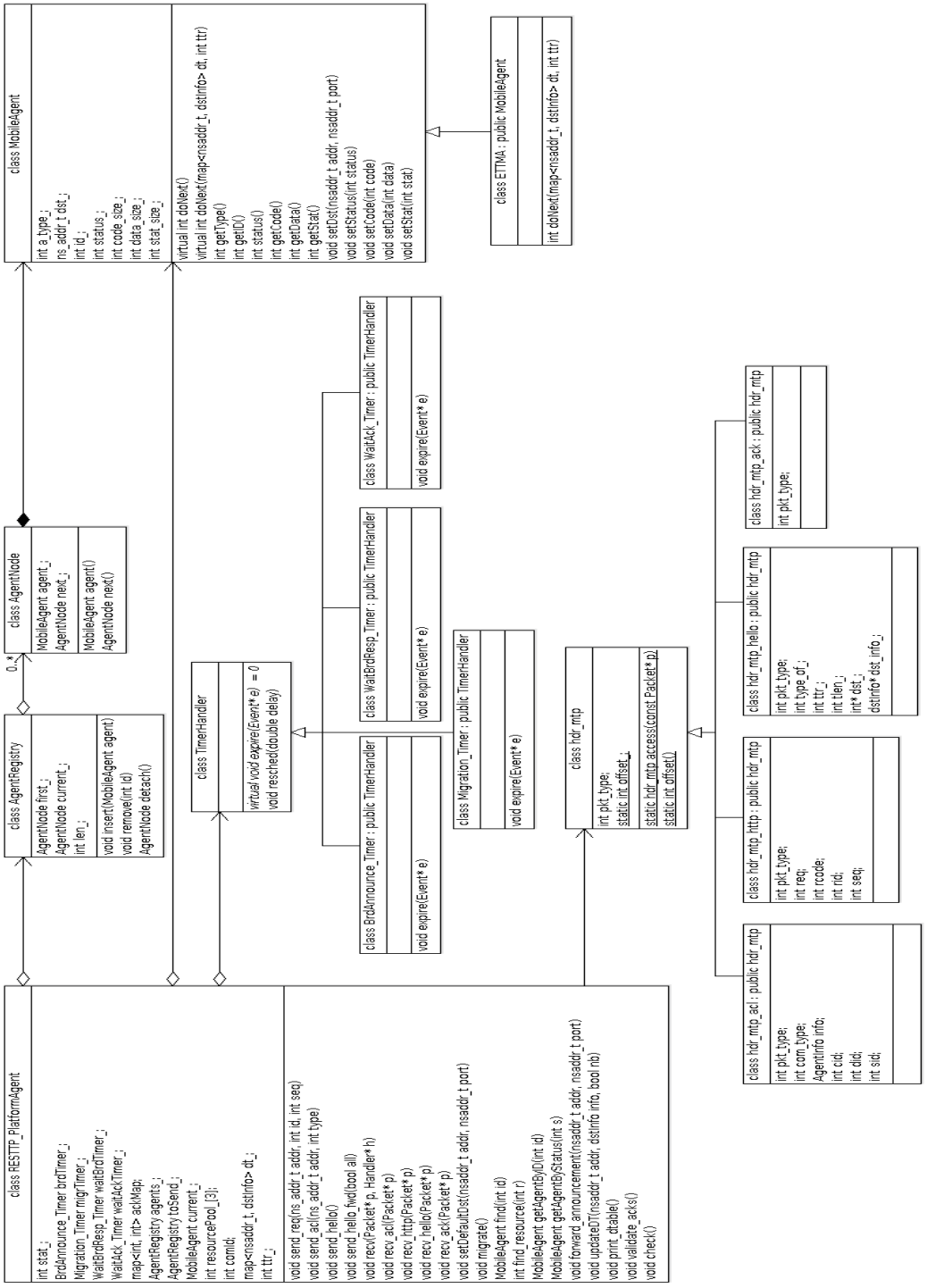

Figura 5.1: Diagrama de classes

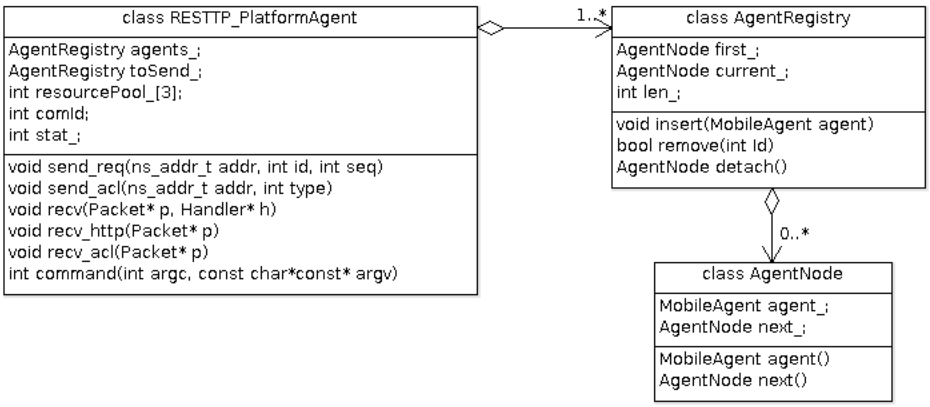

Figura 5.2: Diagrama de classes de la plataforma

poden invocar. Per exemple, tenim la comanda **create-agent** que s'encarregarà de cridar els mètodes necessaris de la classe MobileAgent per crear un nou agent i la comanda **migrate-agents** que procedirà a la migració dels agents que expressin aquesta voluntat.

El servei de pàgines blanques de la plataforma estarà representat per un **Agent Registry**, sent aquest un atribut de la plataforma. L'AgentRegistry serà una llista formada per elements de tipus **AgentNode** que contindran una referència a un agent mòbil i un punter al següent agent mòbil disponible en la llista. No només mantindrem un registre dels agents residents en la plataforma sinó també un registre del agents que estan preparats per migrar per poder aplicar una política de transferència en cas que varis agents vulguin migrar al mateix temps.

Hem afegit també un atribut a la plataforma que representi el seu estat de funcionament. Es a dir, podem tenir una plataforma en un node que si pot reencaminar paquets, però no està operativa per processar agents. Així podem parlar d'un estat ONLINE o OFFLINE de la plataforma. En l'estat OFFLINE els agents no podran migrar cap a aquesta plataforma.

En el diagrama de classes de la figura 5.2 podem veure l'esquema bàsic de la plataforma.

5.1.2 MobileAgent

Per la implementació dels agents mòbils necessitem una classe base de la que derivar tots els tipus concrets d'agents. Aquesta classe ens ha de proporcionar un comportament bàsic, que és el de migrar sempre. Hi ha una serie d'atributs que són comuns a tots els agents i són:

- **id_**: identificador únic de l'agent.
- **a_type_**: tipus d'agent (p.e. ETTMA).
- **dst_**: en el cas de migrar, l'agent ha d'especificar l'adreça de la plataforma destí a la que vol migrar.
- **code_size_**: mida del codi de l'agent.
- **data_size_**: mida de les dades de l'agent.
- **state_size_**: mida de l'estat d'execució de l'agent.

El mètode principal d'aquesta classe, i que els agents que derivin d'aquesta podran sobreescriure, és el mètode *int doNext()*. La plataforma cridarà aquest mètode per saber quina és l'acció que vol realitzar l'agent (MIGRAR, MORIR o altres). En el cas de l'agent base aquesta funció retornarà sempre una constant que representa l'acció de migrar.

Un agent que derivarà d'aquesta classe base és l'ETTMA sobre el qual hem parlat en el capítol de disseny. Aquest agent basa la seva decisió en una taula de decisió que recull els diferents TTRs de les diferents plataformes que estan al seu abast.

La taula de decisió la representarem mitjançant una taula hash indexada per l'adreça de la plataforma i el valor corresponent a la clau serà una estructura de tipus **dstInfo** amb tres camps: *port_*, que indica el port de la plataforma, *ttr_*, que indica el TTR de la plataforma i *nb_*, que indicara si la plataforma es troba en entrega directa o no. Tant la taula de decisió com el TTR seran

atributs de la plataforma. Per tant, haurem de modificar l'esquema bàsic presentat en la figura 5.2 per afegir aquests nous atributs i els mètodes de actualització i tractament d'aquests atributs. També haurem d'afegir un nou mètode *int doNext(map<nsaddr_t,dstInfo> dt, int ttr)* que incorpori aquesta modificació. Llavors l'agent decidirà si ha de migrar o no en base a l'actual topografia de la xarxa i el TTR de la plataforma en la qual es troba.

En la figura 5.1 podem veure un diagrama de classes integral de la implementació feta. En aquest diagrama podem veure els atributs i mètodes afegits a la implementació bàsica de la plataforma per tractar amb aquest nou tipus d'agent.

5.1.3 Paquets

En el capítol anterior hem vist que era necessari l'intercanvi de quatre tipus de paquets diferents entre les diferents plataformes (ACL, HTTP, MTP_HELLO, ACK). En canvi, el NS2 imposa que dos agents situats en els extrems d'una connexió només es poden intercanviar un sol tipus de paquet. Podem solucionar aquest problema definint una jerarquia en la que els paquets mencionats anteriorment derivin d'un mateix paquet base.

Noves capçaleres de paquets són introduïdes al simulador definint una estructura C++ amb els camps necessaris i es guarden en una estructura de tipus *Bag of Bits*. Llavors es proporciona un mètode d'accés mitjançant un *offset* per recuperar els camps de la capçalera. Això no suposa un problema en el nostre cas perquè una estructura C++ és pot veure com una classe els membres de la qual són públics per defecte.

