SIMULACIÓ D'ESCENARI DE XARXA DTN AMB NS-2

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Rubén Martínez Vidal i dirigit per Juan Sebastián Pinzón Gámez.

Bellaterra, 17 de Juny de 2010

El firmant, Juan Sebastián Pinzón Gámez , 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 Rubén Martínez Vidal

Bellaterra, 17 de juny de 2010

Firmat: Juan Sebastián Pinzón Gámez

Dedicat a aquells que m'han servit d'inspiració i motivació en aquests últims anys.

Agraïments

Gràcies a Juan Sebastián Pinzón i a Sergi Robles pel seu suport. També vull mostrar el meu agraïment a la gent del grup de recerca SENDA pels seus comentaris i recomanacions.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

1.1 Introducció

Les xarxes DTN (Delay and Disruption Tolerant Networks) es caracteritzen per ser xarxes on el nodes es troben en moviment i per tant presenten una topologia que varia amb el temps. El que diferencia aquestes xarxes d'altres xarxes amb mobilitat de nodes és el concepte d'endarreriments i talls, és a dir, les xarxes DTN pressuposen que hi hauran instants de temps en els que no existirà un camí entre l'origen i la destinació. Per tant es dissenyen amb mecanismes que permeten garantir que les dades arribaran a la destinació tot i patir aquests talls temporals.

Existeixen multitud d'escenaris DTN, variats i amb característiques pròpies, per a cada escenari es solen utilitzar protocols diferents que estan dissenyats per operar eficientment en l'entorn corresponent al seu escenari concret.

En aquest projecte es presenta un escenari on un conjunt de vehicles aeris no tripulats (Unmanned Aerial Vehicle UAV) cooperen per realitzar tasques de vigilància sobre un terreny. Aquests vehicles es troben en moviment constant i per coordinar-se intercanvien informació entre ells. Els nodes de l'escenari formen una xarxa on els nodes interconnectats varien i presenten connectivitat intermitent. És per això que és necessària la utilització d'un protocol d'encaminament per a xarxes DTN.

Hi han diversos protocols que es poden utilitzar, però majoritàriament no són gaire eficients quan s'apliquen a aquest escenari.

1.2 Objectius

Com a resposta a aquestes necessitats es planteja un projecte en el que es dissenyarà una variació d'un protocol d'encaminament i s'estudiarà el seu comportament amb el simulador NS-2. Per a fer-ho ens plantegem els següents objectius concrets.

Primerament analitzar i entendre els conceptes i protocols utilitzats en les xarxes DTN. Un cop estudiats aquests conceptes s'hauran d'utilitzar per dissenyar un protocol que permeti encaminar les dades rellevants de l'escaneig dels UAVs a una de les estacions de terra.

Després d'aquest disseny teòric s'haurà d'estudiar el funcionament del simulador NS-2 i els seus mòduls i fer-lo servir per programar un mòdul que implementi el protocol d'encaminament i permeti fer una simulació de l'escenari proposat, en la que sigui possible observar el rendiment del protocol dissenyat dins d'aquest escenari.

Podem dir que la motivació per realitzar aquest projecte va ser solucionar el problema de l'encaminament per a l'escenari proposat. Considerem que plantejant aquest conjunt d'objectius serà possible arribar a una solució satisfactòria.

1.3 Contingut de la memòria

A continuació s'exposarà breument el contingut dels diversos capítols que composen aquesta memòria.

L'objectiu del següent capítol es situar al lector dintre del context del projecte i proporcionar els coneixements necessaris per comprendre la nomenclatura i els criteris utilitzats en la presa de decisions del projecte. En aquest capítol es veuran conceptes i definicions que s'utilitzen al model de xarxes DTN. A continuació es farà una descripció de l'escenari sobre el que es treballarà en aquest projecte. Seguidament es presentarà i justificarà la selecció del protocol d'encaminament DTN escollit. Finalment, es descriuran les característiques de l'eina utilitzada per a realitzar aquest estudi.

En el tercer capítol es farà un anàlisi de requisits del projecte, identificant tots els requisits funcionals i no funcionals per tal de veure quines son les necessitats que s'ens plantegen. També s'analitzarà la viabilitat del projecte des d'un punt de vista tècnic, operatiu, econòmic i legal.

En el quart capítol es realitzarà un disseny teòric del protocol d'encaminament a utilitzar en la simulació de l'escenari, a continuació es dissenyarà una solució tecnològica que permeti aplicar aquest protocol a l'entorn de simulació i que alhora l'ampliació final satisfaci tots els requisits i restriccions descrits al tercer capítol.

En el cinquè capítol es veurà el conjunt de proves que hem realitzat, s'utilitzarà la nova implementació per realitzar simulacions, i es farà una discussió sobre els resultats obtinguts: avantatges, inconvenients, problemes, millores, etc.

En el sisè capítol es farà una comparació entre la planificació temporal que es va dur a terme a l'inici del projecte i el temps real que ha estat necessari. Finalment a partir d'aquesta distribució temporal es realitza una estimació econòmica del costos totals del projecte.

En el setè capítol es veuran les conclusions que podem extreure de l'anàlisi final dels resultats del projectes, les millores realitzades i les possibles línies d'ampliació futures.

Capítol 2

Xarxes DTN

2.1 Què és una xarxa DTN?

Avui en dia, s'ha estes l'us de les tecnologies sense fils en la connectivitat de dispositius de xarxa. Normalment la interconnexió d'aquests dispositius s'ha fet a traves d'un conjunt de protocols homogenis definits per l'estàndard TCP/IP.

Però existeixen algunes situacions on no es possible el seu ús. Com per exemple: escenaris on hi ha mobilitat i connexió intermitent entre nodes, comunicacions a llarga distancia amb temps de transmissió massa elevats o simplement escenaris amb limitacions d'infraestructura.

Per solucionar els problemes que presenten aquest tipus de situacions es planteja l'us de les xarxes anomenades DTN (Delay and Disruption Tolerant Networks). Les xarxes DTN [3] fan servir un conjunt de protocols que permeten garantir l'entrega de les dades, encara que es produeixin fallades, endarreriments o modificacions en els nodes de la xarxa.

La problematica que intenten resoldre els protocols de xarxa DTN es centra en dos aspectes principals: el transport de dades i l'encaminament.

El transport de dades a les xarxes DTN és un tema pràcticament resolt. Existeixen dos protocols de transport practicament estandaritzats. Són el protocol Bundle i el protocol LPT (Licklider Transmission Protocol).

El protocol bundle [1] és un protocol end-to-end que requereix encaminament. Es podria situar entre la capa d'aplicació i la de transport. Es basa en un principi store and forward, es a dir, les dades son enviades des de l'origen fins al destí pas-

sant per un conjunt de màquines intermitjes. Les dades poden ser emmagatzemades un temps indefinit entre salt i salt.

El protocol LPT [2] és un protocol de comunicacions point-to-point. No requereix encaminament al ser una comunicació directa entre dues màquines. És bàsicament un protocol punt a punt per enllaços amb endarreriments extremadament grans.

L'encaminament en xarxes DTN és una de les qüestions que encara no han estat resoltes. És també un dels temes principals d'aquest projecte. Actualment es pot destacar l'existencia de dos esquemes d'encaminament: *Deterministic Routing* i *Stochastic* o *Dynamic Routing* [4].

L'encaminament determinista implica tenir un coneixement sobre l'estructura de la xarxa abans de realitzar l'encaminament. Existeixen diverses propostes que depenen del grau de coneixement que es té de la topologia de xarxa. A més, aquest encaminament implica que els nodes de la xarxa tinguin un cert grau de predictibilitat. Tot i que, la majoria de xarxes DTN destaquen per tenir una mobilitat de nodes poc predictible. És per això que sol ser necessari fer servir un esquema d'encaminament dinàmic.

Entre els esquemes d'encaminament dinàmic destaquen els *Epidemic Based Schemes*. Es caracteritzen per enviar copies del mateix paquet a tots els nodes de la xarxa que es trobin a l'abast (*broadcast*). Aquest tipus d'esquemes garanteixen que la informació arribarà a la seva destinació, però produeixen un gran nombre de duplicats i consumeixen un gran nombre de recursos de xarxa innecessari.

Per aquest tipus d'esquema destaquen diverses millores com *Estimation Based Approach* que intenten reduir el consum de recursos dels esquemes epidèmics.

Per altra banda existeixen els *Model-Based Schemes* [5]. Són plantejaments on les decisions d'encaminament es fan a partir del coneixement detallat del comportament de l'escenari sobre el que s'apliquen.

2.2 Escenari DTN: Reconeixement aeri amb UAVs

En aquest projecte s'ha estudiat un escenari on un conjunt de vehicles aeris no tripulats (Unmanned Aerial Vehicle) cooperen per realitzar tasques de vigilància sobre un terreny. Aquests vehicles es troben en moviment constant i per coordinarse intercanvien informació entre ells. Formen una xarxa on els nodes son variables

i presenten connectivitat intermitent.

Des d'un punt de vista funcional l'escenari està compost per tres elements: un conjunt d'unitats terrestres mòbils, les unitats de reconeixement aeri i les estacions receptores de terra. Els elements terrestres mòbils actuen com a desencadenants d'esdeveniments. Els UAV generen dades per notificar la presencia d'aquests últims a les estacions receptores situades en terra. Son aquestes dades les que han de ser encaminades a través de la xarxa amb connectivitat intermitent composta pel conjunt dels UAV.

Una de les principals característiques i alhora inconvenient d'aquest escenari és el fet de la connectivitat intermitent entre els nodes. Normalment les xarxes amb mobilitat de nodes es coneixen com a MANET (Mobile AdHoc Networks) [6]. Els protocols d'encaminament d'aquestes xarxes pressuposen en tot moment existirà un camí entre origen i destinació. Tot i que, dinàmic i variable degut a la mobilitat de nodes. En aquest escenari aquesta assumpció no es compleix. És possible que no existeixi cap ruta per on encaminar les dades en un moment determinat. Per tant, és necessari un esquema de xarxa DTN tolerant a aquestes desconnexions esporàdiques i que garanteixi que les dades arribaran al seu destí.

2.3 Protocol d'encaminament

Com s'ha comentat anteriorment l'encaminament en xarxes DTN és un tema obert. No existeix un estàndard que especifiqui quin protocol utilitzar en una xarxa DTN. De fet, cada tipus d'escenari presenta un entorn amb unes necessitats i requisits a complir diferents. És per això que existeixen múltiples protocols d'encaminament que son aplicables a tipus concrets d'escenari DTN (en funció de les seves característiques).

La selecció d'un protocol s'ha fet seguint un criteri que busca un compromís entre el *Delivery Ratio* (Percentatge de paquets que son entregats de forma satisfactòria) i Overhead (quantitat de duplicats d'un mateix paquet que es generen). S'intenta seleccionar un protocol que tingui un *Delivery Ratio* semblant al d'un encaminament epidèmic però sense generar tant *Overhead*.

A l'hora d'escollir un protocol d'encaminament. S'ens plantegen un conjunt d'alternatives, entre elles podem destacar: els protocols PROPHET [7], Spray and Wait [8] o LAROD[9]. Tots ells haurien tenen la capacitat de realitzar l'encamina-

ment d'aquest escenari de forma satisfactòria. Però, es va escollir LAROD perquè era un protocol que havia estat dissenyat de forma especifica per aquest tipus d'escenari. Per tant, presentava un major grau d'eficiència segons els criteris esmentats anteriorment.

2.3.1 LAROD

L'any 2008 es va publicar un document on es plantejava l'escenari dels vehicles aeris no tripulats com a escenari DTN [9]. En aquest document es definia un model de movilitat basat en feromones per als nodes i es un protocol d'encaminament anomenat LAROD [10] (Location Aware Routing for Opportunistic Delay-Tolerant). Aquest és un protocol d'encaminament geogràfic per xarxes de connectivitat intermitent. Combina el principi store and forward (emmagatzemar el paquet en nodes intermitjos fins que la mobilitat de nodes crea un nou camí) amb la propietat *beaconless* (només es consumeixen recursos de xarxa en el moment que els nodes intervenen en l'encaminament de dades). les simulacions del protocol van mostrar que el percentatge d'entrega de paquets estava molt a prop del màxim possible (a nivell dels esquemes d'encaminament epidèmic) però amb un cost de recursos molt baix.

A continuació s'exposa de forma abreujada el principi de funcionament del protocol. Primerament és necessari introduir el concepte de *Custodian Node*. En els protocols que tenen la propietat *Store-and-Forward*, el *Custodian Node* és aquell node que es troba en possessió d'un paquet en un moment de temps determinat. En principi només existirà una copia del paquet en cada instant de temps, aquesta serà emmagatzemada al Custodian Node durant un interval indefinit. Fins a que es pugui transmetre a un altre que passarà a ser el nou *Custodian Node*.

Quan un node genera o emmagatzema un paquet es converteix en *Custodian Node*. Per transmetre aquest paquet el node inicia un *broadcast*. Tots aquells nodes que es trobin dintre del seu abast rebran el paquet. Quan això es produeix tots els nodes que suposin un apropament cap a la destinació es converteixen en *Tentative Custodian*. Cadascun dels Tentative Custodian inicialitza un temporitzador. El temps d'aquest temporitzador es calcula en funció de la proximitat geogràfica d'aquest node a la destinació final i al propi Custodian (depenent del cas). A més, s'afegeix una certa component aleatòria per assegurar que el temporitzador de tots

els *Tentative Custodian* finalitzarà en instants diferents. Aquell *Tentative Custodian* que finalitzi el seu temporitzador primer es converteix en el nou *Custodian*. Llavors iniciarà de nou el proces de *broadcast* i probablement els nodes que fins ara eren *Tentative Custodian* rebran aquesta transmissió. Un cop rebuda descartaran la seva copia del paquet ja que un altre node s'ha convertit en *Custodian*. El proces s'anirà repetint, apropant cada vegada més el paquet a la destinació fins que finalment es entregat.

Aquest procés pot generar duplicats quan algun dels *Tentative Custodian* no rebi el *broadcast* del que s'ha convertit en *Custodian*. Encara que això no es veu com un problema sinó una forma d'explorar nous camins cap a la destinació. A més, en cas que dos copies es trobin només una es continua encaminant cap a la destinació.

El tractament d'aquests duplicats es el següent. Un cop rebut el paquet la destinació inicia un *broadcast* amb la confirmació (ACK). Els nodes que el reben comencen a propagar aquesta confirmació. Si algun node té una copia del paquet que ha estat confirmat el descarta, de la mateixa forma, si la destinació torna a rebre un altre copia d'un paquet que ja ha confirmat també el descarta.

2.3.2 LODIS

LODIS [11] és un servei de localització utilitzat a LAROD. La seva tasca es mantenir un llistat a cada node amb l'última posició coneguda dels altres nodes. La informació es actualitzada i ampliada cada cop que un node entra en contacte amb un altre.

Aquesta informació s'utilitza per prendre les decisions d'encaminament durant el càlcul dels temporitzadors en LAROD.

2.4 Simulador NS-2

Network Simulator [14] és un simulador d'esdeveniments discrets. Està basat en dos components: un simulador orientat a objectes, escrit en C++, i un interprete d'OTcl [12] utilitzat per executar scripts de comandes escrits per l'usuari.

Existeixen dos jerarquies de classes: la compilada amb C++ i la interpretada en OTcl, amb correspondències d'un a un entre elles. La jerarquia C++ compilada

permet aconseguir eficiència en la simulació i uns temps d'execució més ràpids. Aquesta es fa servir per la definició detallada del comportament dels protocols. La jerarquia OTcl es proveïda per l'usuari i es fa servir per definir les topologies de xarxa, els protocols a utilitzar (el comportament dels quals s'ha definit prèviament en la jerarquia C++), aplicacions i paràmetres que es volen simular. D'aquesta forma OTcl fa us d'objectes compilats en C++ fent servir tclCL (Interface Tcl/C++) que crea un objecte Otcl per cada un dels objectes C++.

El protocol LAROD es troba implementat en llenguatge C++ com una extensió del simulador NS-2. Concretament és un versió modificada que inclou un conjunt de noves classes per implementar el protocol. Aquesta implementació va ser realitzada sobre la versió 2.30 del simulador i no funciona amb versions posteriors. Ja que aquestes inclouen una jerarquia de classes lleugerament modificada. Aquesta és una de les restriccions que ha provocat que fem servir la versió 2.30 en lloc de la versió més actual (2.34).

2.4.1 Simulador gràfic NAM

Quan es realitza una simulació amb NS-2 es possible crear uns fitxers de traça amb una sintaxis determinada que permet descriure la posició, moviments i esdeveniments que es produeixen. Aquests fitxers poden ser interpretats per un programa complementari del simulador. Aquest programa s'anomena NAM (Network Animator) i permet visualitzar de forma gràfica (encara que simplista) els esdeveniments descrits per aquest conjunt de fitxers.

Malauradament la implementació de LAROD no és capaç de generar un fitxer de traça correcte (tot i estar implementat). El motiu es desconegut per el propi autor. Degut a això ens hem vist obligats a prescindir d'aquesta aplicació , analitzar les dades resultants fent servir fitxers generats en format text i realitzar el tractament d'aquestes mitjançant eines externes al simulador.

Capítol 3

Anàlisi de Requisits

L'anàlisi de requisits d'aquest projecte s'ha realitzat aplicant la metodologia establerta a l'estàndard ANSII/IEEE [15].

3.1 Requisits Funcionals

Els requisits funcionals descriuen el comportament desitjat del software. Per a realitzar aquest anàlisi de requisits funcionals, utilitzarem com a eina auxiliar un diagrama de casos d'us (veure figura 3.1).

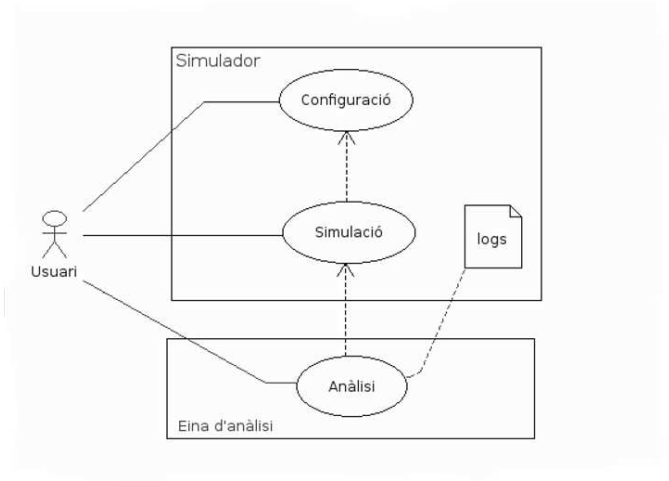

Figura 3.1: Diagrama de casos d'us

A partir del diagrama i observant els casos d'us podem determinar que els

requisits funcionals del projecte són:

3.1.1 Configuració

Introducció

Aquest requisit funcional especifica que l'aplicació ha de donar la possibilitat de configurar tots els paràmetres de les simulacions de forma ràpida i autònoma al propi simulador. Han de poder incloure descripcions dels escenaris amb la seva topologia, els protocols a utilitzar i la possibilitat d'introduir esdeveniments durant la pròpia simulació.

