TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Implementació d'una xarxa P2P Pastry sobre una xarxa MANET MCL

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Xavier Basets Roca

DIRECTOR: Dolors Royo Vallés

DATA: 4 de juny de 2009

Títol: Implementació d'una xarxa P2P Pastry sobre una xarxa MANET MCL

Autor: Xavier Basets Roca

Director: Dolors Royo Vallés

Data: 4 de juny de 2009

Resum

Aquest TFC es centra en la implementació d'una xarxa P2P sobre una xarxa MANET, concretament una xarxa P2P FreePastry sobre una MANET implementada amb MCL.

S'ha analitzat el comportament de la xarxa real mitjançant la transferència de fitxers en TCP, aquest anàlisi s'ha fet sense tenir en compte la mobilitat en la xarxa.

S'han dissenyat diferents escenaris per la realització de les proves en diferents situacions, variant el numero de salts la distancia i els obstacles entre els nodes.

S'ha realitzat una aplicació que ens ha permès complimentar les opcions que ens aportava FreePastry,per a tenir una xarxa P2P mes completa, ja que les opcions que aporta aquesta xarxa són molt bàsiques.

Mitjançant aquest treball s'han extret una sèrie de conclusions, que es podrien resumir en que aquesta implementació pot ser útil en uns entorns concrets, on la proximitat dels nodes i una bona coordinació a l'hora de fer les transferències serien bàsics.

Title: Implementation of a net P2P Pastry over a MANET MCL

Author: Xavier Basets Roca

Director: Dolors Royo Vallés

Date: June, 4th 2009

Overview

This document focuses on the study of a P2P net over a MANET, precisely a net P2P FreePastry over a MANET implemented with MCL.

The behavior of the real net has been analyzed through transfering files in TCP, this analysis has made without using mobility in the net.

Different stages have been designed for the realization of the tests in different situations, variyng the number of jumps, the distance and the obstacles among the nodes.

An application has allowed to compliment the options of FreePastry, to have a net P2P complete, because the options that this sleep net brings are too much basic.

Through this work we obtained different conclusions, these can be summarized, this implementation can be useful in some concrete environments where when making the transfers they would be basic the proximity of the nodes and a good coordination.

ÍNDEX

CAPÍTOL 1. INTRODUCCIÓ

En el primer capítol s'introdueix el projecte indicant quines són les raons principals i les motivacions que porten a realitzar el projecte, a mes s'indiquen quins són els principals objectius a tenir en compte a l'hora de la realització del treball.

En el segon capítol es fa una petita explicació dels conceptes previs que són necessaris per a desenvolupar el treball.

El tercer capítol fa referència a el disseny de l'entorn on es realitzarà el projecte, indicant el software utilitzat, les raons que porten a utilitzar-lo i una breu explicació de la instal·lació i configuració.

El quart capítol ens mostra el disseny de la aplicació realitzada en Java per a poder dur a terme les proves del projecte, indicant les classes que utilitza i explicant les diferents funcions que realitza.

En el cinquè capítol es pot veure els diferents escenaris de les proves realitzades i els resultats obtinguts en cada un d'ells, a mes de un resum de resultats globals obtinguts.

En el sisè i últim capítol podem veure les conclusions extretes de la realització del projecte, on també es pot veure les possibles millores proposades.

1.1. Raó i oportunitat

Avui en dia les tecnologies Wifi evolucionen constantment, estan cada vegada més introduïdes en gran quantitat de entorns del món real, i amb elles els diferents protocols que dia a dia milloren la eficiència d'aquestes xarxes. Actualment existeixen diferents opcions per a desenvolupar la mateixa funció obtenint resultats molt diversos, per això cal fer estudis on es poden copsar les diferencies entre les configuracions. Aquests estudis mostren les diferències de rendiment entre els diferents sistemes i ens guiaran a escollir el més adequat per a cada funció.