Per tant, el que necessitarem és un atribut comú a totes capçaleres que indiqui de quin tipus de paquet es tracta, i per mantenir la coherència en quant a l'accés, necessitarem reservar un espai igual a la mida de l'estructura més gran en el *Bag of Bits*. En la figura 5.3 podem veure un petit diagrama de la jerarquia implementada.

5.1.4 Temporitzadors

A continuació veurem els quatre cassos on el temps juga un paper important:

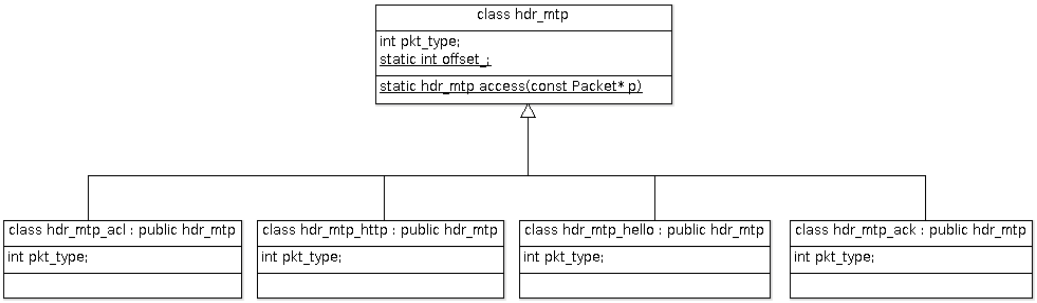

Figura 5.3: Diagrama de classes dels paquets

1. En el capítol anterior ja hem vist que és la plataforma qui pregunta als agents si volen migrar o no. Aquest interrogatori s'ha de produir constantment ja que al llarg de la simulació es poden crear nous agents o un agent pot variar la seva decisió d'acord amb al seu comportament. La forma més fàcil de solucionar aquest problema és interrogar als agents periòdicament. Per tant, necessitem un temporitzador que ens marqui aquest període, que en el nostre cas serà d'**1 segon**.

2. Una altra acció que s'ha de dur a terme periòdicament, en el cas de l'agent ETTMA, és el descobriment de plataformes ja que estem simulant un entorn MANET. En una MANET la topologia de la xarxa és molt dinàmica i canvia molt freqüentment. Per tant, necessitarem un segon temporitzador que ens marqui aquest període d'actualització de la topologia. Basant-nos en una observació empírica, hem decidit triar un període de **10 segons**.

3. Seguint amb l'ETTMA, en la fase d'anunciació d'existència, no sabem a priori quantes plataformes estan al nostre abast i per tant, ens hem d'esperar un cert període de temps per rebre totes les respostes. Després d'observar empíricament l'impacte del valor del temporitzador hem decidit triar un temps d'espera de **0.2 segons**.

4. Seguint amb el mateix escenari dels punts anteriors, ens cal un temporitzador per esperar els ACKs en resposta als missatges UNICAST d'actualització. Utilitzarem el mateix temps d'espera que en el cas anterior: **0.2**

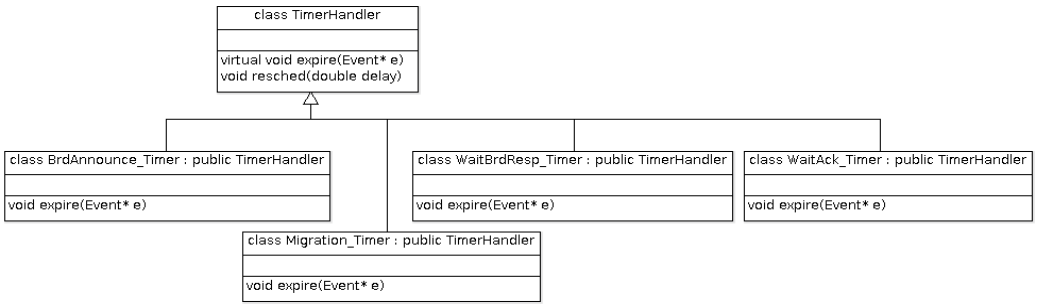

Figura 5.4: Diagrama de classes dels temporitzadors

segons.

En NS2 tots els temporitzadors deriven de la classe base *TimerHandler* i si es vol implementar un nou timer el que hem de fer és sobreescriure el mètode *expire(Event* e)*. Quan es dispara un temporitzador s'invoca el mètode *expire()* per saber quines accions s'han de realitzar. Podem veure el diagrama de classes en la figura 5.4.

5.2 Prova

Pel desenvolupament d'aquest projecte hem utilitzat una metodologia basada en cicles evolutius per facilitar la detecció i correcció d'errors a mesura que s'anava avançant. En l'apèndix A podem veure com configurar i quines són les modificacions necessàries en NS2 per afegir el model implementat i realitzar simulacions a partir d'aquest.

Per realitzar les proves, el que hem fet és dissenyar varis escenaris amb topologia diferent i simular la migració d'agents mòbils dintre d'aquests escenaris. Els escenaris han estat creats mitjançant el llenguatge oTCL, utilitzant el NAM com eina de visualització gràfica de les simulacions. Però, per supervisar totes les accions, l'arribada de missatges, la presa de decisions dels agents no en tenim prou amb una simple eina de visualització gràfica. Per tant, utilitzarem els fitxers de traça amb extensió *.tr* que ens proporciona també el NS2. En aquests fitxers es guarda la seqüència de tots els paquets enviats durant la simulació i d'a-

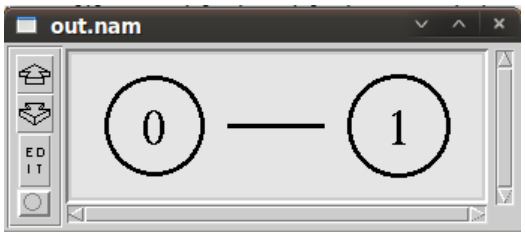

Figura 5.5: Escenari 1

questa manera podem comprovar que realment segueixen els passos del protocol implementat, així com també la política de decisió dels agents.

A continuació veurem en quins escenaris s'han dut a terme les simulacions.

5.2.1 Escenari amb dues plataformes

El primer escenari, és un escenari molt bàsic en el qual tenim dos nodes connectats mitjançant un cable. L'enllaç té un ample de banda de 1Mbps amb un delay de 10ms i la política de gestió de cues que utilitzarem en els dos extrems és la DropTail. En la figura 5.5 podem veure aquest escenari.

Inicialitzem una plataforma en cada un dels nodes i establim com a destinació per defecte dels agents de la plataforma 0, la plataforma 1. Creem un agent mòbil en la plataforma 0 i farem que en l'instant 0.2 de simulació, la plataforma comenci la migració.

Hem pogut comprovar com els paquets s'envien segons la seqüència definida pel protocol RESTTP i com l'agent primer es crea en la plataforma destí i després es destrueix en la plataforma origen. També hem comprovat que el comportament és idèntic si substituïm el cable que connecta els dos nodes per un medi de transmissió wireless.

Una cosa que podem observar a partir de la simulació és el temps que tarda un agent en realitzar una migració completa. Es a dir, el temps que es tarda en migrar a una altra plataforma i tornar a la plataforma origen. En la figura 5.6 podem veure el temps que tarden en migrar varis agents de diferent mida.

El comportament esperat seria que el temps de migració augmenti a mesura que incrementem la mida de l'agent. La figura 5.6 precisament mostra aquest

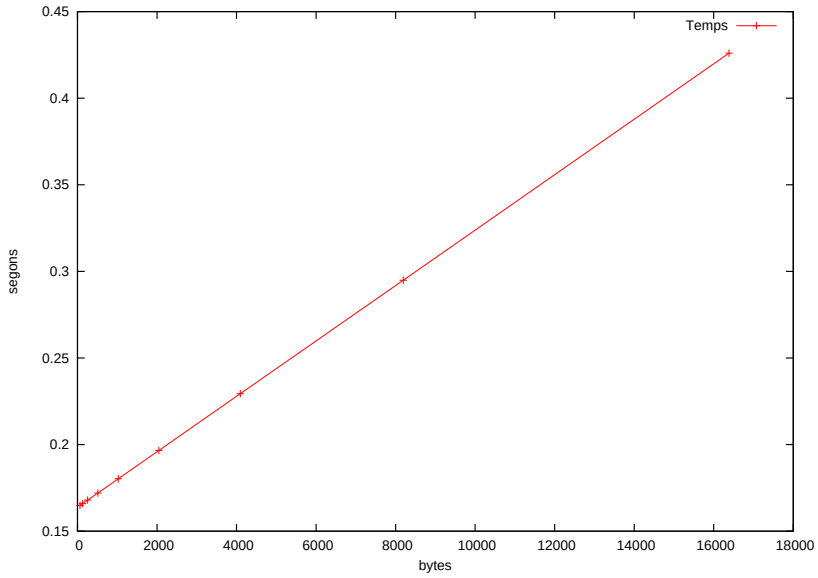

Figura 5.6: Temps en realitzar una migració completa

comportament.

5.2.2 Escenari amb moviment de nodes

L'escenari anterior és un escenari molt bàsic en el qual podem comprovar el correcte funcionament del protocol, però el que realment ens interessa és la simulació de la migració d'agents en entorns molt més complexes, com ara una xarxa MANET. En una xarxa MANET ens afrontem a un canvi dinàmic de la topologia i l'agent ha de saber reaccionar davant aquests canvis.