Entrada

Llistat d'objectes a simular juntament amb el conjunt de paràmetres que han de rebre i els seus valors associats.

Processament

Proces de comprovació dels paràmetres. S'ha de comprovar que els objectes indicats per simular existeixen així com també comprovar que els valors assignats al seus paràmetres siguin correctes.

Sortida

De sortida s'ha de generar un conjunt de fitxers de configuració que puguin ser interpretats pel simulador.

3.1.2 Simulació

Introducció

Aquest requisit funcional especifica que l'aplicació ha d'utilitzar els paràmetres establerts durant la configuració per decidir quines operacions realitzar. L'execució d'aquestes operacions serà el que es diu proces de simulació. Proces durant el qual s'ha de generar un conjunt de resultats. Els resultats han de ser fitxers en un format on es pugui observar que ha succeït i que faciliti el seu anàlisi posterior.

Entrada

D'entrada ha de rebre un o més fitxers de configuració, resultants de l'operació de configuració prèvia.

Processament

Interpretar tots els objectes i valors assignats durant la configuració. Un cop interpretat, iniciar el proces de simulació per aquests paràmetres i generar un conjunt de dades resultants. Aquestes dades han de mostrar els esdeveniments produïts en aquest proces de simulació.

Sortida

De sortida es genera un o més fitxers de dades. En un format que pugui ser interpretat posteriorment durant la fase d'anàlisi.

3.1.3 Anàlisi

Introducció

Aquest requisit funcional especifica que s'han de disposar d'eines auxiliars que permetin fer un anàlisi de resultats.

Entrada

Un o més fitxers de dades resultants d'un proces de simulació.

Processament

Conjunt d'operacions de tractament de dades que realitzin comparacions i un conjunt de mesures estadístiques del conjunt de dades introduït.

Sortida

Documents resultants del tractament de dades que mostrin de forma entenedora els valors rellevants en les mètriques desitjades per permetre la presa de decisions i obtenció de conclusions.

3.2 Requisits no funcionals

Per complir aquest conjunt d'objectius funcionals s'utilitzarà com a eina el simulador Network Simulator 2. Això genera un conjunt de necessitats imposades pel mateix projecte i que afecten al disseny.

3.2.1 Requisit d'interfícies externes

Aquests requisits venen definits per la forma d'interactuar amb el simulador Network Simulator 2.

Interfícies d'usuari

L'interfície d'usuari està orientada a un entorn en mode consola.

Interfície Hardware

Teclat estandard.

3.2.2 Requisit de rendiment

A nivell de simulació el temps de resposta no és un factor crític, encara que es desitjable minimitzar-lo. Per garantir aquest fet, totes les operacions que requereixin una capacitat de càlcul alta es realitzen en llenguatge C++, ja que això ens permet assolir uns temps de resposta inferiors.

3.2.3 Restriccions de disseny

La majoria de restriccions d'aquesta secció son resultat de l'utilització del simulador NS2.

Llenguatge a utilitzar

La implementació del projecte s'ha de realitzar en C++ i OTcl. Concretament totes les especificacions dels protocols s'han de realitzar en C++. La configuració de l'entorn i els paràmetres a utilitzar durant la simulació han de realitzar-se mitjançant scripts OTcl.

Jerarquia de classes

NS-2 presenta una jerarquia de classes ja definida amb un conjunt d'interfícies i mètodes virtuals purs que s'han d'implementar quan es defineixen noves classes, en aquest projecte s'haurà de realitzar el disseny respectant aquestes restriccions.

3.2.4 Atributs del software

Escalabilitat i Reusabilitat

És important realitzar un disseny que permeti la modificació i l'ampliació. Concretament al tractar-se d'una simulació d'un protocol que no es troba finalitzat, és important realitzar un disseny que permeti fer modificacions a certes àrees ràpidament, o la inclusió de noves funcionalitats de forma que cap d'aquestes adicions afecti a la resta del disseny.

3.3 Estudi de viabilitat del projecte

3.3.1 Viabilitat operativa

Per executar NS-2 i el mòdul desenvolupat en el projecte no es necessita cap plataforma o hardware especial. Tan sols és requereix un ordinador de propòsit general amb un sistema operatiu no especific com poden ser GNU/Linux o Windows.

Per tant es pot afirmar que els requisits previs del projecte no son gaire elevats i es disposen de tots el recursos necessaris per al seu correcte desenvolupament.

3.3.2 Viabilitat tècnica

NS està programat en C++, proveeix una interfície de simulació a traves del llenguatge OTcl, una versió orientada a objectes del llenguatge d'scripts tcl. L'usuari descriu la topologia de xarxa amb un script OTcl i el programa principal del simulador simula la topologia fent servir un conjunt especific de paràmetres.

Pel desenvolupament del projecte serà necessari implementar diversos script OTcl i les corresponents classes C++. Per poder descriure la topologia de l'escenari i el comportament del protocol respectivament. D'aquesta forma serà possible realitzar la simulació.

3.3.3 Viabilitat econòmica

NS-2 és una aplicació de software lliure per tant no te costos de llicencia, tampoc son necessaris hardware o plataformes fora del normal. Per tant els costos del projecte es limitaran a temps de treball.

El projecte està valorat en 15 crèdits que es corresponen a un total de 300 hores de treball. Fent una estimació comparativa entre diferents convenis laborals per a feines similars, es podria establir un preu de 25 euros per hora de treball, segons això els costos del projecte ascendeixen aproximadament a un total de 7800 euros.

Aquest és un pressupost coherent i acceptable per aquest tipus de projecte per tant el projecte es econòmicament viable.

3.3.4 Viabilitat legal

El projecte no emmagatzema ni realitza el tractament de dades sensibles o de caràcter personal, per tant no es veu afectat per la Llei Orgànica de Protecció de Dades.

El conjunt de programes i protocols que es faran servir al projecte son de codi lliure i s'utilitzaran respectant la seva llicencia en el que respecta a l'us, la modificació i la distribució, de forma que es realitzarà d'acord amb les lleis de propietat intell·llectual.

L'aplicació resultant del projecte no té com a finalitat causar danys materials o morals als sistemes on s'en faci ús. És per això que es pot dir que no es veurà afectada per possibles lleis que regulin delictes tecnològics.

Capítol 4

Disseny i implementació

En aquest capítol s'explicarà el disseny proposat per aquest projecte final de carrera. És necessari destacar que la major part de l'esforç de disseny va ser encaminada al protocol d'encaminament a utilitzar en la simulació de l'escenari. Això es degut a que el protocol d'encaminament és el component principal de la simulació i com a tal és un dels objectius principals d'aquest projecte.

Aquest disseny s'ha realitzat en dos parts. Primerament una vessant teòrica on es va dissenyar el concepte de funcionament i característiques que devia tenir el protocol. Després un altre on es va realitzar el disseny de la solució tecnològica que posteriorment donarà lloc a la implementació del protocol.

4.1 Disseny teòric

En el segon capítol es va exposar que s'havia escollit un protocol anomenat LA-ROD per realitzar la tasca d'encaminament i s'en va descriure el seu funcionament bàsic. En aquest capítol concretarem exactament com es realitza el càlcul del temporitzadors i com varia aquest en les diferents situacions possibles.

Com vam veure anteriorment en aquest protocol els nodes emissors sempre realitzen broadcast i es responsabilitat dels receptors decidir si han de guardar els paquets o descartar-los. Per realitzar aquesta tasca es fa servir un temporitzador. En cas que múltiples nodes rebin el mateix paquet, aquell que el conservarà serà el node a qui li haguí expirat el temporitzador primer.

El protocol intenta garantir que tots els nodes tinguin un temporitzador diferent.

Per realitzar això el càlcul dels temporitzadors es realitza mitjançant la equació de la figura 4.1 .

$$t_d = \begin{cases} \frac{|\vec{r}|}{l} \cdot t_n & \left|\vec{d}\right| \leq l \\ X \cdot t_n & \left|\vec{n}proj\vec{d}\right| \geq l \\ \frac{l - \left|\vec{n}proj\vec{d}\right|}{l} \cdot t_l + t_n & \text{en altre cas} \end{cases}$$

Figura 4.1: Càlcul del valor dels temporitzadors

Els paràmetres utilitzats a l'equació és descriuen a continuació :

- ***l*** : És una constant que s'anomena *Linear End Point*. Representa l'abast de radio aproximat que té cada node.
- ***X*** : És una variable aleatoria que segueix una distribució uniforme amb valors de 0 a 1.
- t_n : És una constant que representa el valor màxim que poden adquirir temporitzadors calculats de forma aletoria.
- t_l : És una constant que representa el valor màxim que poden adquirir temporitzadors calculats de forma lineal.

La resta de paràmetres son vectors variables. Es pot observar una representació gràfica a la figura 4.2.

Aquesta equació te tres parts, cadascuna es correspon amb una situació determinada.

- El primer cas és quan el *Tentative Custodian* es troba a una distancia de la destinació menor que el *Linear End Point*, en altres paraules la primera equació s'utilitza quan la destinació està a l'abast del node.
- El segon cas s'utilitza quan el *Tentative Custodian* es troba fora del rang de radio del *Custodian Node*. Això és un cas poc comú, però a vegades es produeix com a causa de les propietats de propagació de les ones de radio.

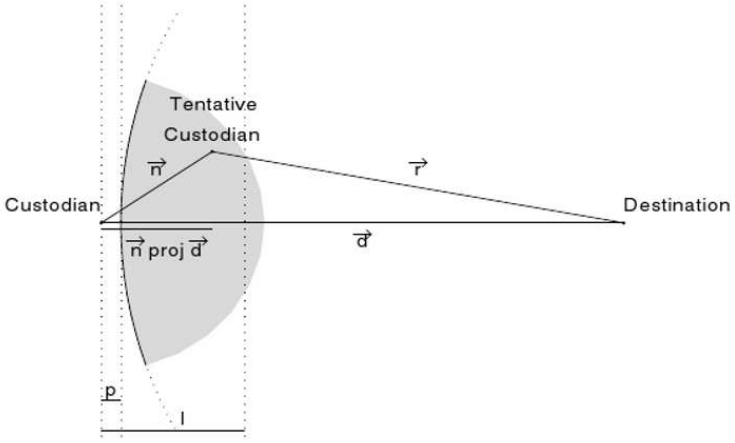

Figura 4.2: Representació gràfica dels paràmetres

- El tercer cas és aquell en que el *Tentative Custodian* es troba dintre del rang de radio del *Custodian Node* i a més no te al seu abast la destinació.

Aquests casos queden il·lustrats a la figura 4.3 seguint el mateix ordre aquí exposat.

4.1.1 Modificacions

L'equació original només considera la distancia relativa dels nodes a la destinació per tal de calcular el valor dels temporitzadors. Això permet la existència d'un cas d'encaminament poc eficient. Suposem que es tenen dos o més *Tentative Custodian*. Un dels nodes es troba més a prop de la destinació però s'esta movent en direcció contraria. Aquest node és el que es converteix en *Custodian*. En contrapartida un altre node que es trobava més lluny però s'estava movent en direcció a la destinació ha estat descartat. Això provoca que el nou *Custodian* al cap de poc temps estigui més lluny que el node descartat. De forma que s'ha produït un error d'encaminament ja que aquesta elecció ha representat un endarreriment en el proces d'entrega.

Es per això que es va considerar com a possible optimització del protocol la utilització de no tan sols la distancia a la destinació, sinó també la direcció en la que s'estan movent els nodes. D'aquesta forma es podria prioritzar als nodes que estiguessin aproximant-se a la destinació en detriment d'aquells que se n'allunyin.

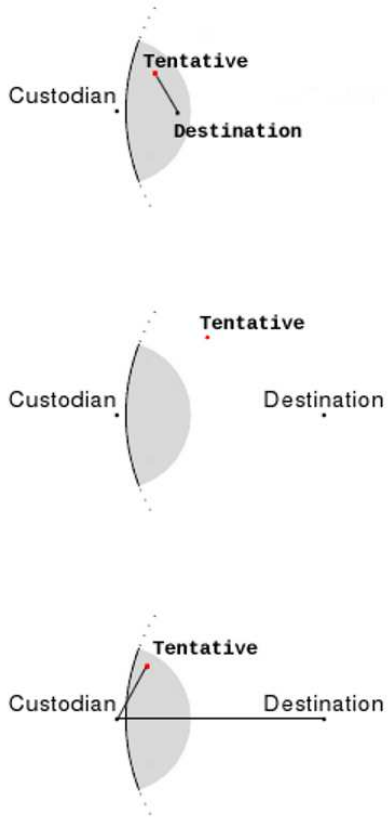

Figura 4.3: Casos d'encaminament

Encara que aquests altres es trobin més a prop en aquell instant de temps determinat.

A continuació es descriu el conjunt de modificacions teòriques que es van realitzar per tal d'aconseguir aquest propòsit.

4.1.2 LODIS

Primerament era necessari disposar d'un mètode per obtenir i emmagatzemar la informació necessària per calcular la direcció en la que s'estaven movent els nodes. El servei de localització LODIS, era una opció molt adequada. LODIS manté un llistat a cada node amb l'última posició coneguda dels altres nodes i d'ell mateix. Per tant realitzant una modificació de LODIS, es podria guardar un historial de les

diferents posicions on es trobaven els nodes en diferents instants de temps.

LODIS realitza actualitzacions periòdiques de coordenades, substituint la informació que es tenia sobre un node per informació actualitzada. L'objectiu de la modificació seria que en lloc de substituir la informació es creí una nova entrada mantenint un cert número de posicions antigues. D'aquesta forma disposarem d'un historial que ens permetrà realitzar el calcul de la direcció en que s'esta movent un node.

L'algorisme s'ha modificat per utilitzar l'historial de punts i fer-lo servir per calcular la direcció del moviment del node. En un primer apropament simple s'utilitzen els dos últims punts, per calcular una recta i fer-la servir com a vector direcció. Encara que la generació de l'historial és dinàmica i la implementació permet altres mètodes més complexes per calcular el vector direcció.

4.1.3 LAROD

Decisió d'encaminament

Vam decidir que es mantindrà el concepte de *broadcast* del Custodian Node i la inicialització d'un temporitzador als *Tentative Custodian*.

Per afegir el concepte de direcció de moviment es fa servir el vector calculat anteriorment pel servei de localització LODIS a partir del seu historial de posicions. Aquest vector direcció del moviment al que a partir d'ara es denominarà vector $\vec{m}$, es comparat amb el vector $\vec{r}$ que defineix la distancia del tentative custodian al destí. Aquest vector era utilitzat a la implementació original de LAROD té com a característica que apunta cap a la destinació.

Mitjançant el producte escalar entre els dos vector s'obté l'angle de separació entre els vectors $\vec{m}$ i $\vec{r}$. A la figura 4.4 es pot veure la representació d'aquest angle.

$$\phi = \arccos \frac{r \cdot m}{|r|\,|m|}$$

El cas ideal és quan l'angle de separació ϕ és zero, el que significa que $\vec{m}$ i $\vec{r}$ tenen la mateixa direcció i per tant el node s'esta movent de forma directa cap a la destinació. Conforme augmenta el valor d'aquest angle que separa els dos vectors menys directa es aquesta aproximació. Quan arriba als 90° el moviment no apropa ni allunya. Un cop supera els 90° significa que comença a allunyar-se fins al cas

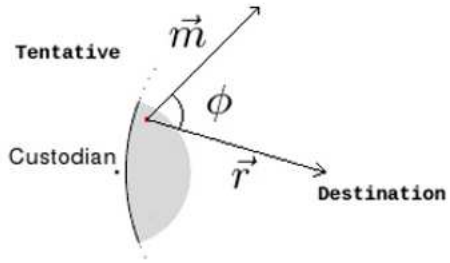

Figura 4.4: Representació de l'angle ϕ

de $180°$ de diferencia que significa que es mou en direcció oposada.

A partir de l'estudi d'aquests casos es va crear un factor direccional (δ) seguint una escala radial i calculat a partir de l'angle que expressava aquestes relacions. Per realitzar el càlcul d'aquest factor s'utilitza la formula:

$$\delta = \frac{(\phi + 10)}{100}$$

L'objectiu d'aquest factor direccional és produir una bonificació o penalització en funció de la direcció en que s'estigui movent el node. L'escala que segueix és simètrica i els valors resultants oscil·len en el rang [0.1 .. 1.9]. Aquest factor direccional s'aplica a l'equació original dels càlculs dels temporitzadors mitjançant una operació de producte.

A la taula 4.1 es pot observar com es relacionen el factor δ i l'angle ϕ.

Cas	ϕ	δ
Aproximació directa	$0°$	0.1
Aproximació nula	$90°$	1
Allunyament directe	$180°$	1.9

Taula 4.1: Relació entre l'angle ϕ i el factor δ

El valor del factor direccional es assignat en funció de l'angle ϕ, d'aquesta forma podem observar com actua en els casos esmentats anteriorment. El valor 0.1 és assignat en el cas on el node s'aproxima de forma directa a la destinació. A l'aplicar aquest factor a la formula de calcul dels temporitzadors, provoquem que

el temps que s'assignarà al temporitzador quedi reduït a un 10% del valor original. D'aquesta forma les probabilitats de convertir-se en nou Custodian augmenten. Per tant el factor direccional actua com una bonificació. En el cas de que s'allunyi en direcció contraria s'aplica un factor de 1.9 per tant el valor del temporitzador queda augmentat en un 190% reduint la probabilitat d'actuar com a nou *Custodian Node*. Per tant en aquest cas es produeix una penalització. Per al cas on la direcció del moviment el manté sempre a la mateixa distancia ($\phi = 90^{\circ}$) el valor que adquireix δ és 1 i per tant no influeix de cap forma en el càlcul del valor dels temporitzadors del protocol.

Ara especificarem com s'ha inclòs el factor δ a l'equació del càlcul dels temporitzadors, l'equació (veure figura 4.1) es troba dividida en tres parts:

La primera es corresponia al cas on es té la destinació dintre de l'abast de radio, vam considerar que en aquest cas no era necessària l'aplicació del factor δ.