Els dispositius mòbils són un exemple important de la constant evolució que estem vivint, a conseqüència d'això, les xarxes sense cables han de satisfer les diferents necessitats dels possibles usuaris, ja siguin particulars o empreses, i aquestes evolucionen constantment.

Les xarxes MANET són una opció interessant a estudiar, ja que aquest tipus de xarxa té grans virtuts a potenciar, com la fàcil i rapida implementació d'aquesta. Aquesta tecnologia podria ser una eina útil en alguns escenaris reduïts .

1.2. Objectius

Actualment les xarxes MANET estan guanyant protagonisme. Sobre aquest tipus de xarxes però, hi ha pocs estudis en entorns reals realitzats, la majoria d'aquests estudis es realitzen en entorns simulats. En l'escola s'han realitzat alguns treballs on s'avaluen xarxes en entorns reals utilitzant tràfic simulat.

En aquest projecte es mostra un escenari d'una xarxa P2P per a la transferència de fitxers, que utilitza la xarxa MANET, per la comunicació per avaluar les tecnologies implicades.

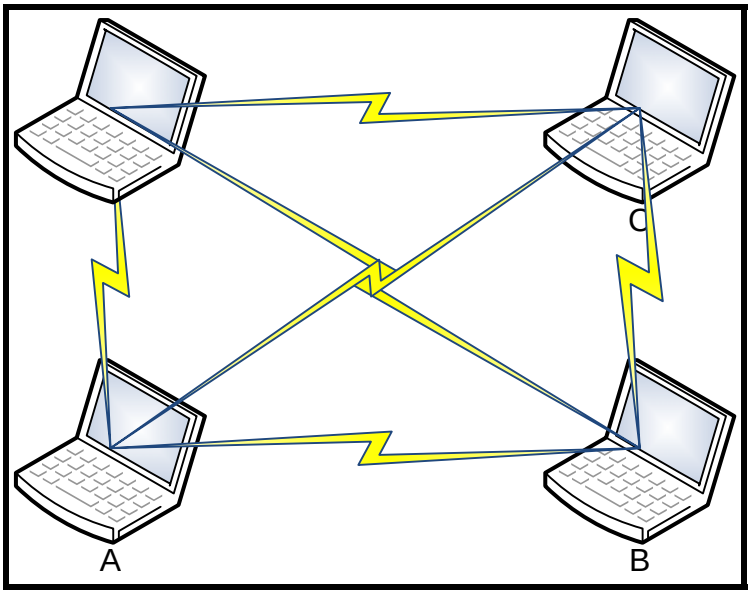


Fig. 1 Exemple Xarxa

Per a poder arribar a els objectius desitjats s'han realitzat els següents passos:

- Estudiar el funcionament del Protocol MCL ja que es el protocol escollit. per la realització del treball (instal·lació, configuració i utilització).
- Estudiar el funcionament de FreePastry i ampliar-lo.
- Programar el Servidor/Client de la gestió dels Fitxers.
- Avaluar el sistema en entorns no mòbils i veure els resultats obtinguts.

1.3. Planificació del projecte

Aquest projecte està dividit en quatre fases, una primera on es farà un estudi de les tecnologies emprades per a la realització del treball. Una segona fase on es crearà l'aplicació. Una tercera fase on es dissenyaran les proves a realitzar, i una quarta fase de anàlisi de els resultats obtinguts prèviament.

Per a la primera fase caldrà estudiar amb deteniment les diferents tecnologies emprades per a la realització del treball, es a dir xarxes MANET, tecnologia del anell FreePastry i la tecnologia MCL.

FreePastry ens proporciona una xarxa P2P basada en DHT que bàsicament permet enviar missatges entre els nodes que conformen la xarxa.
Per a fer la nova aplicació, en la segona fase, es modificaran algunes classes ja existents i s'afegiran noves funcionalitats. Aquesta aplicació permetrà publicar despublicar i buscar els diferents fitxers a mes de la opció de descarregar-los des dels diferents nodes.