L'escenari que proposem aquí esta composat per 6 nodes en una xarxa mòbil ad-hoc (figura 5.7). L'algorisme d'encaminament que utilitzarem és el AODV (*Ad hoc On-Demand Distance Vector*) [PE99]. A partir de l'instant 10.0 de la simulació provocarem que el node 0 comenci a desplaçar-se cap a la posició (0,0) de la topologia, el node 1 es desplaci en direcció als nodes 2 i 5, el node 5 es desplaci en direcció al node 4, i que els nodes 3 i 4 comencin a desplaçar-se horitzontalment.

Sobre cada un d'aquests 6 nodes hi haurà una plataforma i el que farem serà crear dos agents mòbils que tindran com a destinació la plataforma 1. El primer

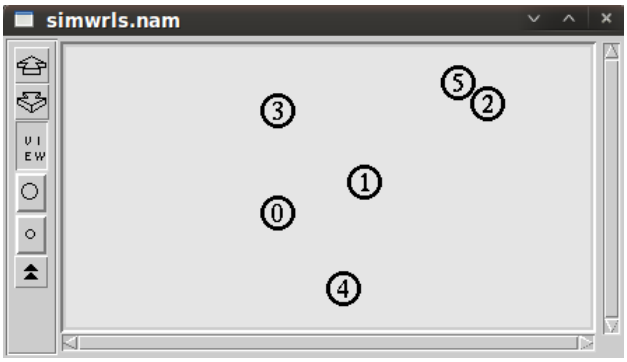

Figura 5.7: Escenari 2

Plataforma	0	1	2	3	4	5
TTR	200	300	250	254	150	50

Taula 5.1: Assignació de TTRs

agent migrarà en l'instant de temps 5.0 mentre que el segon en l'instant 20.0. El que veurem és que la plataforma 0 durà amb èxit la migració del primer agent ja que la plataforma 1 està al seu abast. En canvi, la migració del segon agent no tindrà lloc perquè en l'instant de temps 20.0 la plataforma 1 restarà fora del radi de cobertura de la plataforma 0. Per tant, hem comprovat que les plataformes tenen la capacitat de detectar canvis en l'entorn i actuar davant d'aquests.

En el mateix escenari hem creat un agent de tipus ETTMA en la plataforma 0 per veure que aquest és capaç de prendre una decisió basada en els TTR de les diverses plataformes. Per fer això hem assignat a cada plataforma un valor de TTR d'acord amb la taula 5.1.

Hem observat que la simulació mostra exactament el comportament esperat. Es a dir, l'agent salta a la plataforma 4 (de les que té al seu abast en aquest moment, la plataforma 4 té un TTR menor) per després saltar a la plataforma 5 que és la plataforma amb el mínim valor de TTR i accessible des de la plataforma 4. També hem pogut comprovar la correcta actualització de la taula de decisió de la plataforma 4 ja que inicialment no contenia una entrada per la plataforma 5.

5.2.3 Escenari situació d'emergència

Fins ara hem comprovat diferents aspectes funcionals del nostre sistema, però el que ens interessa realment es simular escenaris reals i analitzar quin és el comportament dels agents dintre d'aquests escenaris.

L'escenari que ens proposem simular aquí és el d'una situació d'emergència provocada per un desastre com un terratrèmol, un tsunami o un atemptat terrorista, en la que es despleguen una sèries de nodes corresponents als serveis mèdics que han d'atendre als ferits. Per donar-li un matís realista a la simulació, hem creat un escenari basant-nos en l'accident aeri que va succeir a l'aeroport de Barajas el 20 d'agost de 2008.

Degut a un error humà dels pilots i, a la mateixa vegada, a un problema amb la mecànica de l'avió, el vol JF3022 amb destí Gran Canària de la companya Spanair va caure al poc d'enlairar-se i es va estavellar al costat de l'aeroport. Les part de l'avió es van dispersar en un radi de 200 metres del lloc on va caure l'avió, al costat d'un camí rural habilitat per a la circulació de vehicles.

El dispositiu d'emergència desplegat va ser impresionant [DEP08, DPC08] i es va aconseguir localitzar 28 supervivents, tot i que en diferent estat de salut. L'escenari té les següents característiques:

- 3 àrees: àrea de l'incident (impacte de l'avió), l'àrea de tractament de pacients (situada al costat de l'àrea de l'incident, amb 11 UVIs) i les àrees d'espera per fer sortir els ferits cap als hospitals i d'on surten les ambulàncies.

- L'àrea de l'incident fa 8000 m2, donat que les part de l'avió van quedar disseminades en un radi de 200 metres des de l'impacte. Aquesta àrea compta amb 15 nodes (cada grup de 4 persones compta com un node i s'han tingunt en compte 60 persones desplaçades).

- L'àrea de tractament de pacients està situada al costat de l'àrea de l'incident i compta amb 11 nodes (un node per cada UVI mòbil).

- L'àrea de sortida de la zona de l'accident està situada al costat de l'àrea de

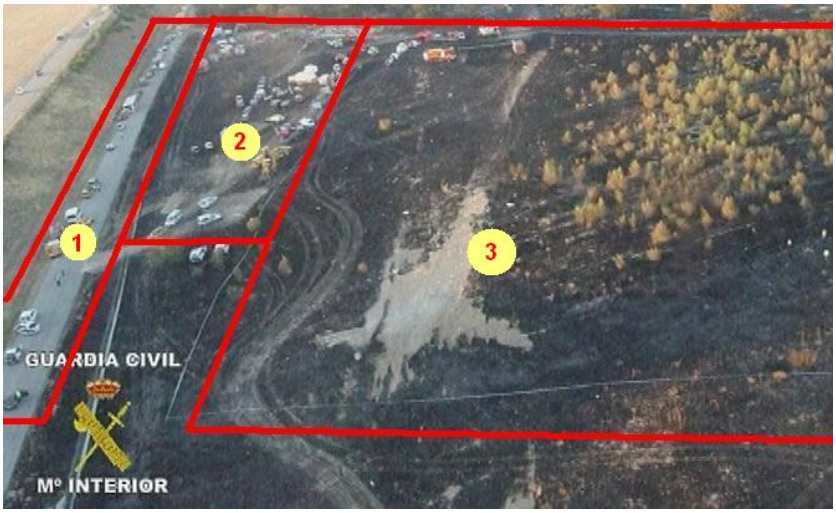

Figura 5.8: Delimitació de la zona d'impacte en zona d'incident (3), zona de tractament (2) i zona de sortida (1)

tractament de pacients i compta amb 35 nodes (un node per cada ambulància).

A la figura 5.8 podem veure les característiques d'aquest escenari.

Per simplificar una mica aquest escenari i amb la finalitat de reduir el temps de simulació, hem reduit el nombre de nodes a 19 (15 a la zona d'incident, 2 a la zona d'espera i 2 a la zona de sortida). Podem veure la disposició inicial dels nodes en la figura 5.9. El que es tracta de simular és el recorregut del personal sanitari des de la zona de tractament fins a la zona d'incidència i a l'inrevés. El temps durant el qual simularem aquest comportament serà de 3600 segons.

El que tenim en aquest escenari són una serie d'agents mòbils ETTMA que intentaran migrar el més ràpid possible cap als nodes de la zona de sortida. El que hem pogut comprovar és com les plataformes actualitzen les seves taules d'acord amb la topologia actual de la xarxa i com els agents mòbils salten sempre cap a les plataformes que els ofereixen un TTR menor. Tots els agents creats han complert amb el seu objectiu: arribar als nodes de la zona de sortida.

Tot i que no és l'objectiu dur a terme una prova de rendiment, sí que és interessant veure si el mòdul afegit per la migració d'agent introdueix algun overhead en el temps d'execució de la simulació. Per comprovar això, utilitzant aquest mateix

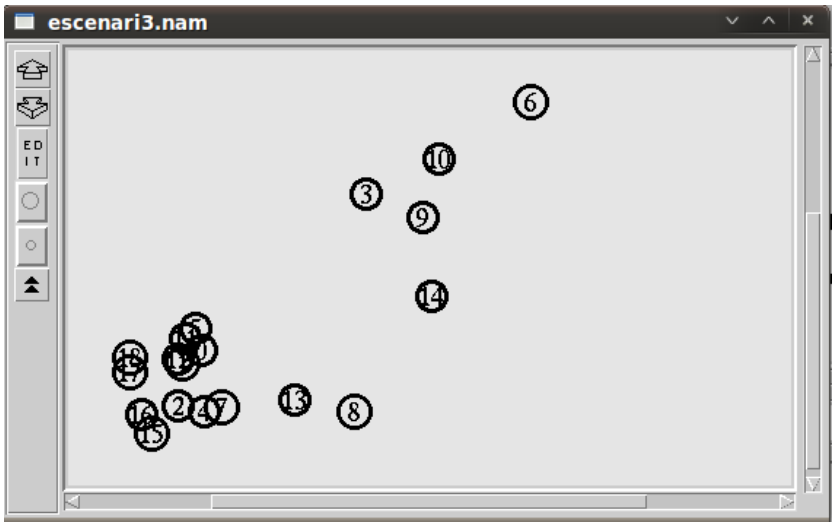

Figura 5.9: Escenari 3

escenari, hem substituit els agents mòbils per connexions FTP. Hem crear dues sessions FTP entre dues parelles de nodes de la zona d'impacte i una sessió entre un node de la zona d'impacte i un altre situat a la zona de sortida. El temps d'execució de la simulació pels dos cassos anteriors es pot veure a la figura 5.10. Per validar els resultats obtinguts hem repetit cinc vegades les simulacions.