El motiu és que el custodian està tant a prop de la destinació que és probable que aquesta ja hagi rebut el paquet durant el *broadcast*. Així que respondrà amb un ACK i els paquets que tinguin els *Tentative Custodian* quedarien descartats. A més en el cas de que la destinació no hagués rebut es prioritzarà el tentative que en aquest moment estigui més a prop. Aquest es convertirà en el nou *Custodian* i començarà a fer broadcast immediatament. Encara que s'estigui allunyant queda pràcticament garantit que aquest cop la destinació rebrà el paquet.

$$\frac{|\vec{r}|}{l} \cdot t_n$$

El segon cas era aquell on un node fora de l'abast del Custodian rebia un paquet. En aquest cas la selecció del temporitzador es realitza d'una forma aleatòria. Per tant, la inclusió del factor δ ens permet eliminar en certa forma l'aleatorietat d'aquest cas i basar-lo en el moviment del node. Fet que pot suposar una considerable millora.

$$X \cdot t_n \cdot \delta$$

El tercer cas és el més comú. És aquell en el que la destinació es troba forà d'abast, però el *Tentative Custodian* sí que es troba a l'abast del *Custodian*. En aquest cas la selecció del temporitzador es realitza mitjançant la distancia del *Ten-*

tative a la destinació. En aquest cas vam considerar que també podia suposar una millora la inclusió del factor δ de forma que també es tingui en compte la direcció del moviment.

En aquest últim cas hem plantejat dos tipus de modificacions. L'equació d'aquest cas conte dos components. El primer utilitza posicions de distancia relativa per calcular el valor del temporitzador. El segon és un valor fixe que equival al màxim factor aleatori que es podia aconseguir a l'equació del segon cas. L'influencia que té el segon component sobre el valor final del temporitzador es bastant menor. Per estudiar com afecten les variacions en aquesta component hem plantejat 2 tipus de modificacions de l'equació:

Una primera on s'aplica el factor δ al component que es correspon a la distancia:

$$\frac{l - \left|\vec{n}proj\vec{d}\right|}{l} \cdot t_l \cdot \delta + t_n$$

I una segona on s'aplica el factor δ al component aleatori:

$$\frac{l - \left|\vec{n}proj\vec{d}\right|}{l} \cdot t_l + t_n \cdot \delta$$

L'equació de càlcul dels temporitzadors modificada es pot observar a la figura 4.5.

$$t_d = \begin{cases} \frac{|\vec{r}|}{l} \cdot t_n & \left|\vec{d}\right| \leq l \\ X \cdot t_n \cdot \delta & \left|\vec{n}proj\vec{d}\right| \geq l \\ \frac{l - \left|\vec{n}proj\vec{d}\right|}{l} \cdot t_l + t_n \cdot \delta & \text{o} \quad \frac{l - \left|\vec{n}proj\vec{d}\right|}{l} \cdot t_l \cdot \delta + t_n \text{ en altre cas} \end{cases}$$

Figura 4.5: Equació de càlcul dels temporitzadors modificada

4.2 Disseny de l'aplicació software

En l'apartat anterior s'ha realitzat un disseny a nivell teòric del protocol on es definia detalladament el seu comportament. En aquesta secció realitzarem un disseny de la solució tecnològica que ens permetrà implementar el protocol definit anteriorment.

En aquesta fase de disseny ens hem trobat limitats per una serie de factors inherents al projecte. Primerament, el fet de partir d'un protocol base impossibilita realitzar un disseny per etapes des de zero. La metodologia aplicada en el disseny està més pròxima a l'etapa d'ampliació i manteniment d'un desenvolupament de software en espiral. On el nostre projecte seria una segona iteració de la versió original amb noves funcionalitats.

Vam partir de la implementació NS-2 del protocol LAROD [13]. Prèviament vam realitzar un estudi de la implementació disponible aplicant enginyeria inversa a partir del codi i la documentació teòrica de la que disposàvem. Un cop vam conèixer els detalls de funcionament de l'aplicació, vam dissenyar el conjunt de modificacions que permetien implementar les modificacions teòriques realitzades al protocol.

4.2.1 LODIS

En aquesta secció tractarem el disseny de les modificacions del sistema de localització de nodes.

Com hem comentat en la secció anterior partim d'una implementació base del protocol, per tant abans de realitzar el disseny de les noves funcionalitats es requereix un estudi de previ del funcionament i jerarquia de classes utilitzada per aquest i a partir d'aquesta decidir un disseny que sigui compatible i permeti expandir la seva funcionalitat.

Les tasques principals del servei de localització son desenvolupades per dos grups de classes, la classe principal **LODIS** que conte el llistat de dades geogràfiques dels nodes i el mètodes per actualitzar i accedir a aquestes dades. Després hi ha un altre classe que s'encarrega de mantenir les dades actualitzades, aquesta classe s'anomena **LodisLocSendTimer**. També hi ha un altre classe que desenvolupa tasques relatives a la simulació, com per exemple l'emmagatzematge de logs. En aquesta secció no entrarem en detalls respecte aquesta tasca ja que no es una

part del protocol pròpiament dit.

A LODIS cada node manté un llistat que conté l'última posició coneguda dels nodes amb els que ha entrat en contacte. Aquesta informació es actualitzada de dues formes. Primerament quan un node rep algun tipus de paquet. Es realitza mitjançant crides al mètode *update_location* que es invocat per la classe corresponent a **LAROD**. El segon mètode d'actualització es realitza de forma periòdica mitjançant un temporitzador. Això es controlat per la classe **LodisLocSendTimer**.

Aquest llistat de nodes es emmagatzemat com una llista d'estructures amb les dades de geogràfiques. Cada element d'aquesta llista es actualitzat cada cop que es realitza una actualització, o afegit en cas de no existir.

La primera modificació que vam realitzar va consistir en modificar aquest llistat de forma que cada cop que s'actualitza no es modifiqui l'entrada, sinó que es crei una de nova i es mantingui l'antiga en forma de llista. D'aquesta forma l'historial de posicions conte una llista d'entrades amb tots els nodes coneguts i cadascuna d'aquestes entrades és una llista amb les posicions conegudes en instants de temps anteriors.

L'altre modificació es realitzada a causa de la necessitat d'incorporar un sistema de càlcul del vector que defineix la direcció del moviment del node, fent servir les dades emmagatzemades a l'historial de punts.

Per realitzar aquesta tasca es proposa una classe anomenada **DirCalcMethod** que té dos tasques. Primerament proveir un mètode que realitzi el càlcul del vector direcció i segon definir la mida màxima que ha de tenir l'historial de posicions conegudes. Mida que estarà limitada per les necessitats del mètode de càlcul.

Aquesta classe **DirCalcMethod** es planteja com una classe interfície que presenta els dos mètodes virtuals purs: *compute_dir_vector* i *get_list_size*, les classes heretades que implementen aquesta classe base han d'implementar aquests dos mètodes de la forma que sigui necessària segons el mètode de càlcul a implementar.

A l'apartat teòric descrivíem un mètode de càlcul d'aquest vector que utilitzava els dos últims punts de l'historial per traçar una recta que era utilitzada com a vector director. Aquest mètode s'ha implementat com a classe derivada amb el nom de **TwoPointLine**, implementa els mètodes *compute_dir_vector* i *get_list_size* utilitzant només els dos últims punts i fent que les llistes de l'historial només tinguin dos punts. Aquesta mida es definida pel métode *get_list_size*.

Aquest esquema de classes s'ha escollit perquè afavoreix l'escalabilitat i per-

met afegir nous mètodes de càlcul de forma ràpida. Tan sols s'ha de crear una nova classe derivada i no influeix de cap manera a cap altre classe. Un altre motiu és el fet que durant la fase de disseny teòric es van estudiar altres mètodes alternatius per realitzar aquest càlcul, però es va escollir el mètode de recta entre dos punts per la seva simplicitat i poc consum de recursos de càlcul. Utilitzant aquest disseny queda oberta la possibilitat d'implementar altres mètodes estudiats en un projecte futur.

A la figura 4.6 es pot observar el diagrama de classes del disseny realitzat, la figura inclou el conjunt de classes del disseny original juntament amb les modificacions introduïdes. Amb l'objectiu de que es pugui observar com, aquest conjunt de noves funcionalitats, ha estat integrat amb l'anterior implementació.

Les classes i mètodes corresponents a la implementació original es mostren en color negre, el conjunt de mètodes i classes que han estat afegits o modificats apareixen en color vermell.

4.2.2 LAROD

En aquesta secció tractarem el disseny de les modificacions del protocol d'encaminament.

Les tasques del protocol d'encaminament es poden dividir en dues parts: gestió de paquets i decisió d'encaminament.

La primera tasca es realitzada per la classe **LAROD** que gestiona la generació, encapsulament, extracció, recepció i enviament de paquets. La segona tasca es realitzada per la classe Forwarder, en aquesta classe es decideix el valor del temporitzador que s'assignara a un *Tentative Custodian* quan aquest rebi un paquet.

La classe Forwarder és una interfície que defineix la serie de mètodes requerits. La implementació original contenia una classe anomenada **LinearRandomForwarder** que realitzava els càlculs dels temporitzadors mitjançant l'equació teòrica original. Vam implementar una segona classe derivada a la que vam anomenar **LinearDirectionForwarder**. A aquesta classe es va afegir un nou mètode anomenat *direction_ratio* per calcular el factor δ. Posteriorment el resultat s'utilitza al mètode *delay_time* que implementa l'equació modificada 4.5 per realitzar el càlcul del temporitzadors.

La classe **LAROD** conte una variable de tipus **Forwarder** que es inicialitzada

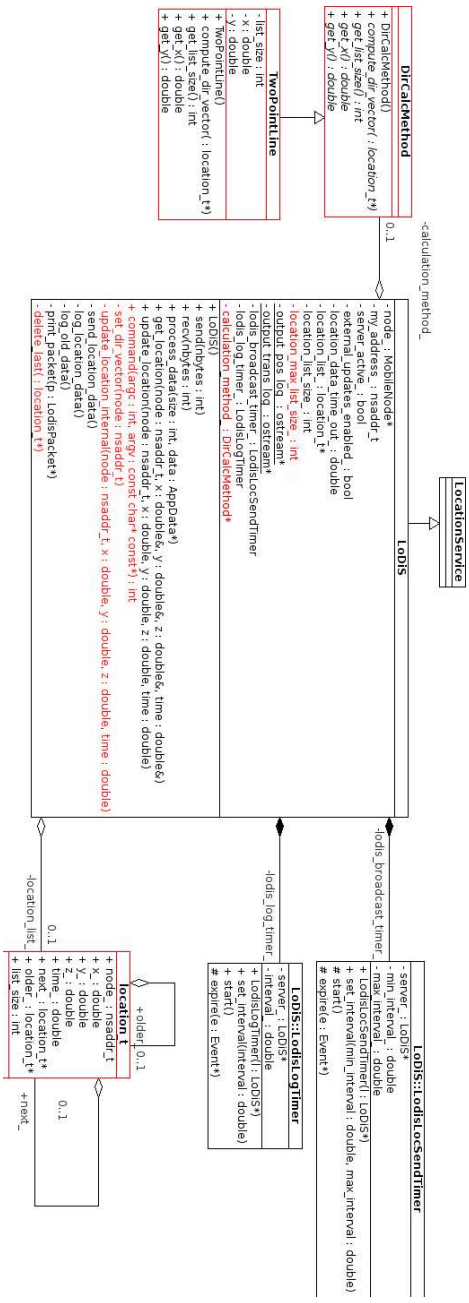

Figura 4.6: Diagrama UML del servei de localització LoDiS

amb un del dos tipus de forwarder de forma que es puguin utilitzar diferents tipus d'encaminament en funció de quina classe s'utilitzi.

Majoritariament les decissions d'encaminament es realitzen quan es rebut un

paquet. La funció que s'executa quan arriba un paquet a algun node es anomenada *recv*. És en aquesta funció on es prenen la major part de les decicions d'encaminament.

Aquesta funció actua de diferents formes depenent de la situació en que es trobi el node. Primerament si la direcció d'origen es la del mateix node s'actua com un router sense més comprovacions ja que el paquet va ser generat pel node en algun moment. Després es comprova si el paquet és de dades o un ACK, si és un paquet de dades es mira qui és el destinatari, en cas de ser el propi node es genera un paquet ACK, en cas contrari ens trobem en la situació que hem estat estudiant durant la fase teòrica, i és aquesta on es realitza la decisió d'encaminament.

En aquest cas es poden donar dos situacions, que sigui un paquet que el node hagi rebut anteriorment o sigui el primer cop que el rebi. En el primer cas significa que algú altre s'ha convertit en custodian i per tant s'elimina la copia local emmagatzemada. En cas de que sigui el primer cop que es rebi el node es converteix en *Tentative Custodian* és en aquest cas on s'invoca al mètode *delay_time* de l'objecte forwarder que es tingui instanciat, d'aquesta forma s'inicialitza el temporitzador que permetrà al node convertir-se en Custodian.

A la figura 4.7 es pot observar la implementació del protocol d'encaminament LAROD.

4.2.3 Comunicació

Per tal de realitzar el càlcul del factor δ i la seva aplicació dintre de l'equació, la classe **LinearDirectionForwarder** precisava de les dades del vector de direcció calculades pel servei de localització. Per tant era necessari dissenyar un mètode d'intercanvi de dades entre el sistema de localització geogràfica i el protocol d'encaminament.

Per realitzar aquesta tasca teníem dues opcions. Primerament, la classe LAROD disposa d'una referencia a un objecte LODIS. Aquesta s'utilitza per actualitzar la posició del node dintre del servei de localització de forma periòdica mitjançant el mètode *update_location()*. La primera opció consistia en crear un nou mètode public en la classe **LODIS** que permetes retornar el valor del vector de direcció del moviment. Això requeriria, a més, un mecanisme de cerca i gestió de múltiples nodes coneguts pel servei, amb l'objectiu de trobar l'adequat i d'aquesta

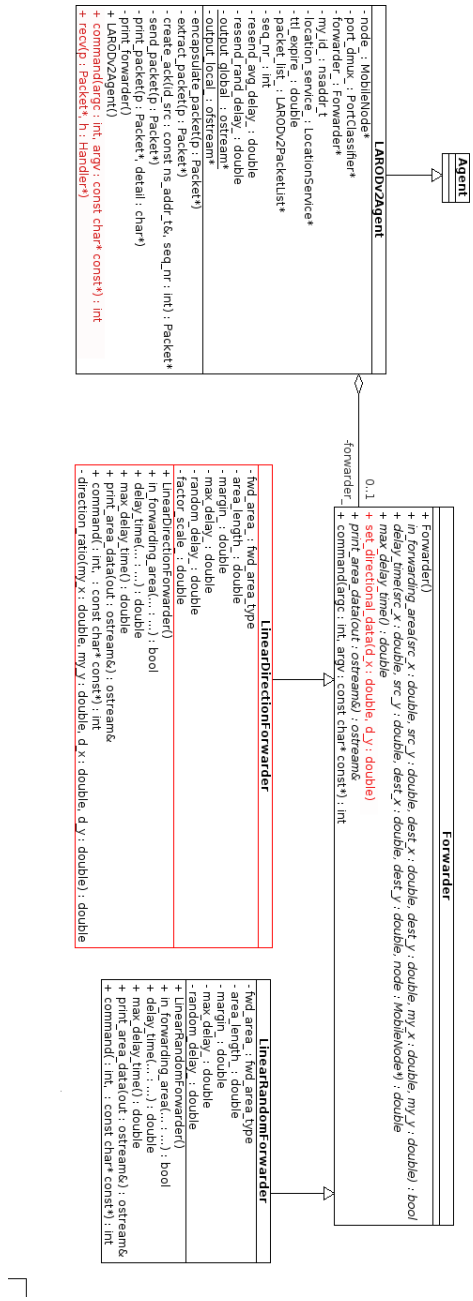

Figura 4.7: Diagrama UML del protocol d'encaminament LAROD

forma poder retornar el valor desitjat.

Per altra banda tant la classe **LAROD** com la classe **LODIS** disposen d'una referencia al node que les està utilitzant en un moment determinat. La classe que

representa el node s'anomena **MobileNode**. La segona opció consistia en afegir a la classe **MobileNode** les variables membre necessàries per emmagatzemar el vector direcció, d'aquesta forma al ser accessible per tots dos serveis. LODIS pot emmagatzemar els valors resultants al propi node i LAROD pot accedir a ell en el moment de prendre les decisions d'encaminament.

Es va escollir la segona opció per ser una implementació més simple i eficient estalviant la creació noves estructures de dades innecessàries. L'altre motiu és el concepte que representa. El propi node emmagatzema la seva posició i direcció. Aquestes dades es veuen gestionades per la classe corresponent al servei de localització. Opinem que a nivell funcional aquest és el concepte més correcte.

A la figura 4.8 es pot observar el diagrama UML corresponent a la comunicació entre classes.

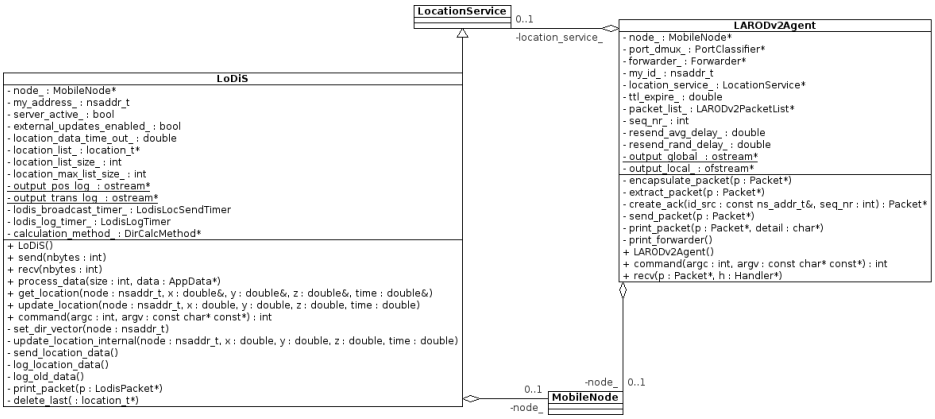

Figura 4.8: Diagrama UML de la comunicació entre classes

4.2.4 Interfície de simulació

Com s'ha explicat anteriorment en el segon capítol NS-2 presenta dos jerarquies de classes: la compilada amb C++ i la interpretada en OTcl. Per definir l'escenari, la topologia, els objectes i protocols que intervenen i els paràmetres d'aquests s'utilitzen scripts OTcl. Es necessari un mètode per poder comunicar els objectes oTcl amb les classes C++ que defineixen el comportament del protocol.

Per afegir els objectes de simulació s'utilitza la correspondència de classes un a un entre C++ i OTcl. Per afegir un nou protocol tal com s'ha fet en aquest projecte és necessari modificar els arxius de codi font del simulador i tornar a compilar.

Aquesta és la jerarquia C++, un cop re-compilat serà possible utilitzar aquestes classes als scripts OTcl que s'utilitzen per simulacions ja que seran reconegudes per l'interpret.