La tercera fase del projecte consisteix en dissenyar les proves, per fer-ho primer es decideixen els diferents factors a analitzar, com per exemple, la quantitat de salts, la mida dels fitxers a transmetre, la distancia, entre nodes. Una vegada els factors a analitzar semblen adequats, es decideixen els diferents escenaris que s'analitzaran, es realitza el muntatge i es prenen les mesures corresponents per posteriorment en la fase 4 poder-les analitzar.

La quarta i última fase consisteix en realitzar un estudi sobre les dades obtingudes en els diferents escenaris, per a poder extraure'n conclusions i representar-les en forma de gràfiques. D'aquests resultats i del seu estudi, se'n extraurà una sèrie de valoracions sobre el rendiment de la xarxa en aquest cas concret.

CAPITOL 2. CONCEPTES PEVIS

2.1. Tipus de xarxes

2.1.1. Xarxa ad-hoc

La locució Ad-hoc prové del llatí i es pot traduir com específic. El concepte de les xarxes Ad-hoc és la seva fàcil i ràpida implementació que permet la seva utilització en casos concrets de curta i llarga durada.

La xarxa Ad-hoc és descentralitzada, autònoma i formada per un grup de nodes connectats entre ells sense cables organitzats dinàmicament. No cal una infraestructura existent.
Els nodes tenen la funció de router i de host, el control de la xarxa està distribuït entre tots ells. Si no existeix comunicació directe entre dos nodes de una xarxa, s'utilitza el multi-salt mitjançant els nodes intermitjos per arribar al destí.

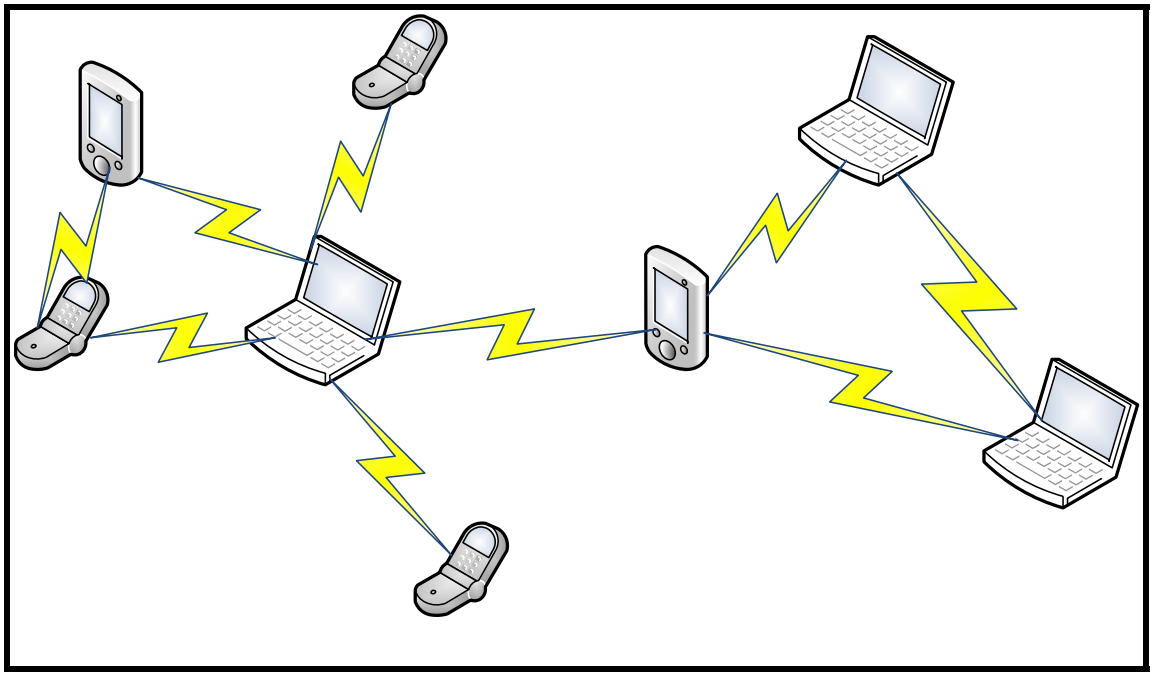


Fig. 2 Exemple Ad-hoc

En la imatge es pot veure un exemple de xarxa ad-hoc formada per un conjunt de nodes interconnectats entre ells.