Hem pogut comprovar que el fet d'utilitzar agents mòbils no introdueix cap overhead en el temps d'execució de la simulació, sinó tot el contrari, el disminueix. En la presentació de l'escenari 2 ja veiem que el temps de migració d'un agent de mides fins i tot mitjanes és baix (aproximadament 0.42 segons), i els resultats obtingut en aquest escenari ens ho confirmen.

5.3 Conclusions

En aquest capítol hem vist quina ha estat l'implementació del protocol de migració d'agents mòbils i del comportament de l'agent ETTMA. Hem vist com las classe RESTTP_PlatformAgent ens pot oferir els serveis bàsics de transport i com un registre d'agents es pot implementar fàcilment mitjançant una llista. Hem vist també com aplicar algunes propietats del llenguatge de programació C++ per evi-

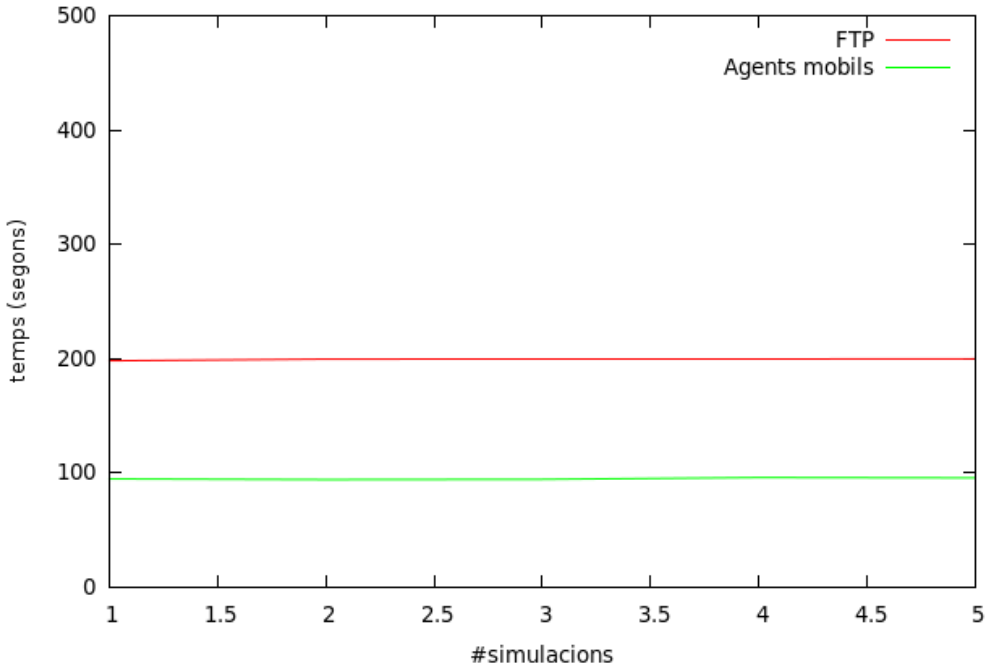

Figura 5.10: Temps de simulació

tar la restricció que imposa l'intercanvi d'un únic tipus de paquet entre dos agents de la capa de transport, i com s'implementen els temporitzadors en el NS2.

Finalment, hem vist tres escenaris de prova, en el quals hem pogut comprovar el correcte funcionament de la nostra aplicació. Hem vist com els resultats de les simulacions són totalment coherents amb els resultats esperats, tant en xarxes cablejades com en xarxes ad-hoc amb nodes mòbils. Una observació directa de les simulacions és que el temps que un agent necessita per realitzar una migració completa augmenta linearment amb la seva mida. A més a més hem pogut comprovar que la simulació del comportament de l'agent ETTMA compleix totes les especificacions d'aquest (descobriment de plataformes de forma periòdica, actualització de les taules de decisó i presa de decisions). També hem pogut comprovar que el mòdul de migració no afegeix cap overhead en el temps de simulació i per tant, podrem dur a terme simulacions en temps raonables (aproximadament 90 segons de temps d'execució per temps de simulació de 3600 segons).

Capítol 6

Conclusions

En aquest capítol veurem un recull dels objectius acomplerts, un resum de la feina realitzada per acomplir aquests objectius, la planificació temporal resultant i algunes línies de treball futur. Finalment, exposarem una valoració personal del projecte.

6.1 Objectius acomplerts

Si recordem, el principal objectiu d'aquest projecte era construir un entorn de treball que ens permeti simular el comportament general dels agents mòbils. Per fer això necessitavem un pas previ d'estudi dels protocols de migració i del simulador NS2. Podem dir que hem assolit amb èxit aquests objectius.

Hi ha diversos protocols de migració d'agents mòbils, un dels quals és el RESTTP. Aquest protocol no està soportat dintre del NS2, fet que ens ha portat a dur a terme una implementació del protocol i afegir-ho al simulador. Dintre d'un escenari de simulació els agents són capaços de migrar entre varies plataformes disponibles. La decisió de quina serà la plataforma destí es pot fer de forma estàtica, on l'usuari especifica quina és aquesta plataforma, o bé de forma dinàmica, on l'agent ha de prendre una decisió en base a una estructura més complexa (taula de decisió) que s'actualitza de forma periòdica per indicar canvis en l'entorn. Serà la pròpia plataforma qui interrogarà als agents per determinar si volen

migrar, i en cas afirmatiu portarà a terme un procés de transferència utilitzant el protocol RESTTP. Una vegada finalitzada la transferència, l'agent quedarà registrat a la plataforma destí i serà eliminat de la plataforma origen. El propi protocol RESTTP, amb l'utilització dels protocols HTTP i ACL, ens ofereix un esquema robust i tolerant a fallades de transferència d'agents.

Un altre objectiu d'aquest projecte era dissenyar varis escenaris per la simulació d'agents. També és un objectiu que hem assolit amb èxit. El propi NS2 ens aporta les eines necessàries per definir un escenari de simulació. Mitjançant un script descrit en llenguatge oTCL es possible definir tots els paràmetres i elements d'una simulació: nombre de nodes, topologia, models de propagació i moviment de nodes en el cas de la tecnologia wireless, etc.

Finalment, un objectiu important d'aquest projecte era poder implementar i simular el comportament d'un tipus d'agent concret. No només volem veure que els agents poden migrar sinó que ho fan en base a uns objectius ben definits, per dur a terme una certa tasca. També podem afirmar que hem acomplert aquest objectiu.

L'agent seleccionat ha estat el *Electronic Triage Tag Mobile Agent*. Utilitzat en situacions d'emergència, l'ETTMA porta informació de la situació de les víctimes al centre d'operació el més ràpid possible. Utilitza un servei de descobriment de plataformes per saber quines plataformes té al seu abast. La decisió de la plataforma destí ja no és estàtica o aleatòria sinó que l'agent pren una decisió en base a un atribut de la plataforma que indica quin és el seu temps restant per tornar al centre d'operació (TTR). Periòdicament, es portat a terme un procés de descobriment de plataformes i es selecciona la que tingui un TTR menor.

En conclusió, podem dir que s'han acomplert tots els objectiu proposat a l'inici d'aquest projecte.

6.2 Planificació temporal final

Com podem veure a la figura 6.1, la planificació temporal final ha estat diferent de l'inicial. Tot i així, la durada del projecte només s'ha vist incrementada amb

set dies, la qual cosa no va afectar l'entrega final del projecte.

La part que es va veure afectada és la codificació del mòdul que simuli el comportament d'un agent concret. Per una banda, l'integració del model dintre del NS2 no ha estat una tasca gens fàcil degut a les varies limitacions del simulador i a la poca ajuda que hi ha referent al desenvolupament en NS2. Això ha fet que hi dediquem més hores de les inicialment havíem previst. Per altra banda, hem hagut de revisar el disseny de la part de descobriment de plataformes de l'agent escollit perquè no quedava molt clara la seva implementació. En canvi, realitzar les simulacions necessàries per prova el funcionament del model ens ha costat menys de l'esperat, reduint la durada d'aquesta tasca a només sis dies.

6.3 Treball futur

Tot i finalitzat el projecte, resten alguns punts que acceptarien millores o encara podem afegir funcionalitats addicionals, que per manca de temps no s'han pogut dur a terme. Aquestes són algunes de línies d'ampliació proposades:

1. Implementació d'un nou mòdul que simuli el comportament d'un altre tipus d'agent. Es podria implementar el comportament del *Bundle Scheduler* proposat pel Carlos Borrego[1] per treballar amb cues. Això ens permetria realitzar simulacions d'aquest agent en diferents escenaris i veure quins són els beneficis que aporta.

2. Dissenyar i implementar, si s'escau, un nou protocol per la fase de descobriment de plataformes de l'agent ETTMA escollit per fer la prova de concepte. Per exemple, si es pensés una forma més eficient per descobrir les les plataformes que hi ha a l'abast. D'aquesta forma es podrien realitzar comparacions sobre l'impacte de diferents protocols.