A l'afegir la nostre classe **LinearDirectionForwarder** es fa possible utilitzar aquest tipus d'objecte OTcl en els scripts de simulació. De forma que els scripts OtCl reconeixeran els dos tipus de forwarder que es poden declarar seguint la següent sintaxi:

set uav_forwarder [new Forwarder/LinearRandom];

o

set uav_forwarder [new Forwarder/LinearDirection];

Però per tal d'especificar paràmetres la correspondència d'objectes no es suficient. És per això que a les diferents classes desenvolupades en el projecte s'ha hagut d'implementar una funció anomenada *command()*. La funció command és un *hook* TCL per a totes les classes de ns-2 que permeten la crida a funcions C++ en scripts TCL. Els arguments de *command()* son els mateixos que als programes C bàsics, fent servir argc i argv per rebre les comandes donades a un objecte.

En aquesta funció es realitza una serie de comparacions per detectar quin tipus de comanda s'ha realitzat i s'actua en conseqüència. Nosaltres l'hem utilitzat per rebre dos arguments. Primer al servei de localització LODIS un paràmetre anomenat **calculation-method**. En funció d'aquest paràmetre instanciem el mètode de càlcul amb la classe que s'especifica en l'script OTcl i a partir d'aquesta també queda determinada la longitud màxima de l'historial de punts utilitzat pel servei.

El segon paràmetre que hem utilitzat s'anomena **factor-scale** i s'utilitza a LAROD. A diferencia de l'anterior, la tasca d'aquest paràmetre no afecta al funcionament intern de les classes, sinó que és un paràmetre de simulació. La seva funció es graduar l'escala del factor δ. La possibilitat de graduar els valors límits que pot adquirir aquest factor ens facilita la tasca d'anàlisi del rendiment en la fase de simulació.

Capítol 5

Simulacions i anàlisi de resultats

En aquest capítol veurem el conjunt de proves utilitzat per comprovar el correcte funcionament de la nova implementació. A continuació s'explicarà la metodologia utilitzada i la infraestructura que va ser necessària per realitzar simulacions . Per últim es farà una discussió sobre els resultats obtinguts: avantatges, inconvenients i problemes.

5.1 Fase de proves

Les proves de l'aplicació es van realitzar a tres nivells, primerament a nivell de mòduls a continuació a nivell d'integració i finalment a nivell de sistema.

Vam realitzar proves a nivell de mòdul per al servei de localització i per al protocol d'encaminament.

Les proves del servei de localització buscaven comprovar dues coses. Primerament que es realitzava una correcta gestió de l'historial de punts. Segon que els càlculs del vector de direcció i d l'angle de separació eren correctes.

Per comprovar el correcte funcionament de l'historial de punts vam realitzar proves de *White Box* mitjançant *Data flow Testing*. Concretament vam observar com es modificaven les variables corresponents a les llistes de l'historial de punts. Provocant deliberadament l'execució de diferents casos que requerien diferents tipus de tractament de dades.

Per comprovar si el càlcul dels vectors i els angles era correcte es van realitzar proves de *Black Box*. Donant entrades conegudes i observant que la sortida

coincidís.

Les proves del protocol d'encaminament buscaven comprovar que el nou factor afegit a les formules de càlcul fos correcte. Per realitzar això es van realitzar proves de *Black Box* amb entrades conegudes.

El següent va ser realitzar proves a nivell d'interconnexió de moduls. Per realitzar això es van fer novament proves de caixa negra. Aquest cop per validar s'observava que les sortides estiguessin dintre dels rangs de valors permesos. Això era degut al fet de que no es tenia un control directe de les entrades. Per tant no es podia determinar exactament el valor de sortida exacte.

Per últim es va realitzar la comprovació a nivell de sistema. Per realitzar això es va utilitzar el sistema complet per realitzar un conjunt de simulacions de prova, amb l'objectiu d'assegurar el correcte funcionament sense fallades del sistema i que les dades generades tenien sentit.

5.2 Simulació

Per a realitzar simulacions és necessari utilitzar un script OTcl. Aquest script OTcl descriu els objectes que apareixeran en la simulació, el tipus de protocols que utilitzaran i el conjunt de paràmetres que rebran. Com s'ha comentat a la secció d'interfície de simulació del capítol 4, per utilitzar les noves funcionalitats del protocol només és necessari canviar el tipus d'objectes que es creen en oTCL.

L'script OTcl que utilitzem [13] descriu l'escenari i el model de mobilitat que seguiran els nodes durant la simulació. L'hem modificat per a que tingui la configuració de paràmetres que es mostra a la taula 5.1.

Es simulen 3 hores de l'escenari. Durant la primera hora es deixa circular els UAV sense generar cap esdeveniment de detecció. L'únic trafic que es produeix es el generat per part del servei de localització. Durant tota la segona hora es realitzen deteccions d'unitats de terra i comencen a generar-se paquets de dades. A la tercera hora no es produeixen deteccions, es deixa que els paquets generats anteriorment siguin entregats.

El resultat de cada simulació son dos fitxers amb tot el trafic de paquets produït durant aquestes tres hores. Un primer fitxer conté el trafic produït per part del servei de localització. El segon fitxer té tot el trafic de dades generat pels nodes al realitzar la seva tasca de vigilància i detecció.

Paràmetres	**NS-2**
Àrea	2000x2000 m
Número de nodes	20 a 60
Velocitat UAVs	1.4 m/s i 1.0 a 10 m/s
Temps de vida dels paquets	200-600 s
Interval de generació de paquets	100 s
Abast de radio	250 m
Mètode de càlcul de direcció	TwoPointLine
Forwarder	LinearDirectionForwarder
Factor d'escala de δ	1, 2, 2.5, 3, 5

Taula 5.1: Paràmetres de simulació

Vam realitzar simulacions amb totes les combinacions possibles que es podien realitzar amb els paràmetres descrits anteriorment. L'objectiu era veure com anava variant l'eficiència del protocol al modificar els seus paràmetres.

Amb l'objectiu d'aconseguir valors vàlids estadísticament, vam considerar que eren necessàries un mínim de 100 mesures. Cada simulació de l'escenari triga un temps aproximat de 15 minuts. Per tal d'aconseguir totes les mesures que ens interessaven eren necessàries 2500 hores de simulació. Com això era una tasca pràcticament impossible de realitzar dintre dels terminis del projecte va ser necessari crear una infraestructura hardware per realitzar aquesta tasca.

5.2.1 Infraestructura

Per realitzar les simulacions vam utilitzar un clúster d'ordinadors proporcionat pel departament d'Enginyeria de la Informació i les Comunicacions. Aquest clúster disposava d'un total d'onze ordinadors. Vam distribuir les simulacions entre els ordinadors assignant diferents paràmetres de simulació a cada màquina.

Per realitzar això vam haver de crear un conjunt d'scripts que ens permetessin coordinar i automatitzar aquesta tasca. L'esquema de funcionament era el següent, l'usuari que utilitzàvem tenia un directori de xarxa compartit accessible per a totes les màquines, en aquest directori emmagatzemàvem els scripts OTcl que es desitjaven simular. Cada màquina accedia a aquest directori i executava la simulació de forma loca. Durant tot el proces de simulació anaven emmagatzemant dades en els seus disc durs locals, un cop finalitzades les simulacions els resultats eren recollits

i copiats en el directori compartit per al seu posterior anàlisi.

5.3 Resultats

Al finalitzar cada simulació vam utilitzar un script Python proveït per la implementació original [13]. La tasca d'aquest script consistia en analitzar els fitxers de tràfic i crear un sumari amb dades de xarxa rellevants. Aquestes dades rellevants consistien en valors com el nombre total de paquets generats, nombre de paquets perduts, carrega de xarxa, percentatge d'entrega, etc.

Un cop vam finalitzar les simulacions vam disposar d'uns quants milers de fitxers de sumari, per tal de poder estudiar aquest conjunt de resultats, vam crear un programa en MATLAB capaç d'interpretar el format dels fitxers de sumari, un cop introduïts tots els nostres fitxers de sumari, ens permetia realitzar una representació gràfica molt més fàcil d'interpretar.

Amb aquest programa vam generar totes les taules i gràfiques que es poden veure en l'apèndix A d'aquesta memòria.

5.4 Conclusions

Quan s'analitza un protocol d'encaminament de xarxa DTN el que ens interessa observar es el seu percentatge d'entrega i la carrega de xarxa que produeix. A continuació realitzarem un anàlisi del protocol seguint aquests criteris.

Primerament parlarem del delivery ratio. Hem observat que el valor del delivery ratio depèn principalment de tres factors: la densitat de nodes, el temps de vida dels paquets i la velocitat.

Si creix la densitat de nodes el delivery ratio o percentatge de xarxa augmenta. Això es degut al fet de que al haver-hi més nodes s'incrementa la cobertura que es té sobre la totalitat del terreny i es més fàcil encaminar les dades. Si s'augmenta el temps de vida dels paquets es dona més temps per realitzar l'entrega i per tant paquets que s'haurien descartat ara poden ser entregats. Per últim augmentar la velocitat augmenta la possibilitat de trobar-se amb altres nodes i per tant també incrementa el delivery ratio.

En la figura 5.1 es pot observar una representació del delivery ratio. L'eix de les X es correspon a la densitat de nodes. L'eix de les Y mostra el delivery ratio.

Aquesta gràfica s'ha realitzat amb un temps de vida de 400 segons i una velocitat de 1.4 m/s. Es pot observar com es confirma l'afirmació de l'anterior paràgraf. Per observar com varia el delivery ratio en funció de la velocitat i el temps de vida dels paquets veure l'apèndix A d'aquesta memòria.

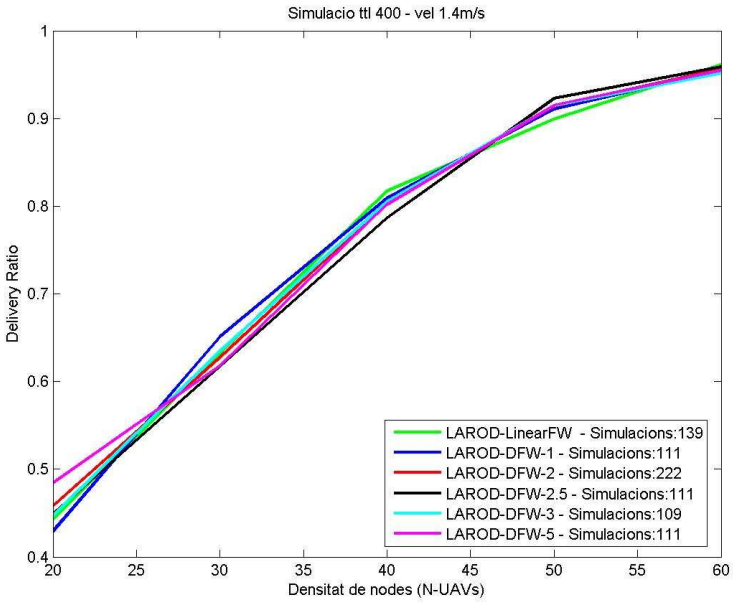

Figura 5.1: Representació gràfica del percentatge d'entrega

Ara parlarem de l'overhead o carrega de xarxa, aquest és un valor que es veu afectat pels mateixos factors però de forma diferent, a diferencia del delivery ratio que va augmentant cada cop més de forma directament depenent d'aquest factors, l'overhead presenta una corba de creixement i decreixement amb un màxim en una certa densitat de nodes concreta que varia en funció del temps de vida. El motiu d'aquestes variacions es que si hi han pocs nodes, hi ha poc contacte entre nodes i per tant no es produeixen gaires transmissions. Això va augmentant fins un punt on es produeix bastant contacte entre nodes però no prou per entregar els paquets ràpidament a la destinació. Per tant el paquet va circulant entre nodes generant trafic. Si s'augmenta més la densitat l'overhead disminueix degut a que ara sí que hi ha prou connectivitat per entregar el paquet ràpidament. El temps de vida influeix en el temps que està el paquet generant trafic. La velocitat augmenta el nombre de contactes que es produeixen per tant també influeix.

En la figura 5.2 es pot observar una representació de l'overhead. Fa servir la mateix configura de paràmetres que la figura esmentada anteriorment, per observar més dades veure l'apèndix A d'aquesta memòria.

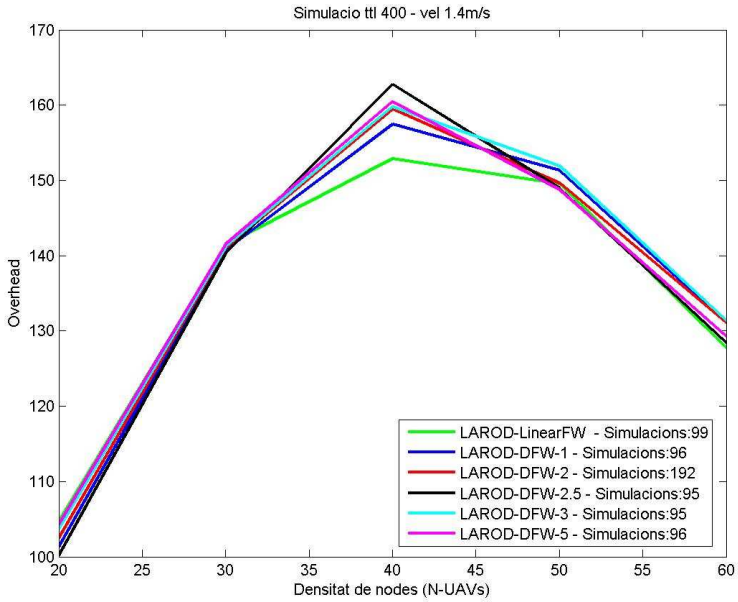

Figura 5.2: Representació gràfica de la carrega de xarxa

Un dels objectius de la simulació era realitzar una comparació de l'eficiència del nou protocol respecte a l'original. Anteriorment hem comentat com afecten els diferents paràmetres a la simulació. Ara comentarem les diferencies de rendiment entre la nostra modificació i el protocol original.

Aquesta comparació la vam realitzar en funció del delivery ratio i de l'overhead. Per realitzar això vam utilitzar mesures del protocol original i mesures del nou protocol. Durant la fase de disseny es va incloure un paràmetre que ens permetia modificar l'escala del factor δ (Veure secció 4.2.4). Com s'ha comentat anteriorment en el llistat de paràmetres hem fet servir 5 d'aquests valor d'escala per fer les simulacions, per tant la comparació es realitzarà entre el protocol original i 5 versions del nostre protocol. Això es pot observar a les figures 5.2 i 5.1 cada línia representa un protocol.

La primera conclusió a la que vam arribar era que el factor d'escala tenia poc impacte sobre els resultats obtinguts. Les 5 versions del nostre protocol tenien

comportaments molt semblants. Sembla que modificar la quantitat de bonificació o penalització que se li aplica a un timer no es gaire rellevant. Tan sols el fet de penalitzar o bonificar.

La següent conclusió va ser que el nou protocol presentava un comportament pràcticament idèntic al del protocol original tant en overhead com en delivery ratio. Només presentava alguna millora en certs casos aïllats normalment amb densitats de node més baixes. Però en contrapartida es perdia eficiència en diversos casos. Creiem que això es degut a les característiques de mobilitat i poca predictibilitat inherents de l'escenari, característiques que van superar els valors que s'havien previst inicialment durant el disseny i per tant, tot i que en uns casos i conjunts aïllats pot ser millor, no té un gran impacte sobre el conjunt global.

Capítol 6

Planificació i costos

En aquest capítol es presenta la planificació temporal de les tasques realitzada a l'inici del projecte en comparació amb el que s'ha produït realment. Juntament amb una valoració econòmica de costos.

6.1 Tasques

La planificació del temps es va dividir en tres grans blocs :

- El temps reservat per tasques de documentació com l'informe previ o la memòria.
- El temps de formació va ser el temps que es va dedicar a estudiar les teories sobre xarxes DTN, familiaritzar-se amb l'entorn de simulació i al disseny teòric del protocol.
- El temps de desenvolupament va ser tot el temps que es va fer servir per implementar el codi necessari per realitzar la simulació , les diverses tasques per comprovar el seu correcte funcionament i l'anàlisi dels seus resultats.

Els terminis s'han pogut acomplir pràcticament sense desviar-se de la planificació inicial, tant sols es va produir una desviació important a les tasques d'implementació i anàlisi de resultats. Per la tasca d'implementació s'havia previst una durada aproximada de 40 dies i tant sols van ser necessaris 10. Per altra banda per la tasca de simulacions s'havien previst 15 díes i van ser necesaris gairebé 50.

Encara que gracies al fet d'haver finalitzat molt més aviat la tasca d'implementació el retard produït a la tasca de simulacions no va suposar un impacte greu en l'acompliment dels terminis.

A la figura 6.1 es mostra un diagrama de gantt on es compara la planificació original del projecte i el temps real necessari per realitzar el projecte.

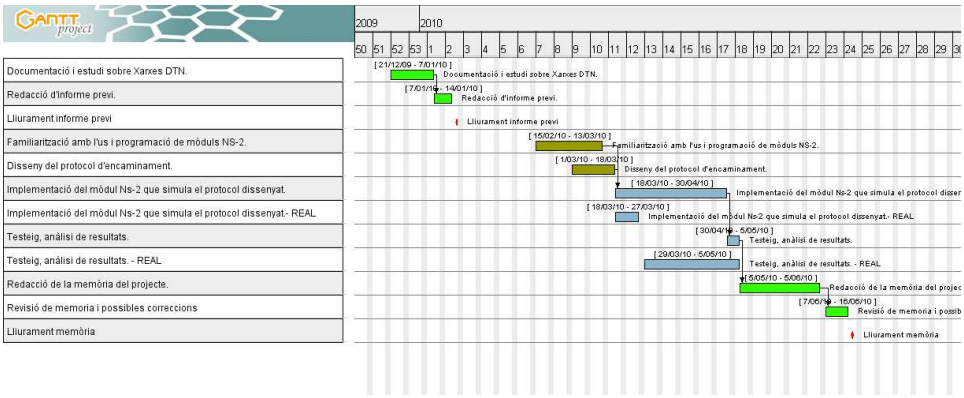

Figura 6.1: Diagrama de Gantt

6.2 Costos

A continuació es presenta el cost que s'estima que ha tingut el projecte realitzat. Aquest projecte ha consistit en realitzar una implementació d'un protocol, simular-lo en un entorn donat i estudiar les dades resultants, un volum gran d'informació complexa que posteriorment s'havia d'analitzar.

Per realitzar el pressupost prenem com a base el salari del personal del Grup 1 (Titulació universitària superior) del conveni de treball de la UAB [16]. En aquest conveni el preu per hora de treball son 26 euros.

A la taula 6.2 es mostra una estimació comparativa de costos.