3. Implementar varis protocols de migració. Hi ha altres protocols disponibles per la migració d'agents i es podria fer una implementació d'aquests

[1] Carlos Borrego és un investigador del grup de recerca SeNDA i el treball al que es fa referència està pendent de publicació, per això encara no existeix una referència

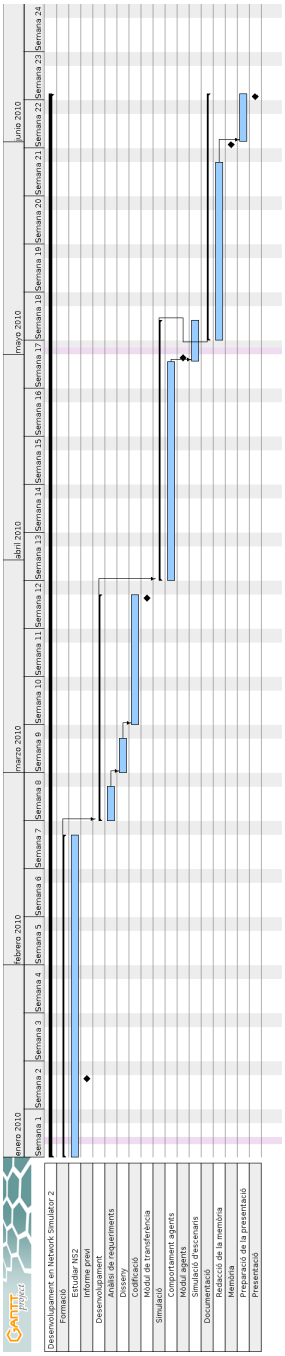

Figura 6.1: Planificació temporal final

protocols i veure quines són les seves diferències. Això s'hauria d'integrar dintre de l'arquitectura IPMA [FIPA-IPMA] i la plataforma hauria de poder escollir quin és el millor protocol que necessita per migrar.

4. Modificar o millorar una eina per la generació automàtica d'escenaris, com per exemple el Bonn Motion, desenvolupat per investigadors en el camp de les xarxes MANET i les xarxes de sensors de la Universitat de Bonn. Necessitem una eina que ens generi de forma automàtica la posició inicial i moviment dels nodes que s'adaptin a les característiques d'un determinat escenari.

Aquestes línies de treball futur es podrien convertir en propostes per a propers projectes relacionats amb l'anàlisi del rendiment i comportament dels agents mòbils.

6.4 Valoració personal

Amb aquest projecte hem fet la nostra petita aportació al món de les simulacions i als interessats en la tecnologia d'agents mòbils. Amb aquest projecte puc dir que he aprés molt, tant a nivell tècnic com a nivell personal. És el final d'una etapa de la meva vida, però al mateix temps el punt de partida d'una nova etapa com a futur enginyer informàtic.

Amb aquest projecte he pogut viure totes les etapes necessàries en la realització d'un projecte i notar en primera persona la complexitat que això suposa, des de desviacions en la planificació fins a l'estrès provocat per l'acumulació de feina. És un projecte que no només m'ha servit per posar en pràctica els coneixements adquirits durant la carrera o per adquirir de nous, sinó que també m'ha servit a nivell personal. He pogut conèixer gent extraordinària, veure amb uns altres ulls el món de la recerca i donar-me compte que molt cops la pròpia motivació i la satisfacció per tenir una nova idea té molt més valor que una gran quantitat de diners.

El fet de treballar amb un projecte de codi obert i saber que qualsevol persona

pot utilitzar-lo en els seus futurs projectes suposa un motiu de satisfacció addicional. És un sentiment molt afalagador el fet de que la teva feina sigui reconeguda per altres persones.

Bibliografia

[AJ03] Altman, E., & Jiménez, T. (2003). *NS Simulator for beginners*. Univ. de Los Andes, Venezuela; Sophia-Antipolis, França

[CEM08] Cucurull, J. (2008). *Efficient Mobility and Interoperability of Software Agents*. Tesi doctoral - Universitat Autònoma de Barcelona, Departament d'Enginyeria de la informació i de les Comunicacions.

[DEP08] La experiencia del 11-M, fundamental en el protocolo de actuación activado por la Comunidad.(2008). Diari social digital Europa Press <http://www.europapress.es/epsocial/cooperacion-y-desarrollo-00331/noticia-accidente-barajas-experiencia-11-fundamental-protocolo-actuacion-activado-comunidad-20080822115553.html>

[DPC08] 153 muertos en el peor siniestro aéreo de los últimos 25 años.(2008). Diari digital El Periodico de Catalunya <http://www.elpais.com/articulo/espana/153/muertos/peor/siniestro/aereo/ultimos/25/anos/elpepuesp/20080820elpepunac_11/Tes>

[FIPA-IPMA] Cucurull, J., Martí, R., Robles, S., Borell, J., & Navarro, G. (2007). *FIPA-based Interoperable Agent Mobility*. Artícle presentat al Multi-Agent Systems and Applications V, Leipzig, Alemania. , 4696 319-321.

[FIPAMTP-HTTP] FIPA Agent Message Transport Protocol for HTTP Specification(2002). <http://www.fipa.org/specs/fipa00084/SC00084F.html>

[FIPASpec] Foundation for Intelligent Physical Agent (FIPA). (2010). <http://www.fipa.org/>

[FT02] Fielding, R. T., & Taylor, R. N. (2002). *Principled design of the modern web architecture*. ACM Transactions on Internet Technology. 2(2), 115-150.

[FV09] Fall, K., & Varadhan, K. (2009). *The ns Manual* <http://www.isi.edu/nsnam/ns/doc/index.html>

[GTu] Greis, M. Marc Greis Tutorial for the UCB/LBNL/VINT Network Simulator "ns2". <http://www.isi.edu/nsnam/ns/tutorial/>

[HM04] Herrera M., J. M. (2004). NS2 - Network Simulator. Valparaíso

[IEEE-SRS] IEEE. Recommended practice for software requierements specifications(1998). <http://standards.ieee.org/reading/ieee/std_public/description/se/830-1998_desc.html>

[MUN] Muñoz, J. L. *Análisis de protocolos con Net Simulator 2 (ns-2)*. Unpublished manuscript.

[MR+08] Martí, R., Robles, S., Martín-Campillo, A., & Cucurull, J. (2009). *Providing early resource allocation during emergencies: The mobile triage tag*. Journal of Network and Computer Applications, 32(6), 1167-1182.

[NS2Tu] NS-2 Tutorial. Demokritos University of Thrace <http://utopia.duth.gr/ skontog/work/wlesson.pdf>

[PE99] Perkins, C. E., & Royer E. M. (1999). *Ad hoc On Demand Distance Vector Routing*. Proceedings of the Second Annual IEEE Workshop on Mobile Computing Systems and Applications, 90-100.

[RG00] Rigo, A., & Genescà, G. (2000). *Tesis i treballs. Aspectes formals*. EUMO Editorial.

[RR04] Ros, F. J., & Ruiz, P. M. (2004). *Implementing a New Manet Unicast Routing Protocol in NS2*. Departament d'Enginyeria de la Informació i de les Comunicacions. Universitat de Murcia.

[SEs09] Serrano, R. (2009). *Estudi i simulació dels protocols d'encaminament en MANETs amb nodes d'alta mobilitat*. Tesina - Universitat Autònoma de Barcelona, Departament de la Informació i de les Comunicacions.

[uml05] UML Unified Modeling Language. (2005). <http://www.uml.org/>

[VMRC+06] Viera-Marques, P. M., Robles, S., Cucurull, J., Cruz-Correia, R. J., Navarro, G., & Martí, R. (2006). *Secure Integration of Distributed Medical Data Using Mobile Agents*. IEEE Intelligent Systems, 21(6), 47-54.

[WMA96] White, J. (1996). *Mobile Agents White Paper*

Apèndix A

Guia per incorporar el framework d'agents en el NS2

Hem implementat un nou protocol dintre del NS2, però no hem acabat. Necessitem dur a terme una sèries de canvis per integrar el nostre codi dintre del simulador. L'objectiu d'aquest annex és precisament donar a conèixer quins són aquests canvis. És com una mena de tutorial per facilitar la feina de futurs projectistes o persones interessades en desenvolupar protocols pel NS2.

A.1 Decidir l'estructura de directoris

En aquest punt donem per fet que es disposa del simulador NS2 correctament instal·lat amb totes les seves dependències. La versió utilitzada per realitzar aquest projecte és la 2.34.

Els fitxers font C++ amb l'implementació del nou protocol han d'estar continguts a la carpeta *ns-2.34/*, que la podem trobar dintre del directori on hem instal·lat el NS2. Podem copiar-los directament en aquesta carpeta o podem crear una nova carpeta dintre de l'estructura de directoris per posar els fitxers. Aquesta decisió esta a la lliure disposició del desenvolupador. En el nostre cas hem decidit crear una nova carpeta, que anomenarem *mtp*.

A partir d'ara, suposarem que totes les referències a fitxers de configuració del

NS2 estan en base a la carpeta *ns-2.34*.

A.2 Declarar nous tipus de paquets

Si recordem, hem tingut que declarar nous tipus de paquets i definir les seves capçaleres (ACL, HTTP i MTP_HELLO). El primer pas és informar al NS2 de l'existència d'aquests nous tipus de paquets. Totes les declaracions de paquets les podem trobar dintre del fitxer *common/packet.h*. En aquest fitxer podem veure que els paquets es defineixen com nombres constants. Es defineix un nou tipus de dades *packet_t* a partir del tipus bàsic *unsigned int* i els nous paquets es defineixen de la següent forma:

```
static const packet_t nom = constant
```

Hem de tenir en compte que la declaració *static const packet_t PT_NTYPE* ha de ser l'última. No es poden definir paquets després d'aquesta línia.

A partir d'ara, el NS2 ja sabrà de l'existència d'un nou tipus de paquet, que en el nostre cas li hem assignat el valor 62. El següent pas és assignar un nom textual al valor constant per identificar de forma més senzilla el nou paquet. Per fer això, en el mateix fitxer hem de localitzar la classe *p_info*. Aquesta classe té un atribut *name_* que realitza el mapeig entre el valor constant assignat al paquet i el seu nom. L'inicialització d'aquest vector es duu a terme en la funció *initName()* on haurem d'afegir una línia com la següent:

```
name_[nom] = valor textual
```

A partir d'aquest moment ja podem utilitzar la constant definida anteriorment com a identificador del nous tipus de paquet.

A.3 Generació de traces

Si recordem, l'objectiu de la simulació és obtenir un fitxer de traça que descriu tots els events que han tingut lloc durant la simulació. En el capítol 26 de [FV09] podem trobar una descripció de les traces. L'informació d'un paquet s'escriu en un fitxer sempre que es rebut, enviat o descartat. La classe o objecte encarre-

gat de fer aquest volcat és un objecte de tipus *Trace*. En el cas de la tecnologia wireless, aquesta funcionalitat s'obté mitjançat un objecte de tipus *CMUTrace*. Podem trobar una descripció més detallada sobre la generació de traçes en simulacions wireless en el capítol 16 de [FV09].

Per tant, haurem d'editar el fitxer *trace/cmu-trace.h* i afegir una nova funció dintre de la classe CMUTrace. El prototipus d'aquesta funció podria ser el següent:

```
void format_nom(Packet *p, int offset);
```

Hem d'implementar aquesta funció dintre del fitxer *trace/cmu-trace.cc*. A continuació podem veure l'implementació que hem fet nosaltres pels paquets HTTP.

```
struct hdr_ip *ih = HDR_IP(p);
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_mtp_http *hh = HDR_MTP_HTTP(p);

if (pt_->tagged()) {
   sprintf(pt_->buffer() + offset,
       "-http:t %d -http:c %d -http:r %d "
       "-http:sn %d -http:sz %d -http:s %d "
       "-http:sp %d -http:d %d -http:dp %d HTTP",
       hh->req,
       hh->rcode,
       hh->rid,
       hh->seq,
       ch->size_,
       ih->src_.addr_,
       ih->src_.port_,
       ih->dst_.addr_,
       ih->dst_.port_);
} else if (newtrace_) {
   sprintf(pt_->buffer() + offset,
```

```
        "-P http -Pt %d -Pc %d -Pr %d -Psn %d "
        "-Psz %d -Ps %d -Psp %d -Pd %d -Pdp %d HTTP",
        hh->req,
        hh->rcode,
        hh->rid,
        hh->seq,
        ch->size_,
        ih->src_.addr_,
        ih->src_.port_,
        ih->dst_.addr_,
        ih->dst_.port_);
} else {
   sprintf(pt_->buffer() + offset,
        "[%d %d %d %d %d [%d:%d %d:%d]] (HTTP)",
        hh->req,
        hh->rcode,
        hh->rid,
        hh->seq,
        ch->size_,
        ih->src_.addr_,
        ih->src_.port_,
        ih->dst_.addr_,
        ih->dst_.port_);
}
```

De l'anterior codi podem deduir que hi ha tres tipus de traces: amb etiquetes, amb un nou format i tradicionals. La sintaxi utilitzada és bastant senzilla i intuïtiva com es pot observar. Tant en el cas de les traces amb etiquetes com les de nou format s'utilitzen etiquetes (tags) per identificar cada camp de la traça. En el nostre cas hem utilitzat *t* pel tipus de missatge HTTP (REQUEST, RESPONSE), *c* pel codi retornat en un RESPONSE, *r* pel recurs demanat i *sn* pel número de seqüència.

Per poder utilitzar aquesta funció, hem de modificar també la funció *format()*

que es troba en el fitxer *trace/cmu-trace.cc*. En aquesta funció ens trobem amb un selector del tipus de paquet i una crida a la funció que genera les traçes per aquest tipus de paquet. Per tant, haurem d'afegir un nou cas que es correspongui amb el nostre nou tipus de paquet i que cridi a la funció que acabem de crear.

A.4 Llibreries Tcl

Finalment, necessitem dur a termes algunes modificacions en els fitxers de configuració Tcl del NS2. El que farem serà afegir un nou tipus de paquet i proporcionar valors per defecte a atributs disponibles en oTCL.

Com hem explicat en el capítol 2 d'aquesta memòria, en NS2 trobem dos jerarquies: una jerarquia compilada (C++) i una jerarquia interpretada (Tcl). Per poder utilitzar un nou tipus de paquet durant la simulació, l'hem de definir també en la jerarquia interpretada. Per fer això, modificarem el fitxer *tcl/lib/ns-packet.tcl*. Hem de localitzar el següent codi i afegir el nom del nou tipus de paquet:

```
foreach prot {
    ...
    Encap  # common/encap.cc
    IPinIP  # IP encapsulation
    HDLC  # High Level Data Link Control

    MtpHTTP  # MTP-HTTP protocol
} {
    add-packet-header $prot
}
```

Una altra cosa que ens permet l'NS2 és accedir a atributs definits en les classes C++ des de l'script oTCL. Per exemple, si volem fer que des de l'script de simulació es pugui accedir directament a l'atribut TTR de la plataforma, amb el mètode *set*, dins del constructor de la classe RESTTP_PlatformAgent hem de fer el següent:

`bind(`*`nom de la variable Tcl`*`,&`*`atribut`*`);`

Llavors podrem accedir al atribut de la classe C++ utilitzant el nom especificat com a primer paràmetre de la funció *bind()*. Si fem això, també haurem de proporcionar un valor per defecte de l'atribut. El fitxer *tcl/lib/ns-default.tcl* conté l'assignació dels valors per defecte. Hem d'afegir al final d'aquest fitxer una línia com la següent:

```
Agent/RESTTP_PlatformAgent set ttr_ 100
```

D'aquesta forma estarem assignant un valor per defecte 100 a l'atribut *ttr_* de la classe RESTTP_PlatformAgent.

A.5 Makefile

Ara que ja ho tenim tot implementat, l'únic pas que ens queda és compilar! Per fer això haurem de modificar el *Makefile* per afegir els nostres fitxers. Hem d'afegir els fitxers objecte en la variable *OBJ_CC*. Per exemple, si només tenim un fitxer *mtp-http.cc* dins de la carpeta mtp hem d'afegir la següent línia en el *Makefile*:

```
OBJ_CC = \
    ...
    wpan/p802_15_4trace.o wpan/p802_15_4transac.o \
    apps/pbc.o \
    mtp/mtp-http.o \
    $(OBJ_STL)
```

Ara ja podem executar la comanda *make* i disfrutar del nostre propi protocol (o passar una estona resolent problemes de compilació!).

A.6 Exemple

A continuació podem veure un exemple d'script oTCL on es detalla la forma de crear una plataforma, assignar-la a un node i com es pot dur a terme una migració entre dos nodes.

```
#Creem un objecte simulador
set ns [new Simulator]

#Obrim dos fitxers de traça per guardar els
#resultats de la simulació
set tf [open out.tr w]
set nf [open out.nam w]
$ns trace-all $tf
$ns namtrace-all $nf

#Definim un procediment que serà cridat en
#última instància per tancar els fitxers de
#traça i aturar la simulació
proc finish {} {
       global ns nf tf
       $ns flush-trace
       close $nf
       close $tf
       exec nam out.nam &
       exit 0
}

#Creem dos nodes i els connectem amb un enllaç
#dedicat d'1Mbps, 10ms de delay i política de
#gestió de cues Drop Tail
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right

#Creem dues noves plataformes
```

```
set a0 [new Agent/RESTTP_PlatformAgent]
set a1 [new Agent/RESTTP_PlatformAgent]

#Posem les plataformes sobre el primer i
#segon nodes respectivament
$ns attach-agent $n0 $a0
$ns attach-agent $n1 $a1

#Especifiquem que les plataformes estan preparades
#per processar agents
$a0 on
$a1 on

#Creem un nou agent mòbil sobre la primer plataforma
#i especifiquem la seva mida
$a0 create-agent 1
$a0 set-agent-size 1 64 512 24

#Asignem una plataforma destí per defecte en cada
#plataforma. Tots els agents existents tindran com
#destinació per defecte aquesta plataforma
$a0 set-default-dst $a1
$a1 set-default-dst $a0

#Planifiquem els events que s'han de dur a terme
#durant la simulació.
$ns at 0.2 "$a0 migrate-agents"

#Aturem la simulació en l'instant 1.0 segons
$ns at 1.0 "finish"
$ns run
```

Apèndix B

Proves de funcionament

L'objectiu d'aquest apèndix es donar a conèixer els primers passos en l'interpretació dels resultats de les simulacions amb NS2. Per això, veurem el contingut dels fitxers de traça generats en els dos escenaris de prova.