	Hores			Costos		
Tasca	Previst	Real	Δ hores	Previst	Real	Δ cost
Formació						
Seminaris SENDA	-	25h	-	-	650	-
Curs PFC DEIC	-	30h	-	-	780	-
Lectura articles	15h	15h	-	390	390	-
Familiarització amb simulador NS-2	50h	20h	-5h	650	520	-130
Disseny Teòric	15h	5h	-10h	390	130	-260
Documentació						
Informe Previ	-	10h	-	-	260	-
Memòria	50h	75h	+25h	1300	1950	+650
Desenvolupament						
Familiarització amb el codi anterior	-	15h	-	-	390	-
Disseny	20h	10h	-10h	520	260	-260
Implementació	50h	20h	-30h	1300	520	-780
Testeig	5h	5h	-	130	130	-
Simulacions	20h	50h	+30h	520	1300	+780
Anàlisi de resultats	5h	15h	+10h	130	390	+260
Totals	-	295h	-	-	7670	-

Taula 6.1: Costos en Euros

6.3 Resum

S'han presentat les planificacions tant de cost com de temps i s'han comparat amb els valor reals que s'han produït. Cal destacar que s'han pogut complir els terminis marcats, tot i que s'ha produït algunes desviacions. En referencia al cost s'ha vist que no es tracta d'un cost excessivament elevat i que els costos estan repartits d'una forma bastant equitativa entre les tres tasques principals.

Capítol 7

Conclusions

En aquest capítol es realitzarà una valoració del projecte, exposarem un resum de la feina realitzada, comentarem els resultats del projecte, els objectius assolits i finalment exposarem possibles línies de continuïtat que es podrien realitzar per ampliar el projecte.

7.1 Treball realitzat

El principal objectiu del projecte era realitzar el disseny i la simulació d'un protocol, aplicat a un escenari de xarxa que presentava les característiques d'una xarxa DTN. Això requeria el disseny d'un protocol d'encaminament que pogués funcionar correctament en un entorn d'aquest tipus. A més teníem la condició de que aquesta simulació s'havia de realitzar utilitzant una eina especifica, el simulador de xarxes NS-2.

NS-2 és un entorn complex i bastant extens, tant en utilització com en funcionament intern, el projecte requeria realitzar modificacions importants sobre el seu funcionament intern, i per tant s'havia de realitzar un estudi i familiarització prèvia amb els seus diferents mecanismes de funcionament i utilització.

Per poder assolir aquests objectius vam haver de realitzar un estudi sobre la teoria i conceptes que defineixen les xarxes DTN, les seves característiques, la seva problemàtica, i d'aquests problemes quins són els que ja es troben resolts i aquells que encara no ho estan.

Aquests coneixement adquirits es van utilitzar per analitzar l'escenari presentat

i seleccionar un protocol adequat que garantís un correcte funcionament i rendiment.

Posteriorment ens vam plantejar com a tasca realitzar una modificació sobre el protocol escollit, després d'analitzar amb deteniment el funcionament del protocol, vam cercar algun possible aspecte que tingues carències i vam trobar alguns casos on el rendiment del protocol es podia millorar, tenint en compte això vam dissenyar una millora teòrica que consideràvem podia corregir aquesta carència.

A continuació, després de finalitzar aquesta fase que podríem anomenar d'estudi, anàlisi i disseny teòric, vam realitzar un anàlisi de requisits de l'aplicació, tenint en compte tot el conjunt de funcionalitats que requeria i el conjunt de restriccions de disseny imposades, majoritàriament per la utilització del simulador NS-2 com a nucli de l'aplicació.

Un cop realitzat l'anàlisi de requisits vam començar el disseny de l'aplicació, la major part de l'esforç de disseny es va centrar en el disseny del mòdul corresponent al protocol d'encaminament, vam utilitzar com a base una implementació prèvia d'aquest protocol, per això primerament era necessari estudiar l'estructura i funcionament d'aquesta implementació. A partir d'això vam realitzar el disseny de les millores respectant i buscant la compatibilitat amb l'anterior implementació, un cop fet això i fent servir el nou disseny vam modificar la implementació i vam crear una nova versió del protocol que podia ser executada com com a part del nucli de simulació NS-2.

Posteriorment va ser necessari comprovar el correcte funcionament de la implementació. Vam aplicar una metodologia de testeig per mòduls, on es comprovaven que les sortides obtingudes es trobessin dintre del conjunt dels rangs de valors vàlids. Un cop superades aquest conjunt de proves es va unir tot el conjunt de mòduls i vam procedir a la prova de l'aplicació completa realitzant diverses simulacions per generar un conjunt de dades resultants. Per últim, després d'obtenir resultats coherents es va donar per finalitzada la fase de proves.

Finalment i després d'assegurar el correcte funcionament de la nova implementació, vam preparar el conjunt de paràmetres requerits per simular aquest escenari en forma de fitxers, llavors vam iniciar el proces de simulació. Per realitzar això vam preparar una infraestructura que ens permetia paral·lelitzar les simulacions i automatitzar el proces mitjançant el suport d'un clúster format per 11 ordinadors. Posteriorment vam crear un mètode que permetia tractar d'una forma semi-

automatitzada els fitxers de dades resultants d'aquestes simulacions, per obtenir finalment valors estadístics i gràfiques que ens permetien realitzar un anàlisi de resultats.

7.2 Assoliment dels objectius

En la secció anterior hem exposat el conjunt d'accions que s'han anat realitzant per assolir el conjunt d'objectius, podem afirmar que hem assolit completament els objectius del projecte sense complicacions. Hem adquirit un bon coneixement sobre xarxes DTN. Hem realitzat el disseny i la implementació d'un nou protocol. I finalment, hem realitzat la simulació d'aquest escenari aplicant el protocol dissenyat.

Com a conclusió es pot dir que aquest projecte m'ha permès adquirir un bon coneixement sobre conceptes de xarxes DTN. M'ha donat l'oportunitat de familiaritzar-me amb el simulador de xarxes NS-2, una eina molt potent i amb molts usos. També s'ha de dir que ha estat molt instructiu per conèixer la metodologia utilitzada en temes de recerca.

7.3 Resultats

Com resultats hem desenvolupat una aplicació que respecta les restriccions imposades i segueix l'esquema de funcionament plantejat durant la fase d'anàlisi.

Aquesta aplicació permet realitzar simulacions de l'escenari que es va proposar utilitzant diferents paràmetres. A més, també inclou eines d'anàlisi que ens permeten comprovar l'eficiència del nou protocol d'encaminament que vam dissenyar i realitzar comparacions amb altres protocols.

En el que respecta al nou protocol dissenyat es pot dir que la implementació final no va complir totes les expectatives inicials. Es preveia que la seva eficiència fos superior a la del protocol model que s'havia utilitzat, però després d'analitzar el resultats vam observar que no s'havia produït una millora significativa respecte a l'original, tot i presentar millors resultats en alguns casos concrets.

El disseny del projecte s'ha realitzat tenint en compte l'escalabilitat i la possibilitat d'expansió, per tant aquestes propostes son factibles i no seria necessari

realitzar modificacions en el disseny actual. Aquest tipus de disseny facilita la creació de línies de continuitat.

7.4 Línies de continuïtat

Com a línies de continuïtat del projecte es presenten dues, ambdues línies van encaminades a millorar el protocol d'encaminament.

Una primera línia de continuïtat és la de trobar una demostració empírica de les conclusions a les que s'ha arribat sobre l'eficiència del protocol. El nou objectiu seria demostrar que si aquest protocol s'apliques a un escenari diferent, un escenari on la mobilitat i trajectòries fossin més estables, l'eficiència es veuria augmentada considerablement. Aquesta línia requeriria modificar l'escenari creant un nou model de mobilitat per als nodes o bé aplicar el protocol a un altre escenari completament diferent.

La segona línia de continuïtat és més ambiciosa, consistira en superar la inestabilitat de moviment inherent del propi escenari. L'objectiu que es proposa és modificar de nou el protocol d'encaminament, utilitzant com a nou concepte la tendència del moviment i no pas l'última direcció coneguda.

Aquesta modificació implicaria utilitzar un model de mobilitat complexe basat en feromones, proveït per l'escenari però que no vam utilitzar degut a la gran capacitat de càlcul requerida. Utilitzant com a suposició que els nodes tendeixen a moure's cap a zones inexplorades, i que per tant aquella zona no explorada que estigui propera a la trajectòria calculada prèviament, serà probablement el lloc cap a on s'encaminarà el node. Seria necessària realitzar una comunicació entre els protocols d'encaminament, el servei de localització geogràfica i el model de mobilitat. Gracies a això el servei de localització disposaria d'accés al mapa de feromones, el que li permetria corregir les trajectòries calculades utilitzant la suposició descrita prèviament. D'aquesta forma seria possible millorar l'eficiència del protocol ja que les direccions del moviment serien molt més acurades.

Apèndix A

Dades de simulació

A continuació s'exposen el conjunt de resultats de les simulacions en format gràfic.

Les gràfiques estan dividides en 2 grups:

- Percentatge d'entrega (Delivery Ratio).
- Carrega de xarxa (Overhead).

A les gràfiques es mostren 6 classes de dades diferents. La primera (color verd) es correspon amb el protocol original, les altres 5 representen el nostre nou protocol amb diferents factors d'escala (Veure capítol 5).

Per cada grup de gráfiques es mostren 2 conjunts diferents , cada conjunt es correpon amb una de les variacions de l'equació mostrada a la figura 4.5 (Veure Capítol 4).

El primer conjunt correspon a l'equació:

$$\frac{l-|\vec{n}proj\vec{d}|}{l} \cdot t_l + t_n \cdot \delta$$

El segon conjunt correspon a l'equació:

$$\frac{l-|\vec{n}proj\vec{d}|}{l} \cdot t_l \cdot \delta + t_n$$

A les gràfiques es representen els paràmetres de la taula 5.1. Per classificar les gràfiques s'han fixat dos paràmetres: la velocitat de nodes i el temps de vida dels

paquets. Per veure com afecten aquests dos paràmetres es necessari fer comparacions entre gràfiques, la resta de paràmetres son observables en una sola gràfica.

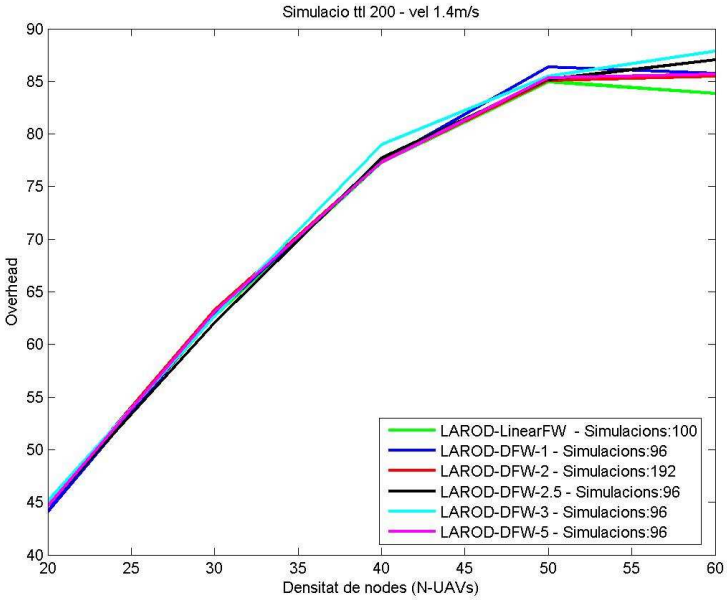

Figura A.1: Overhead - Conjunt 1: Velocitat 1.4 m/s - TTL = 200 s

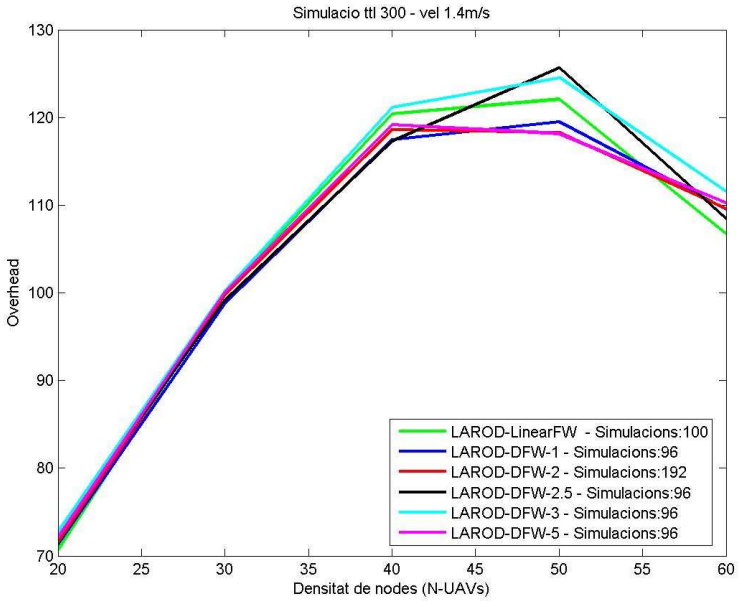

Figura A.2: Overhead - Conjunt 1: Velocitat 1.4 m/s - TTL = 300 s

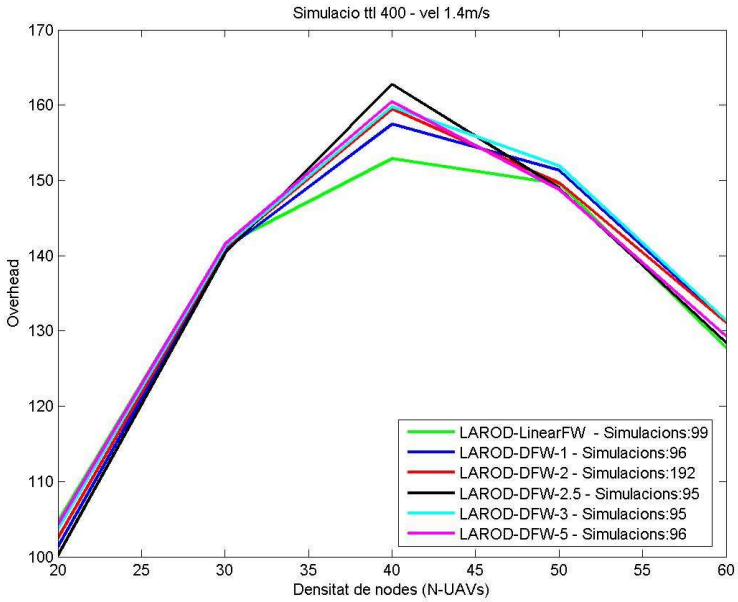

Figura A.3: Overhead - Conjunt 1: Velocitat 1.4 m/s - TTL = 400 s

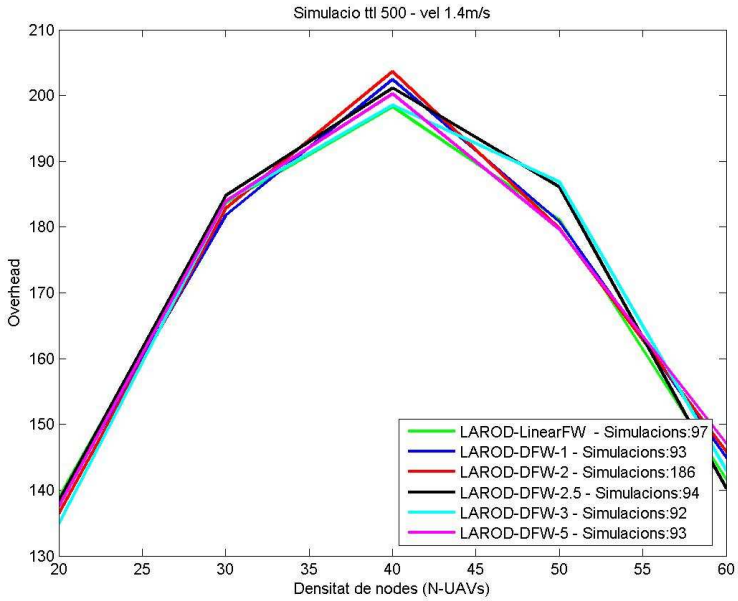

Figura A.4: Overhead - Conjunt 1: Velocitat 1.4 m/s - TTL = 500 s

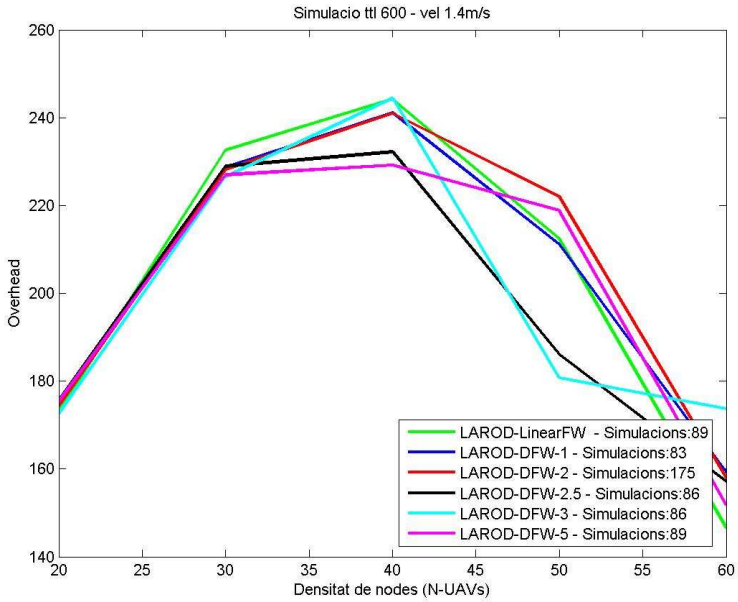

Figura A.5: Overhead - Conjunt 1: Velocitat 1.4 m/s - TTL = 600 s

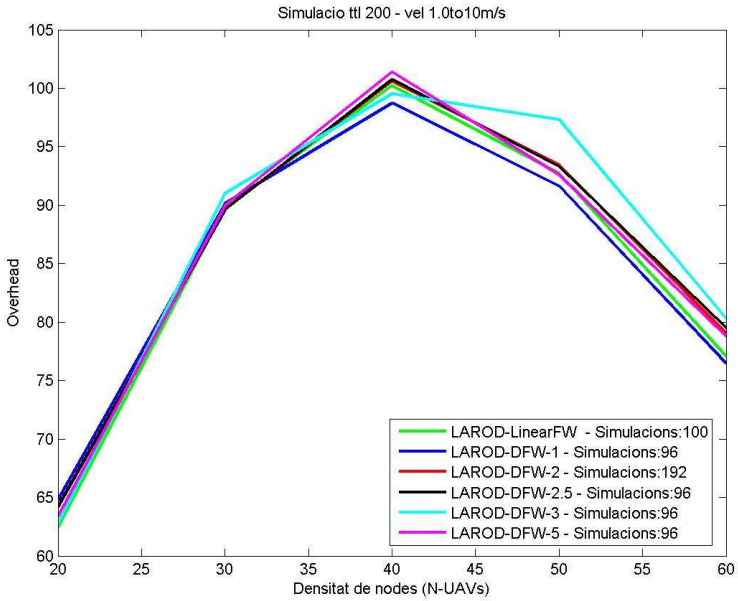

Figura A.6: Overhead - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 200 s