B.1 Escenari 1

En el primer escenari es simula la migració entre dos nodes formant una petita LAN. El resultat esperat és que es produeixi la seqüència de passos indicada pel protocol RESTTP per migrar un agent del node 0 al node 1. El fitxer generat és el següent:

```
+ 0.2 0 1 MTP_ACL 36 ------- 0 0.0 1.0 -1 0
- 0.2 0 1 MTP_ACL 36 ------- 0 0.0 1.0 -1 0
r 0.210128 0 1 MTP_ACL 36 ------- 0 0.0 1.0 -1 0
+ 0.210128 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 1
- 0.210128 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 1
r 0.220288 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 1
+ 0.220288 0 1 MTP_HTTP 84 ------- 0 0.0 1.0 -1 2
- 0.220288 0 1 MTP_HTTP 84 ------- 0 0.0 1.0 -1 2
r 0.23096 0 1 MTP_HTTP 84 ------- 0 0.0 1.0 -1 2
+ 0.23096 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 3
- 0.23096 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 3
r 0.24112 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 3
+ 0.24112 0 1 MTP_HTTP 532 ------- 0 0.0 1.0 -1 4
- 0.24112 0 1 MTP_HTTP 532 ------- 0 0.0 1.0 -1 4
r 0.255376 0 1 MTP_HTTP 532 ------- 0 0.0 1.0 -1 4
```

```
+ 0.255376 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 5
- 0.255376 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 5
r 0.265536 1 0 MTP_HTTP 40 ------- 0 1.0 0.0 -1 5
+ 0.265536 0 1 MTP_HTTP 44 ------- 0 0.0 1.0 -1 6
- 0.265536 0 1 MTP_HTTP 44 ------- 0 0.0 1.0 -1 6
r 0.275888 0 1 MTP_HTTP 44 ------- 0 0.0 1.0 -1 6
+ 0.275888 1 0 MTP_ACL 36 ------- 0 1.0 0.0 -1 7
- 0.275888 1 0 MTP_ACL 36 ------- 0 1.0 0.0 -1 7
r 0.286016 1 0 MTP_ACL 36 ------- 0 1.0 0.0 -1 7
```

Podem veure 24 traces generades durant la simulació, 8 operacions d'encuament (indicades amb el símbol "+" en la primera columna), 8 operacions d'eliminació de la cua (indicades amb el símbol "-") i 8 events de rebre paquets (indicats amb el caràcter "r"). La segona columna indica l'instant de temps en el que s'ha produït l'event. Les dos columnes següents indiquen entre quins nodes s'ha produït aquest event. El camp següent mostra un nom descriptiu assignat al tipus de paquet tractat (veure apèndix A). La següent columna ens indica la mida del paquet incloent la mida de la capçalera IP.

El següent camp conté alguns flags, que en aquest exemple no s'utilitzen. Aquests flags s'utilitzen en alguns cassos pel control de la congestió, prioritat, mida de la finestra i altres (capítol 26 [FV09]). El següent camp és un indicador del flux al que pertany el paquet. En NS2 podem identificar el flux de paquets entre dos nodes per tal de diferenciar els paquets que pertanyen a una connexió. Els següents dos camps indiquen les adreces origen i destinació respectivament del paquet. La penúltima columna ens indicaria el número de seqüència però en NS2, són només aquells Agents interessats en implementar un esquema basat en número de seqüència qui modificaran aquest camp. Per últim, es proporciona un identificador únic del paquet. A cada paquet creat durant la simulació se li assigna un identificador únic.

B.2 Escenari 2

El segon escenari és un escenari més complex, s'utilitza un algorisme d'enrutament wireless i es generen moviments de nodes. El que es vol comprovar és que