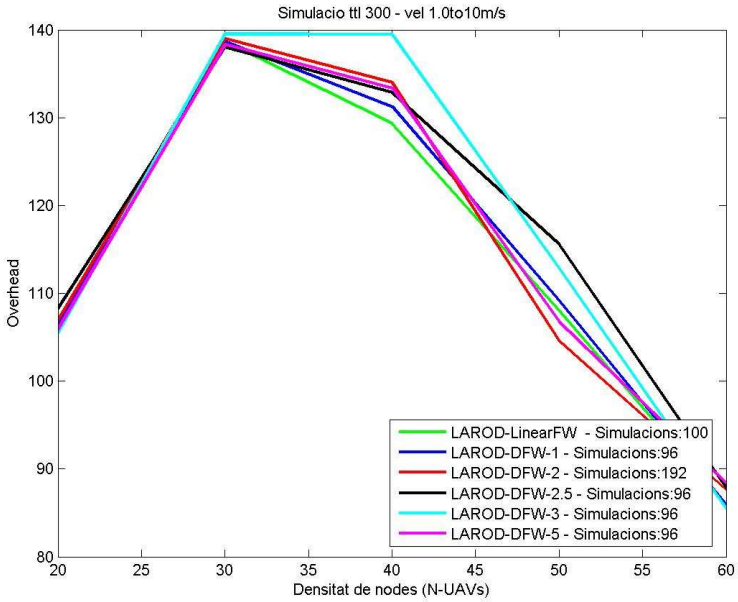

Figura A.7: Overhead - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 300 s

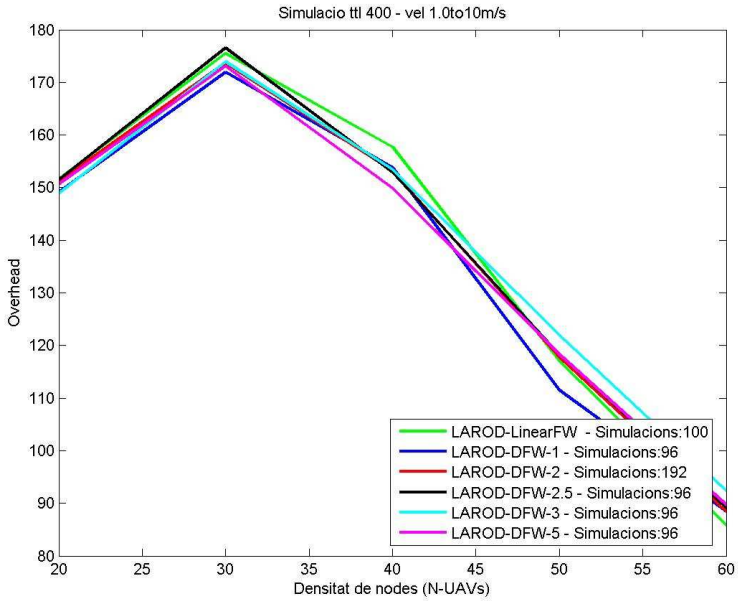

Figura A.8: Overhead - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 400 s

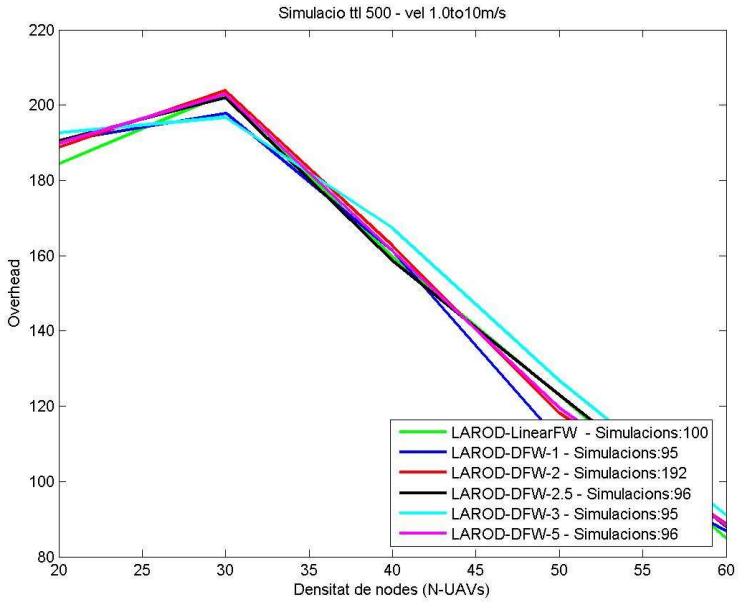

Figura A.9: Overhead - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 500 s

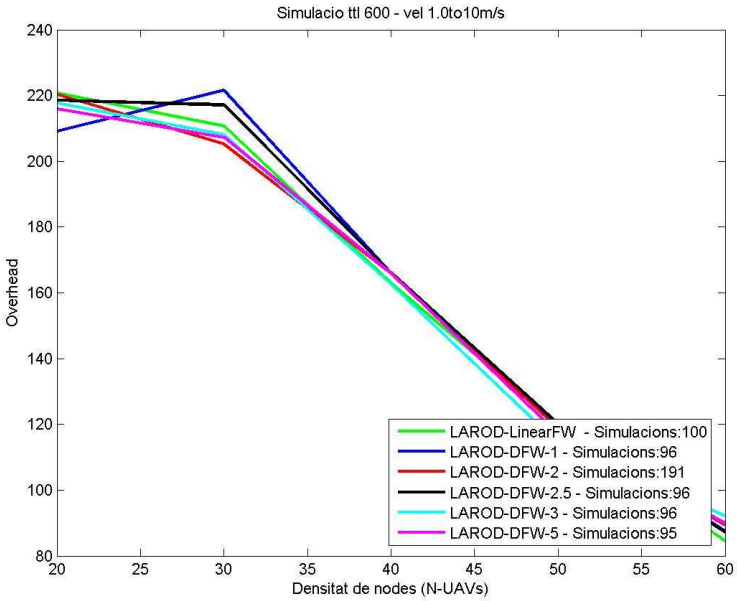

Figura A.10: Overhead - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 600 s

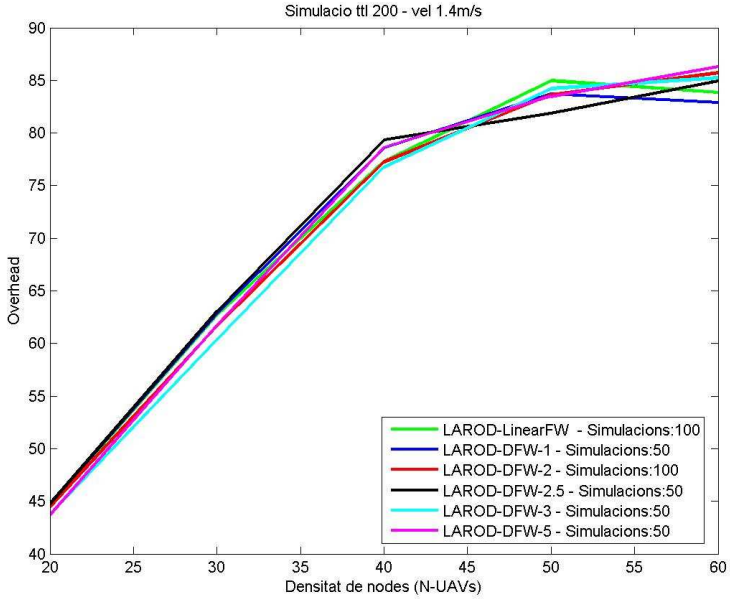

Figura A.11: Overhead - Conjunt 2: Velocitat 1.4 m/s - TTL = 200 s

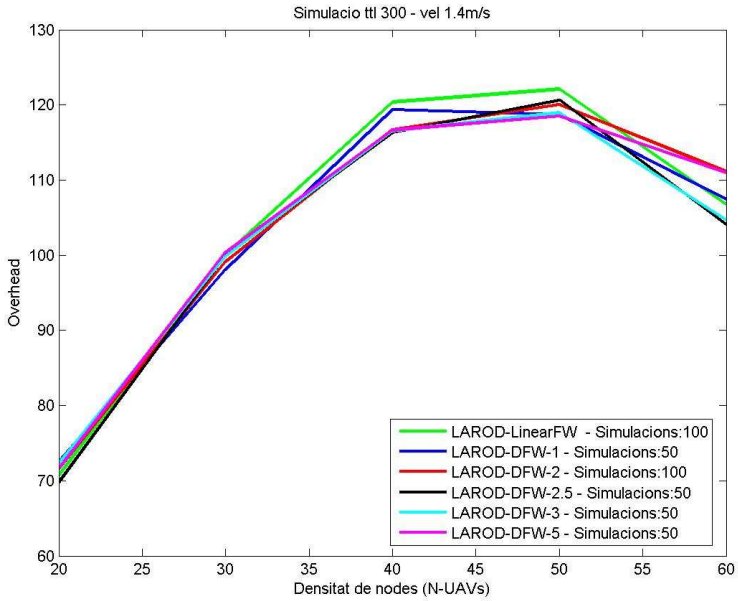

Figura A.12: Overhead - Conjunt 2: Velocitat 1.4 m/s - TTL = 300 s

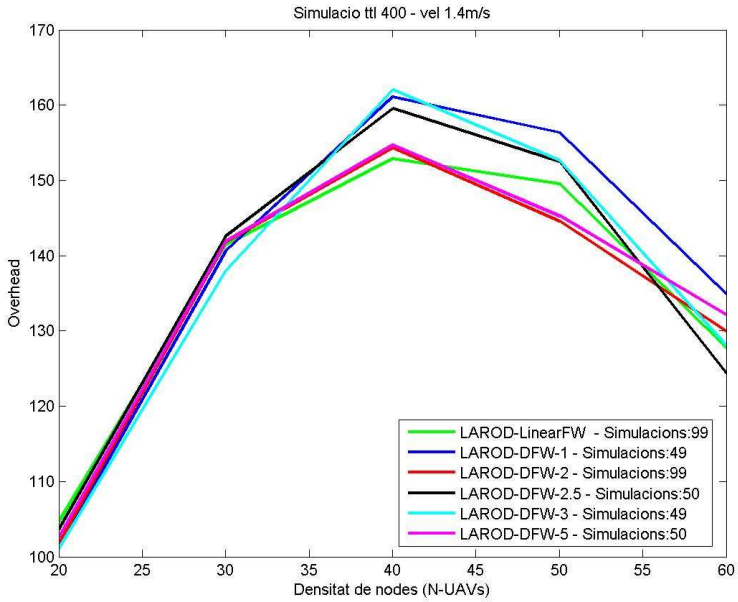

Figura A.13: Overhead - Conjunt 2: Velocitat 1.4 m/s - TTL = 400 s

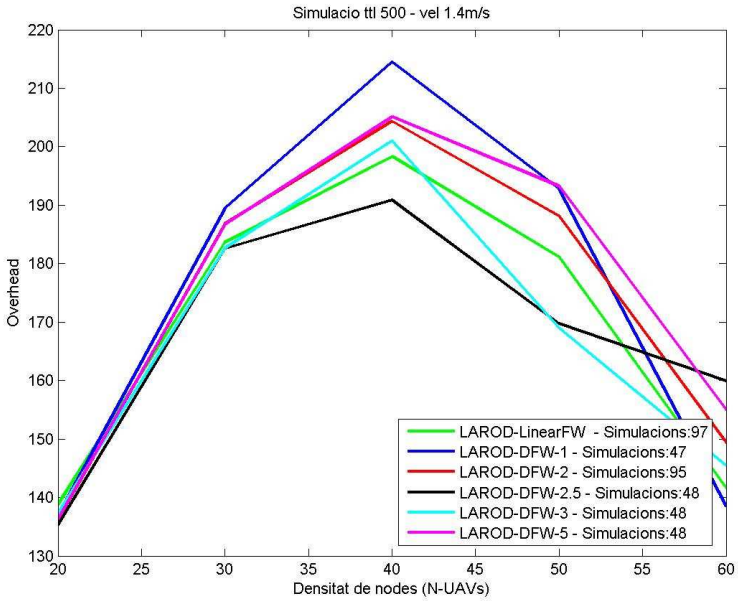

Figura A.14: Overhead - Conjunt 2: Velocitat 1.4 m/s - TTL = 500 s

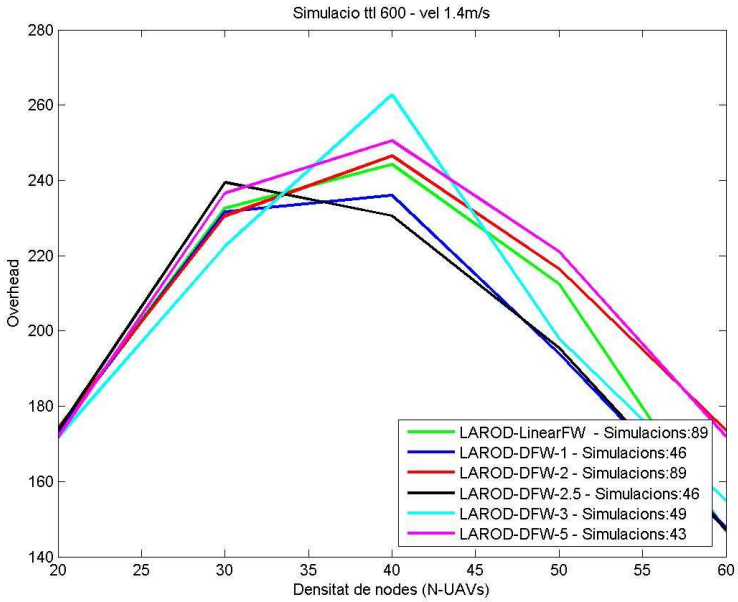

Figura A.15: Overhead - Conjunt 2: Velocitat 1.4 m/s - TTL = 600 s

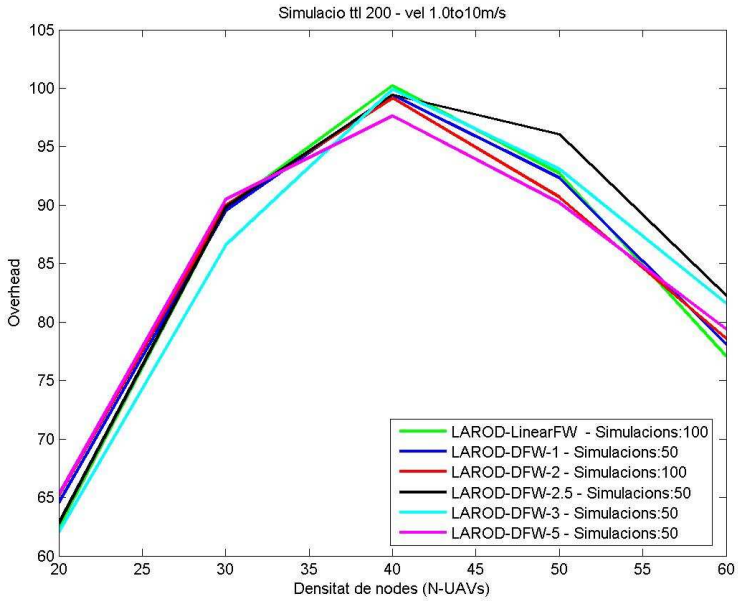

Figura A.16: Overhead - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 200 s

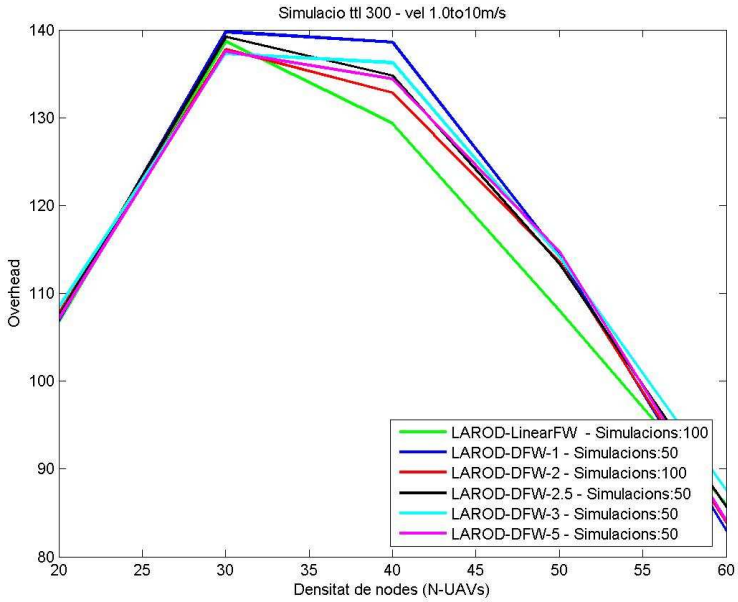

Figura A.17: Overhead - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 300 s

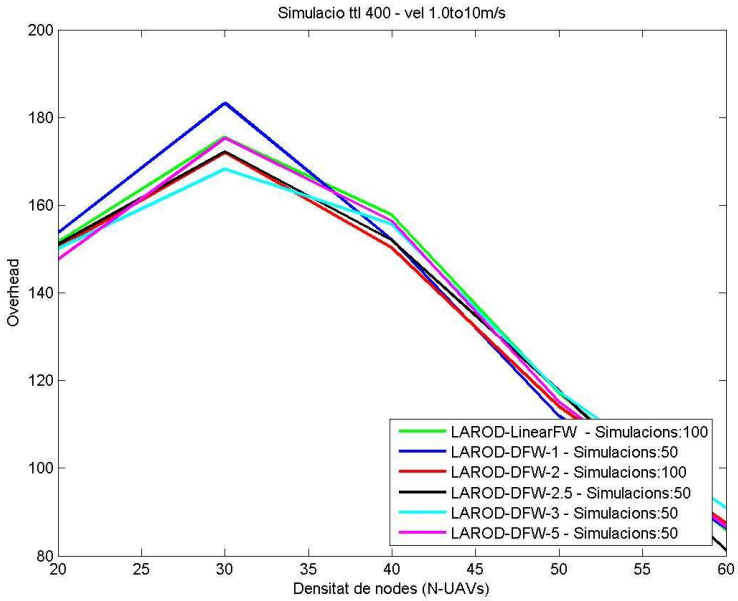

Figura A.18: Overhead - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 400 s

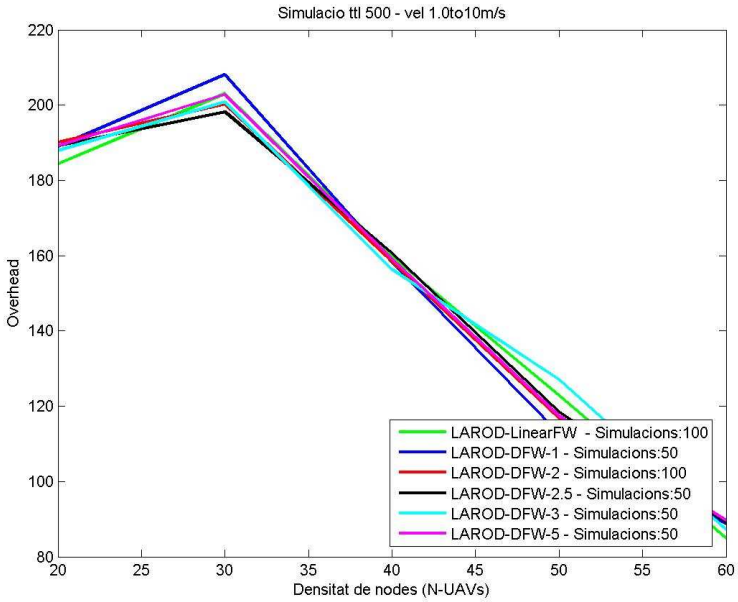

Figura A.19: Overhead - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 500 s

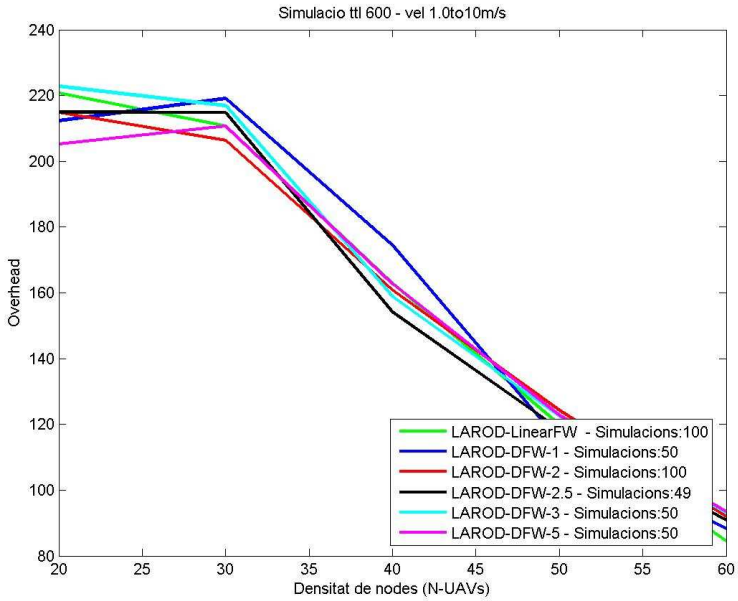

Figura A.20: Overhead - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 600 s

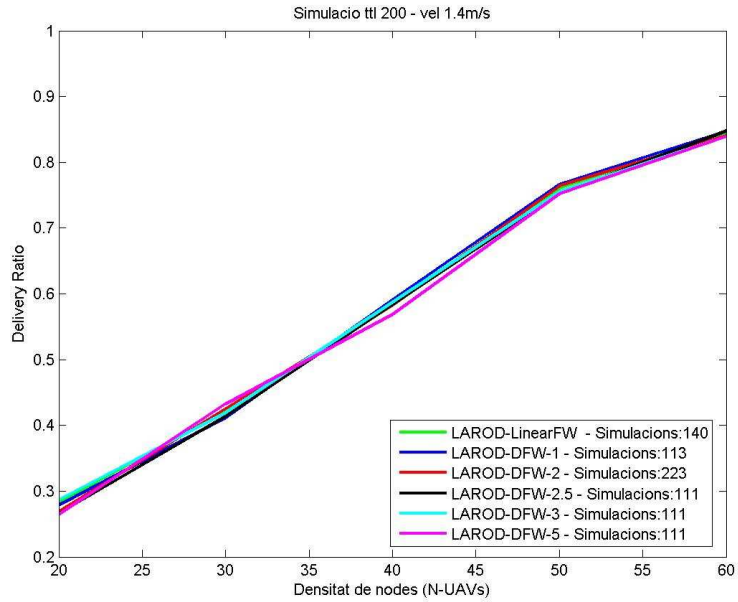

Figura A.21: Delivery Ratio - Conjunt 1: Velocitat 1.4 m/s - TTL = 200 s

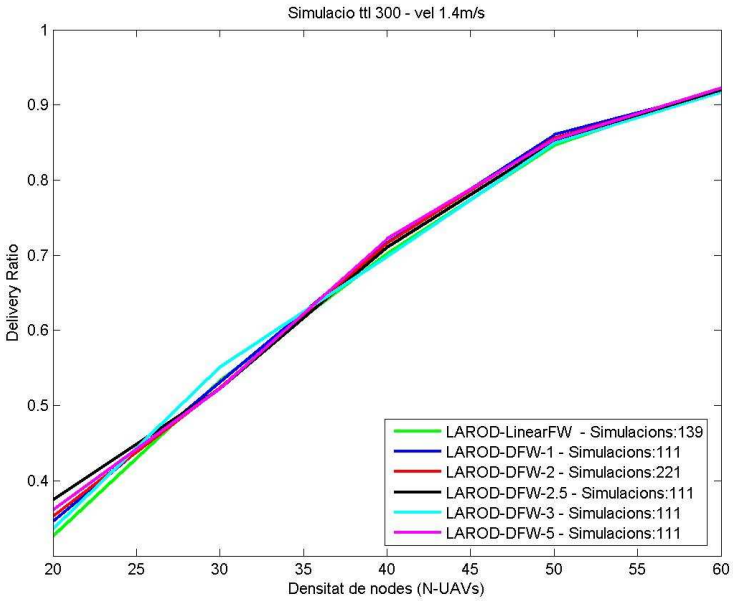

Figura A.22: Delivery Ratio - Conjunt 1: Velocitat 1.4 m/s - TTL = 300 s

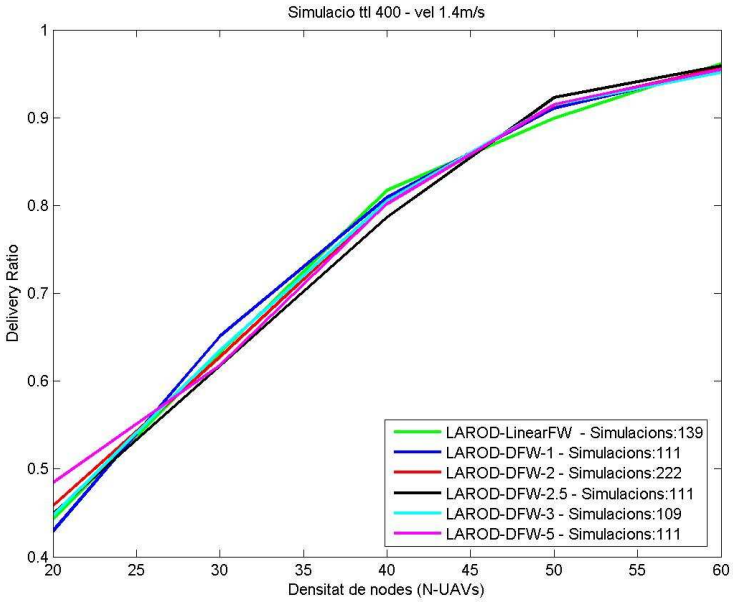

Figura A.23: Delivery Ratio - Conjunt 1: Velocitat 1.4 m/s - TTL = 400 s

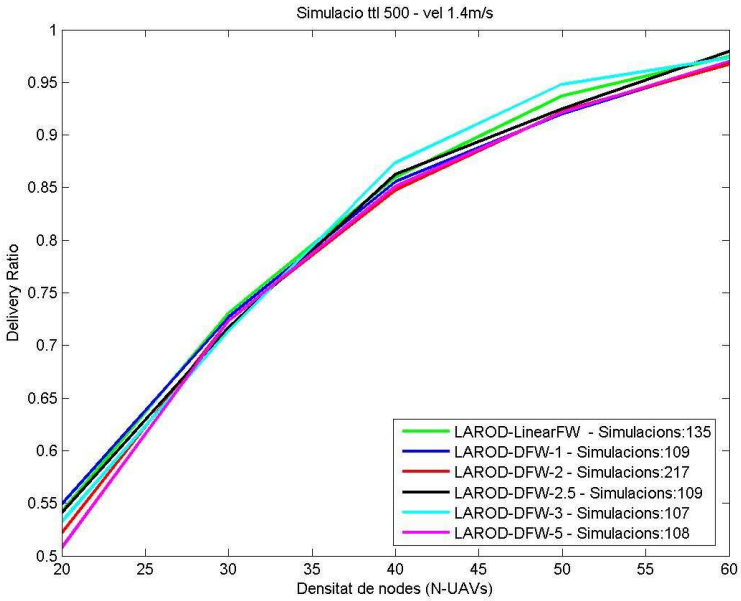

Figura A.24: Delivery Ratio - Conjunt 1: Velocitat 1.4 m/s - TTL = 500 s

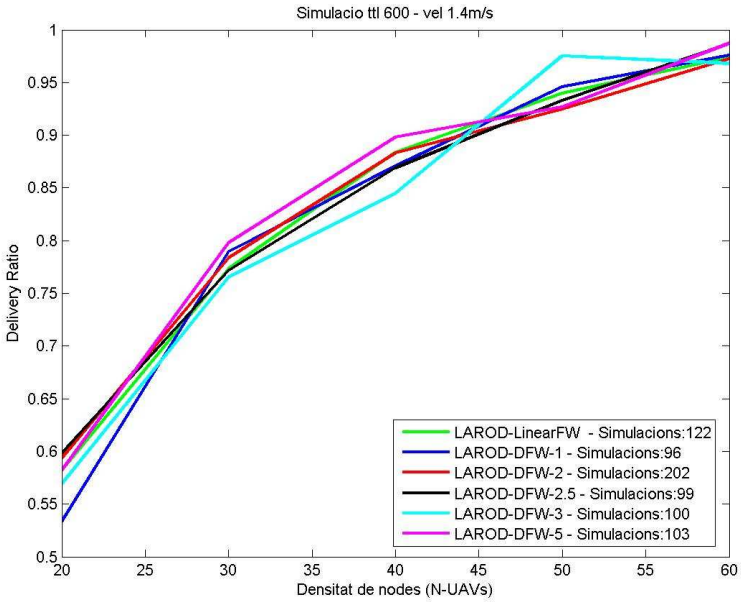

Figura A.25: Delivery Ratio - Conjunt 1: Velocitat 1.4 m/s - TTL = 600 s

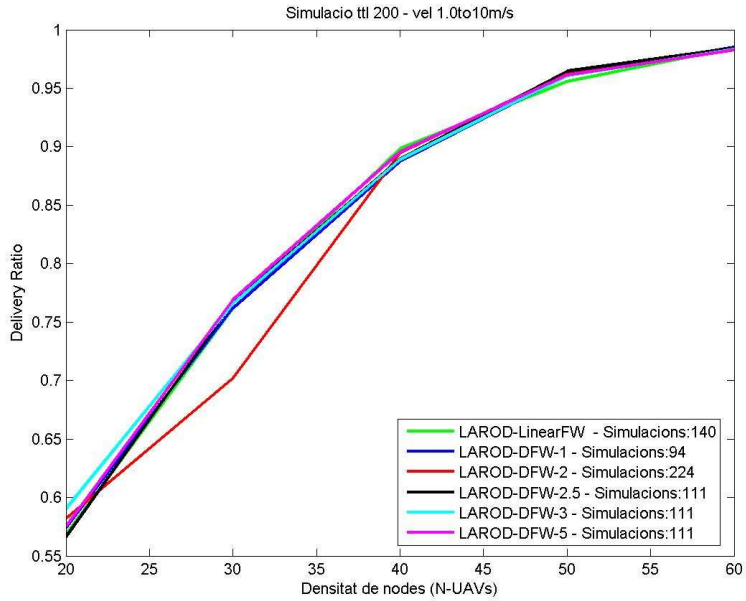

Figura A.26: Delivery Ratio - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 200 s

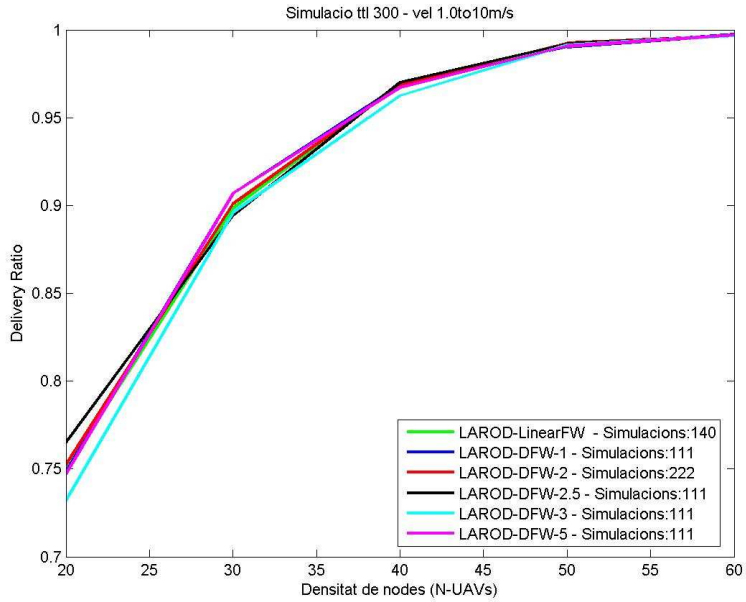

Figura A.27: Delivery Ratio - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 300 s

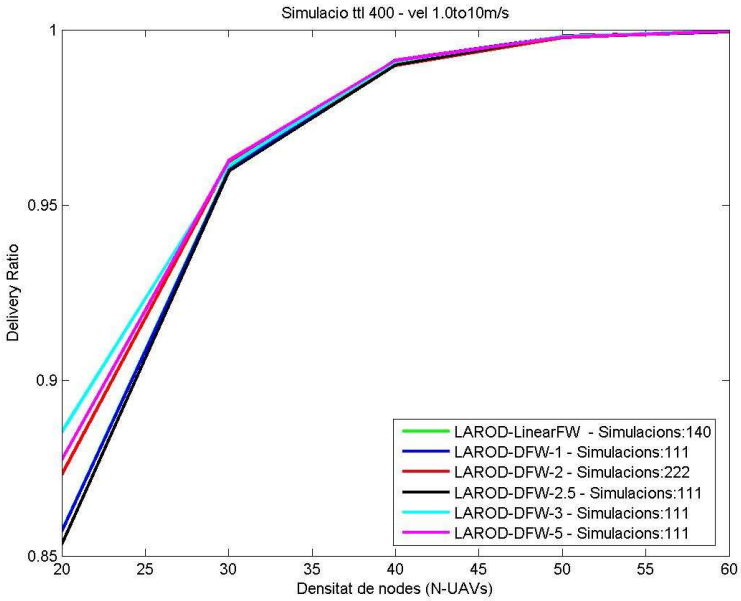

Figura A.28: Delivery Ratio - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 400 s

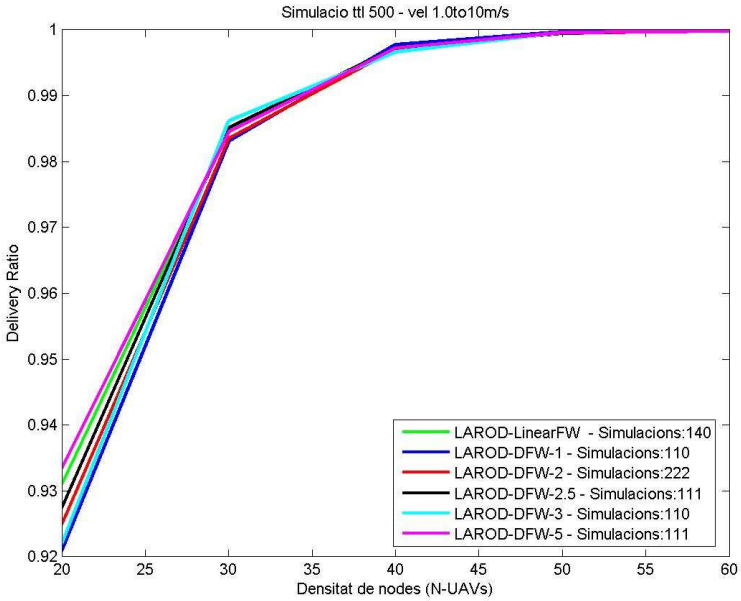

Figura A.29: Delivery Ratio - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 500 s