les plataformes tenen la capacitat d'adaptar-se a canvis en l'entorn i per això s'intentaran realitzar dues migracions. En el primer cas, l'agent trobarà un camí fins a la plataforma destí i la migració es completarà amb èxit. En canvi, el segon agent no podrà migrar perquè la seva plataforma destí estarà fora del radi de cobertura de la plataforma origen. A continuació podem veure part del contingut del fitxer resultant de la simulació amb un format diferent al que havíem vist anteriorment. El diferent format es degut a que ara estem realitzant una simulació wireless, i per tant, l'informació que necessitem és diferent.

```
s 1.000000000 _0_ AGT  --- 0 HELLO 12 [0 0 0 0] ------- [0:0 -1:0 2 0]
  [80 0 12 [0:0 -1:0]] (MTP_HELLO)
r 1.000000000 _0_ RTR  --- 0 HELLO 12 [0 0 0 0] ------- [0:0 -1:0 2 0]
  [80 0 12 [0:0 -1:0]] (MTP_HELLO)
s 1.000000000 _0_ RTR  --- 0 HELLO 32 [0 0 0 0] ------- [0:0 -1:0 2 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
r 1.001280587 _1_ RTR  --- 0 HELLO 32 [0 ffffffff 0 800] ------- [0:0 -1:0 2 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
r 1.001280587 _1_ AGT  --- 0 HELLO 32 [0 ffffffff 0 800] ------- [0:0 -1:0 1 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
r 1.001280662 _4_ RTR  --- 0 HELLO 32 [0 ffffffff 0 800] ------- [0:0 -1:0 2 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
r 1.001280662 _4_ AGT  --- 0 HELLO 32 [0 ffffffff 0 800] ------- [0:0 -1:0 1 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
r 1.001280667 _3_ RTR  --- 0 HELLO 32 [0 ffffffff 0 800] ------- [0:0 -1:0 2 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
r 1.001280667 _3_ AGT  --- 0 HELLO 32 [0 ffffffff 0 800] ------- [0:0 -1:0 1 0]
  [80 0 32 [0:0 -1:0]] (MTP_HELLO)
s 1.100000000 _1_ AGT  --- 1 HELLO 12 [0 0 0 0] ------- [1:0 -1:0 2 0]
  [80 0 12 [1:0 -1:0]] (MTP_HELLO)
r 1.100000000 _1_ RTR  --- 1 HELLO 12 [0 0 0 0] ------- [1:0 -1:0 2 0]
  [80 0 12 [1:0 -1:0]] (MTP_HELLO)
s 1.100000000 _1_ RTR  --- 1 HELLO 32 [0 0 0 0] ------- [1:0 -1:0 2 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
r 1.101380586 _4_ RTR  --- 1 HELLO 32 [0 ffffffff 1 800] ------- [1:0 -1:0 2 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
r 1.101380586 _4_ AGT  --- 1 HELLO 32 [0 ffffffff 1 800] ------- [1:0 -1:0 1 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
r 1.101380587 _0_ RTR  --- 1 HELLO 32 [0 ffffffff 1 800] ------- [1:0 -1:0 2 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
r 1.101380587 _0_ AGT  --- 1 HELLO 32 [0 ffffffff 1 800] ------- [1:0 -1:0 1 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
s 1.101380587 _0_ AGT  --- 2 HELLO 22 [0 0 0 0] ------- [0:0 1:0 32 0]
  [81 1 22 [0:0 1:0]] (MTP_HELLO)
r 1.101380587 _0_ RTR  --- 2 HELLO 22 [0 0 0 0] ------- [0:0 1:0 32 0]
```

```
  [81 1 22 [0:0 1:0]] (MTP_HELLO)
r 1.101380750 _5_ RTR  --- 1 HELLO 32 [0 ffffffff 1 800] ------- [1:0 -1:0 2 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
r 1.101380750 _5_ AGT  --- 1 HELLO 32 [0 ffffffff 1 800] ------- [1:0 -1:0 1 0]
  [80 0 32 [1:0 -1:0]] (MTP_HELLO)
s 1.107296284 _0_ RTR  --- 2 HELLO 42 [0 0 0 0] ------- [0:0 1:0 30 1]
  [81 1 42 [0:0 1:0]] (MTP_HELLO)
r 1.110197369 _1_ AGT  --- 2 HELLO 42 [13a 1 0 800] ------- [0:0 1:0 30 1]
  [81 1 42 [0:0 1:0]] (MTP_HELLO)

s 5.000000000 _0_ AGT  --- 22 ACL 16 [0 0 0 0] ------- [0:0 1:0 32 0]
  [16 [65 1 1804289383 846930886 1681692777] 16 [0:0 1:0]] (ACL)
r 5.000000000 _0_ RTR  --- 22 ACL 16 [0 0 0 0] ------- [0:0 1:0 32 0]
  [16 [65 1 1804289383 846930886 1681692777] 16 [0:0 1:0]] (ACL)
s 5.000000000 _0_ RTR  --- 22 ACL 36 [0 0 0 0] ------- [0:0 1:0 30 1]
  [16 [65 1 1804289383 846930886 1681692777] 36 [0:0 1:0]] (ACL)
r 5.001529761 _1_ AGT  --- 22 ACL 36 [13a 1 0 800] ------- [0:0 1:0 30 1]
  [16 [65 1 1804289383 846930886 1681692777] 36 [0:0 1:0]] (ACL)
s 5.001529761 _1_ AGT  --- 23 HTTP 20 [0 0 0 0] ------- [1:0 0:0 32 0]
  [1 0 1804289383 1714636915 20 [1:0 0:0]] (HTTP)
r 5.001529761 _1_ RTR  --- 23 HTTP 20 [0 0 0 0] ------- [1:0 0:0 32 0]
  [1 0 1804289383 1714636915 20 [1:0 0:0]] (HTTP)
s 5.001529761 _1_ RTR  --- 23 HTTP 40 [0 0 0 0] ------- [1:0 0:0 30 0]
  [1 0 1804289383 1714636915 40 [1:0 0:0]] (HTTP)
r 5.003435523 _0_ AGT  --- 23 HTTP 40 [13a 0 1 800] ------- [1:0 0:0 30 0]
  [1 0 1804289383 1714636915 40 [1:0 0:0]] (HTTP)
s 5.003435523 _0_ AGT  --- 24 HTTP 84 [0 0 0 0] ------- [0:0 1:0 32 0]
  [2 200 1804289383 1714636916 84 [0:0 1:0]] (HTTP)
r 5.003435523 _0_ RTR  --- 24 HTTP 84 [0 0 0 0] ------- [0:0 1:0 32 0]
  [2 200 1804289383 1714636916 84 [0:0 1:0]] (HTTP)
s 5.003435523 _0_ RTR  --- 24 HTTP 104 [0 0 0 0] ------- [0:0 1:0 30 1]
  [2 200 1804289383 1714636916 104 [0:0 1:0]] (HTTP)
r 5.006253284 _1_ AGT  --- 24 HTTP 104 [13a 1 0 800] ------- [0:0 1:0 30 1]
  [2 200 1804289383 1714636916 104 [0:0 1:0]] (HTTP)
s 5.006253284 _1_ AGT  --- 25 HTTP 20 [0 0 0 0] ------- [1:0 0:0 32 0]
  [1 0 846930886 1714636917 20 [1:0 0:0]] (HTTP)
r 5.006253284 _1_ RTR  --- 25 HTTP 20 [0 0 0 0] ------- [1:0 0:0 32 0]
  [1 0 846930886 1714636917 20 [1:0 0:0]] (HTTP)
s 5.006253284 _1_ RTR  --- 25 HTTP 40 [0 0 0 0] ------- [1:0 0:0 30 0]
  [1 0 846930886 1714636917 40 [1:0 0:0]] (HTTP)
r 5.008419046 _0_ AGT  --- 25 HTTP 40 [13a 0 1 800] ------- [1:0 0:0 30 0]
  [1 0 846930886 1714636917 40 [1:0 0:0]] (HTTP)
s 5.008419046 _0_ AGT  --- 26 HTTP 148 [0 0 0 0] ------- [0:0 1:0 32 0]
  [2 200 846930886 1714636918 148 [0:0 1:0]] (HTTP)
r 5.008419046 _0_ RTR  --- 26 HTTP 148 [0 0 0 0] ------- [0:0 1:0 32 0]
  [2 200 846930886 1714636918 148 [0:0 1:0]] (HTTP)
s 5.008419046 _0_ RTR  --- 26 HTTP 168 [0 0 0 0] ------- [0:0 1:0 30 1]
```

```
  [2 200 846930886 1714636918 168 [0:0 1:0]] (HTTP)
r 5.011628807 _1_ AGT  --- 26 HTTP 168 [13a 1 0 800] ------- [0:0 1:0 30 1]
  [2 200 846930886 1714636918 168 [0:0 1:0]] (HTTP)
s 5.011628807 _1_ AGT  --- 27 HTTP 20 [0 0 0 0] ------- [1:0 0:0 32 0]
  [1 0 1681692777 1714636919 20 [1:0 0:0]] (HTTP)
r 5.011628807 _1_ RTR  --- 27 HTTP 20 [0 0 0 0] ------- [1:0 0:0 32 0]
  [1 0 1681692777 1714636919 20 [1:0 0:0]] (HTTP)
s 5.011628807 _1_ RTR  --- 27 HTTP 40 [0 0 0 0] ------- [1:0 0:0 30 0]
  [1 0 1681692777 1714636919 40 [1:0 0:0]] (HTTP)
r 5.013674568 _0_ AGT  --- 27 HTTP 40 [13a 0 1 800] ------- [1:0 0:0 30 0]
  [1 0 1681692777 1714636919 40 [1:0 0:0]] (HTTP)
s 5.013674568 _0_ AGT  --- 28 HTTP 44 [0 0 0 0] ------- [0:0 1:0 32 0]
  [2 200 1681692777 1714636920 44 [0:0 1:0]] (HTTP)
r 5.013674568 _0_ RTR  --- 28 HTTP 44 [0 0 0 0] ------- [0:0 1:0 32 0]
  [2 200 1681692777 1714636920 44 [0:0 1:0]] (HTTP)
s 5.013674568 _0_ RTR  --- 28 HTTP 64 [0 0 0 0] ------- [0:0 1:0 30 1]
  [2 200 1681692777 1714636920 64 [0:0 1:0]] (HTTP)
r 5.015992330 _1_ AGT  --- 28 HTTP 64 [13a 1 0 800] ------- [0:0 1:0 30 1]
  [2 200 1681692777 1714636920 64 [0:0 1:0]] (HTTP)
s 5.015992330 _1_ AGT  --- 29 ACL 16 [0 0 0 0] ------- [1:0 0:0 32 0]
  [17 [0 0 0 0 0] 16 [1:0 0:0]] (ACL)
r 5.015992330 _1_ RTR  --- 29 ACL 16 [0 0 0 0] ------- [1:0 0:0 32 0]
  [17 [0 0 0 0 0] 16 [1:0 0:0]] (ACL)
s 5.015992330 _1_ RTR  --- 29 ACL 36 [0 0 0 0] ------- [1:0 0:0 30 0]
  [17 [0 0 0 0 0] 36 [1:0 0:0]] (ACL)
r 5.017886091 _0_ AGT  --- 29 ACL 36 [13a 0 1 800] ------- [1:0 0:0 30 0]
  [17 [0 0 0 0 0] 36 [1:0 0:0]] (ACL)

s 20.000000000 _0_ AGT  --- 78 ACL 16 [0 0 0 0] ------- [0:0 1:0 32 0]
  [16 [65 2 1957747793 424238335 719885386] 16 [0:0 1:0]] (ACL)
r 20.000000000 _0_ RTR  --- 78 ACL 16 [0 0 0 0] ------- [0:0 1:0 32 0]
  [16 [65 2 1957747793 424238335 719885386] 16 [0:0 1:0]] (ACL)
s 20.000000000 _0_ RTR  --- 78 ACL 36 [0 0 0 0] ------- [0:0 1:0 30 1]
  [16 [65 2 1957747793 424238335 719885386] 36 [0:0 1:0]] (ACL)
D 20.026045000 _0_ RTR  CBK 78 ACL 36 [13a 1 0 800] ------- [0:0 1:0 30 1]
  [16 [65 2 1957747793 424238335 719885386] 36 [0:0 1:0]] (ACL)
```

El que podem aquí són les traces generades durant la fase de descobriment de plataformes, la primera migració i la segona migració respectivament. Hi ha varis camps que són comuns a totes les traces i que podem arribar a deduir fàcilment, però el que ens interessa més són els camps específics dels diferents tipus de paquets utilitzats durant la simulació.

En el cas dels paquets MTP_HELLO, el primer camp ens indica si es tracta d'un paquet enviat en BROADCAST, en resposta a un missatge en BROADCAST,

o bé es tracta d'un paquet enviat en UNICAST. El segon camp ens indica el nombre d'entrades de la taula de decisió que s'envien. El següent camps indica la mida en bytes del paquet. Finalment, els últims dos camps indiquen l'origen i destí del paquet en format *adreça:port*.

En el cas dels paquets ACL, el primer camp indica si es tracta d'una petició de migració, un informe d'èxit de la migració o un informe d'error. Els cinc següents camps ens ofereixen informació sobre l'agent a transferir (en el cas d'una petició de transferència), així com tipus d'agent, identificador i identificadors únics escollits pel codi, dades i estat d'execució respectivament. Finalment, tenim un camp que ens indica la mida del paquet, i igual que en el cas anterior, informació sobre l'origen i destí del paquet.

En el cas del paquets HTTP, el primer camp indica si es tracta d'una petició o d'una resposta. Si es tracta d'una resposta, el segon camp indica un codi resultant del processament de la petició. El següent camp és un identificador del recurs que s'està demanant o s'està servint en aquest moment. També necessitem números de seqüència per identificar el flux de missatges HTTP, número de seqüència que queda reflectit en el quart camp. I ja per acabar, igual que en els dos cassos anteriors, els últims camps indiquen la mida del paquet i informació sobre el seu origen i destí.

Firmat: Cristian Stefan Tanas

Bellaterra, Juny de 2010

Resum

Avui en dia, estem assistint a una expansió de la tecnologia d'agents mòbils i noves aplicacions basades en aquesta s'estan obrint pas constantment. Les aplicacions han de demostrar la seva viabilitat sobretot en entorns heterogenis i complexos com les xarxes MANET. En aquest projecte es desenvolupa un sistema per simular el comportament dels agents mòbils, ampliant l'actual simulador de xarxa NS2, i també es comprova la viabilitat de l'implementació de l'ETTMA pel triatge de víctimes en situacions d'emergència.

Resumen

Hoy en día, estamos asistiendo a una expansión de la tecnología de agentes móviles y nuevas aplicaciones basadas en esta se están abriendo paso constantemente. Las aplicaciones han de demostrar su viabilidad sobretodo en entornos heterogéneos y complejos como las redes MANET. En este proyecto se desarrolla un sistema para simular el comportamiento de los agentes móviles, ampliando el actual simulador de red NS2, y también se comprueba la viabilidad de la implementación del ETTMA para la selección de víctimas en situaciones de emergencia.

Abstract

Nowadays we are attending an expansion of the mobile agents technology and new applications based on it are arising constantly. The applications need to prove their viability specially in heterogeneous and complex environments such as MANET networks. In this project a new system to simulate the behaviour of the mobile agents is developed by extending the existing network simulator NS2. Also the viability of the implementation of ETTMA for victim classification in emergency situations is proved.