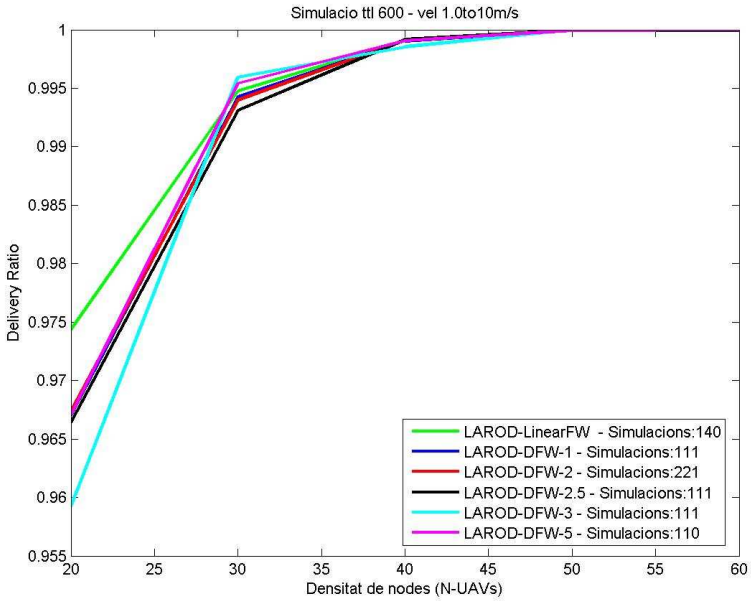

Figura A.30: Delivery Ratio - Conjunt 1: Velocitat 1.0 a 10 m/s - TTL = 600 s

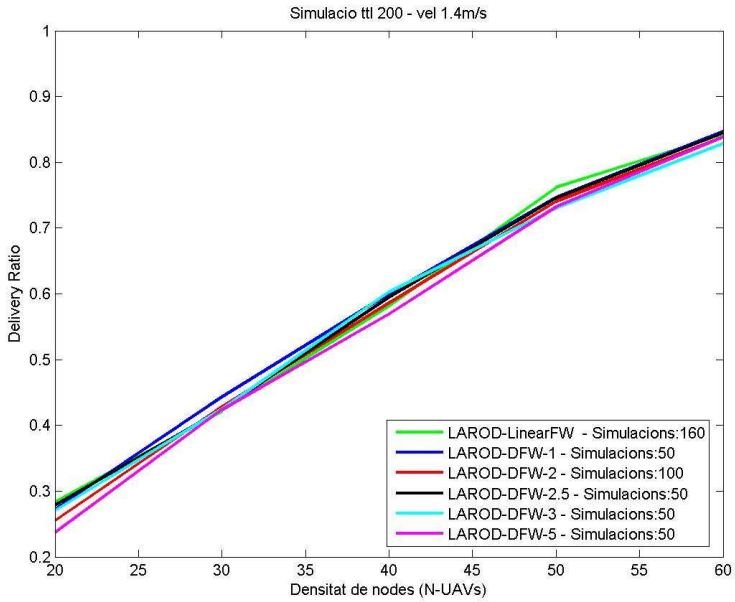

Figura A.31: Delivery Ratio - Conjunt 2: Velocitat 1.4 m/s - TTL = 200 s

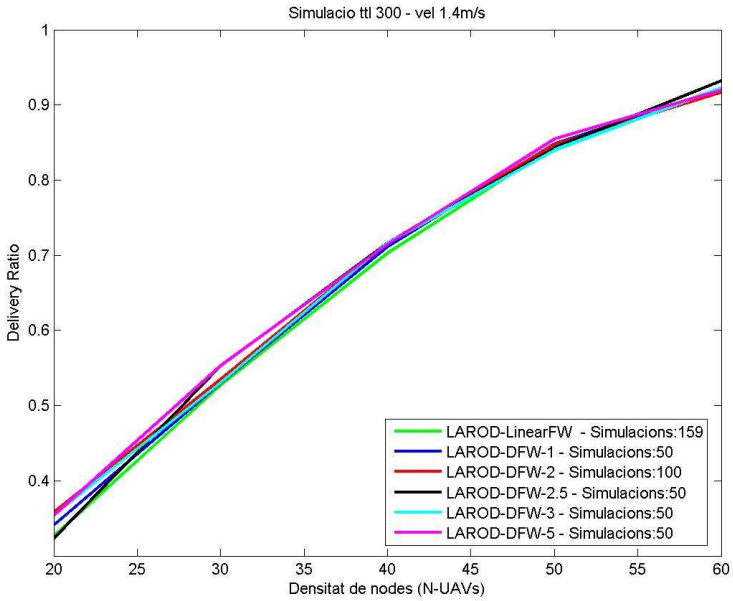

Figura A.32: Delivery Ratio - Conjunt 2: Velocitat 1.4 m/s - TTL = 300 s

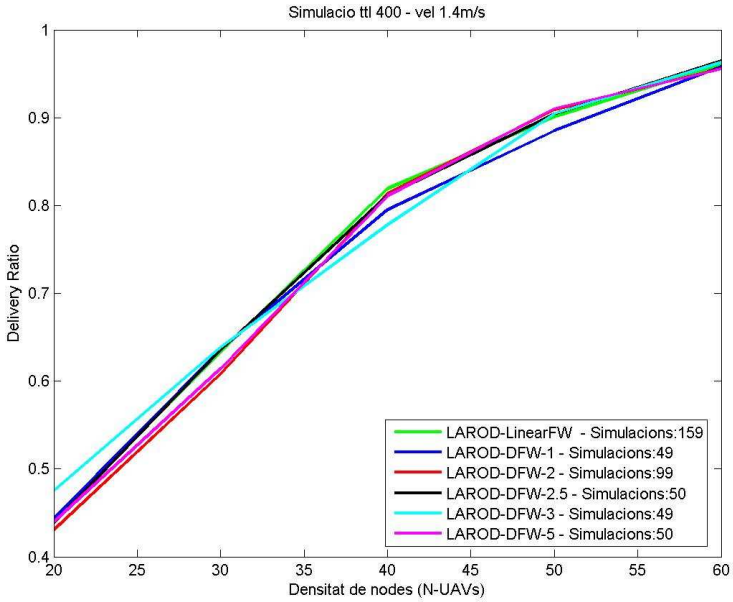

Figura A.33: Delivery Ratio - Conjunt 2: Velocitat 1.4 m/s - TTL = 400 s

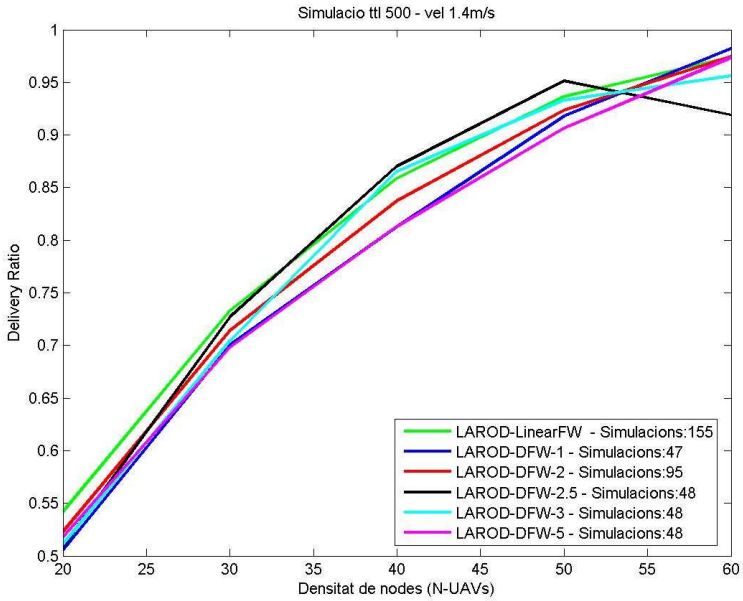

Figura A.34: Delivery Ratio - Conjunt 2: Velocitat 1.4 m/s - TTL = 500 s

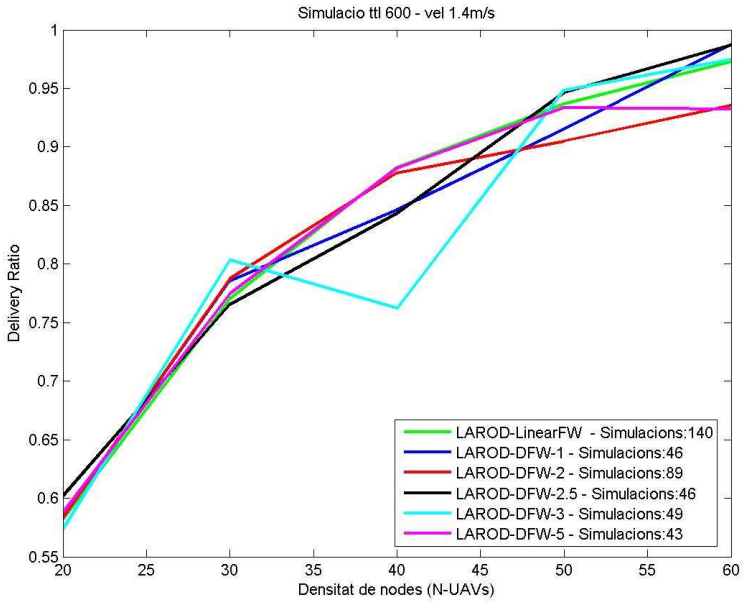

Figura A.35: Delivery Ratio - Conjunt 2: Velocitat 1.4 m/s - TTL = 600 s

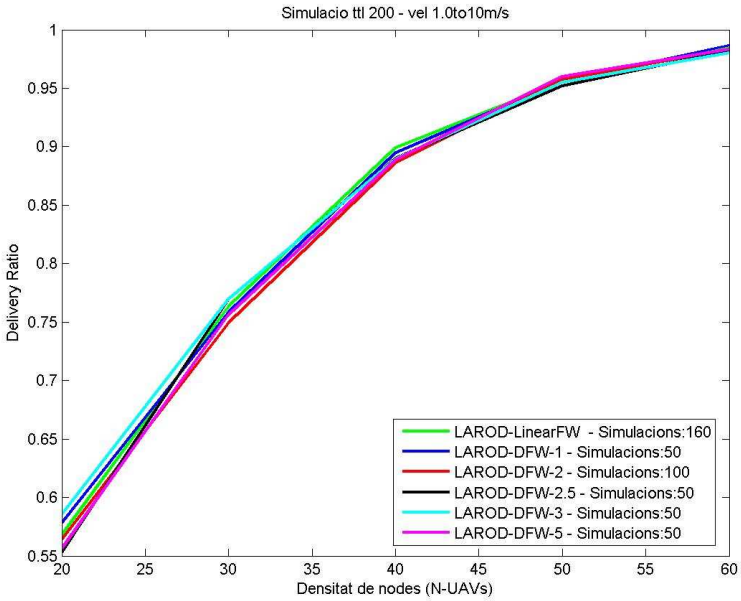

Figura A.36: Delivery Ratio - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 200 s

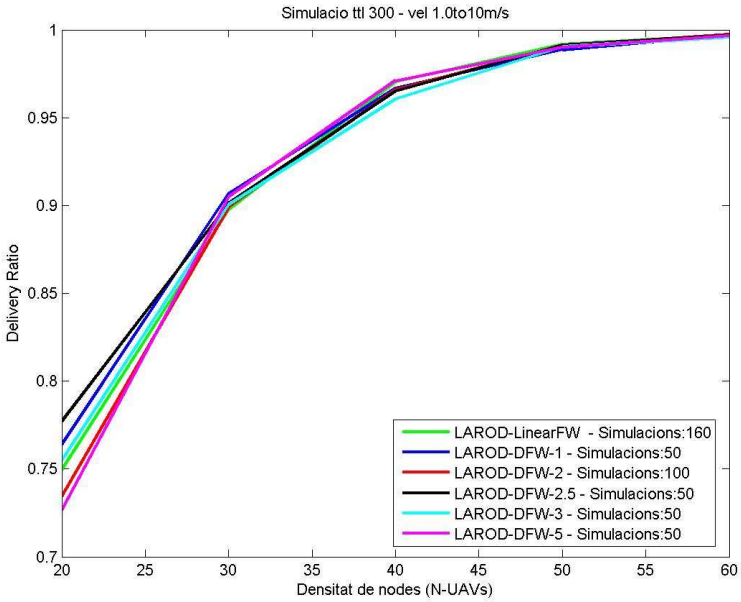

Figura A.37: Delivery Ratio - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 300 s

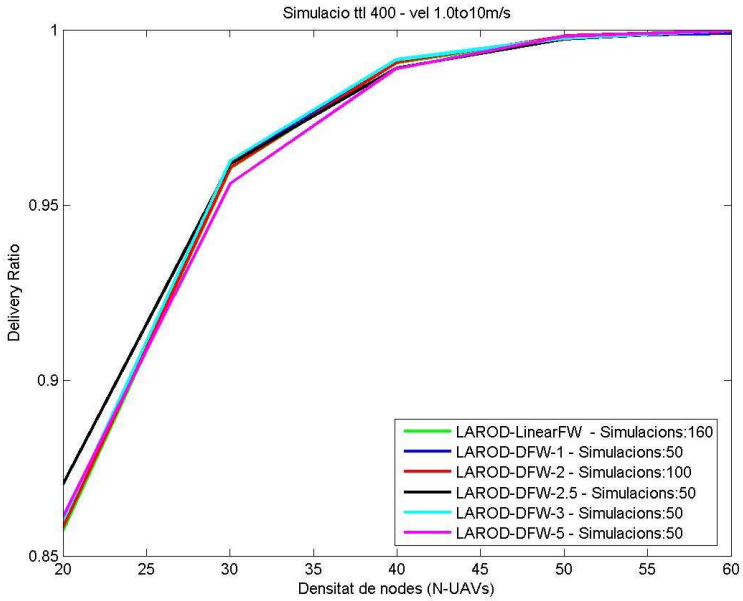

Figura A.38: Delivery Ratio - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 400 s

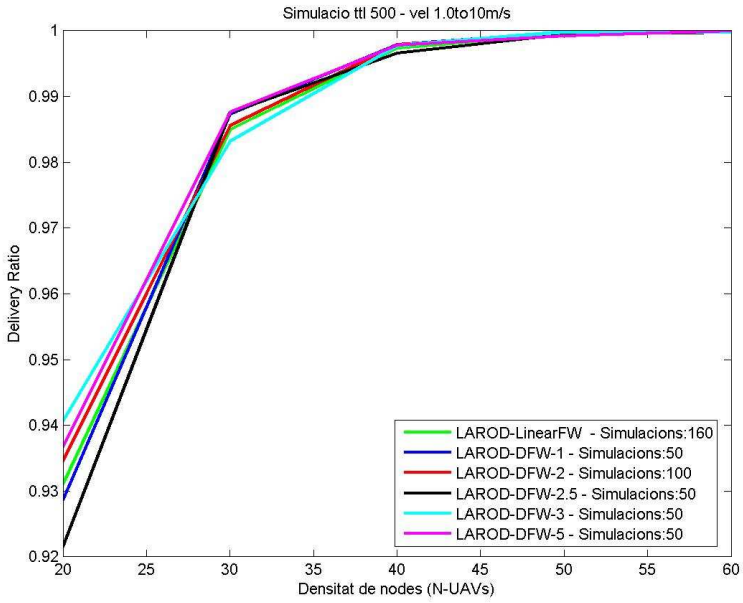

Figura A.39: Delivery Ratio - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 500 s

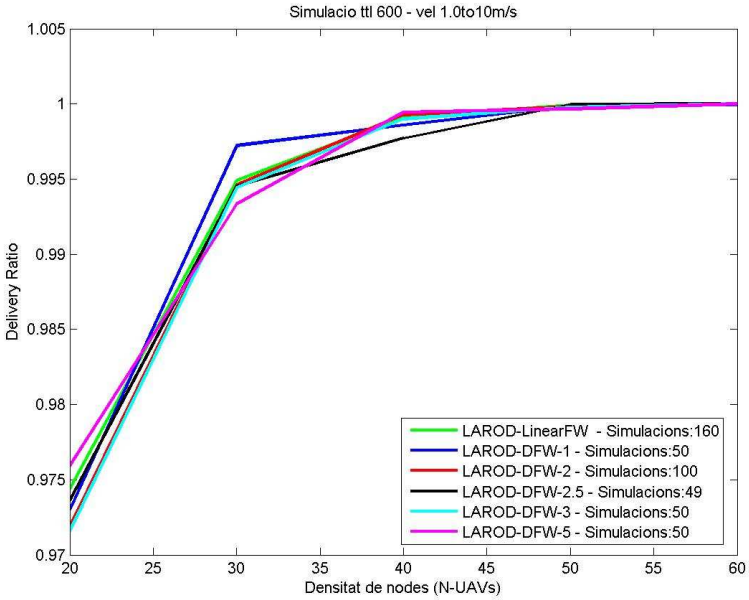

Figura A.40: Delivery Ratio - Conjunt 2: Velocitat 1.0 a 10 m/s - TTL = 600 s

Bibliografia

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

[1] M. Ramadas, S. Burleigh, S. Farrell, Licklider Transmission Protocol - Specification, September 2008, RFC 5326.

[2] S. Burleigh, K. Scott, Bundle Protocol Specification, November 2007, RFC 5050.

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, Delay-Tolerant Networking Architecture, April 2007, RFC 4838.

[4] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Networking. Proc. ACM SIGCOMM, 2004.

[5] C. Becker and G. Schiele. New Mechanisms for Routing in Ad Hoc Networks. 4th Plenary Cabernet Wksp., October 2001.

[6] Royer, E., Toh, C. "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks". IEEE Personal Communications, April 1999.

[7] A. Lindgren, A. Lindgren, E. Davies, S. Grasic, Probabilistic Routing Protocol for Intermittently Connected Networks, February 18, 2010, Internet-Draft Version 5.

[8] T. Spyropoulos, K. Psounis, C. S. Raghavendra, Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks, February 2008.

[9] Erik Kuiper, Mobility and Routing in a Delay-tolerant Network of Unmanned Aerial Vehicles, Department of Computer and Information Science Linköpings universitet SE-581 83 Linköping, Sweden, April 2008.

[10] Erik Kuiper, Simin Nadjm-Tehrani, Geographical Routing in IntermittentlyConnected Ad Hoc Networks. The First IEEE International Workshop on Opportunistic Networking. March 2008.

[11] Erik Kuiper, Details on LAROD and Spray and Wait Implementations, Version: 1.0, December 2009.

[12] Tcl/Tk for object-oriented programming. Febrer 2010 <http://otcl-tclcl.sourceforge.net/otcl/>

[13] LAROD ns-2 extentions. Novembre 2009 <http://www.ida.liu.se/ẽ-eriku/ns-2/kuiper-ns-2.30-patch-2009-11-18.zip/>

[14] The Network Simulator ns-2, Març 2010. <http://www.isi.edu/nsnam/ns/doc/>

[15] ANSI/IEEE Std 830-1984 IEEE Guide to Software Requirements Specifications, July 20, 1984.

[16] TRE/4038/2008 - 5è Conveni col·lectiu de treball del personal d'administració i serveis laboral de la Universitat Autònoma de Barcelona. 2004-2009 (codi de conveni 7902770).

Firmat: Rubén Martínez Vidal

Bellaterra, 17 de Juny de 2010

Resum

En aquest projecte es presenta un escenari de xarxa DTN. Aquest escenari té una serie de problemes que dificulten l'encaminament. Al projecte s'ha proposat un protocol d'encaminament que permet superar aquestes dificultats. Per realitzar això ha estat necessari estudiar i adquirir un coneixement profund sobre el funcionament dels protocols de xarxes DTN. Per demostrar que aquesta proposta solucionava els problemes es presenta l'anàlisi dels resultats d'aplicar el protocol a l'escenari, aquests resultats s'han obtingut amb l'eina de simulació de xarxes NS-2.

Resumen

En este proyecto se presenta un escenario de red DTN. Este escenario tiene una serie de problemas que dificultan el enrutamiento. En el proyecto se ha propuesto un protocolo de enrutamiento que permite superar estas dificultades. Para realizar esto ha sido necesario estudiar i adquirir un conocimiento profundo sobre el funcionamiento de los protocolos de redes DTN. Para demostrar que esta propuesta soluciona estos problemas, se presenta el análisis de resultados de aplicar el protocolo al escenario, estos resultados se han obtenido utilitzando la herramienta de simulación de redes NS-2.

Abstract

In this project we present a DTN network scenario with several challenges regarding routing. We propose a routing protocol that overcomes these dificulties. To achieve this we had to study and adquire an extense knowledge about DTN network protocols. To prove that our proposal solves these problems, we provide an analisys of the results of applying this protocol in the given scenario. These results have been obtained using the network simulation tool NS-2.