2.1.2 . Xarxa Mesh

Aquest tipus de xarxa són bàsicament xarxes amb topologia en infraestructura, però que permeten unir-se a nodes que estan fora del rang de cobertura de la xarxa, però si dintre del rang de cobertura de un node que forma part de la xarxa.
Les xarxes Mesh es poden classificar en dos tipus:

Total: On tots els nodes estan connectats entre ells directament.

Parcial: On hi ha nodes que no estan dintre de la cobertura del punt d'accés però si dintre de la cobertura de algun node que si que hi es.

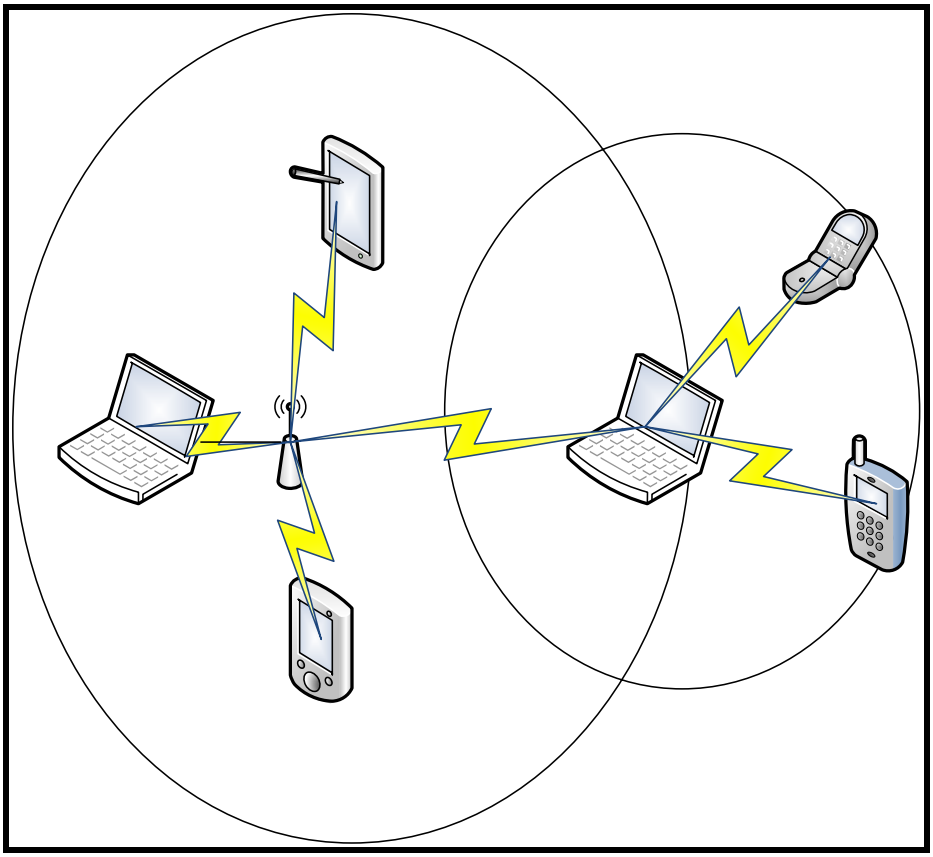


Fig. 3 Exemple Xarxa MESH

2.1.3. MANET

Una MANET es un conjunt de nodes mòbils, sense cables, que es comuniquen entre ells, aquests no tenen cap tipus de infraestructura, utilitzen enllaços multi-salt i tenen la gestió distribuïda.
La topologia d'aquest tipus de xarxes es totalment imprevisible.

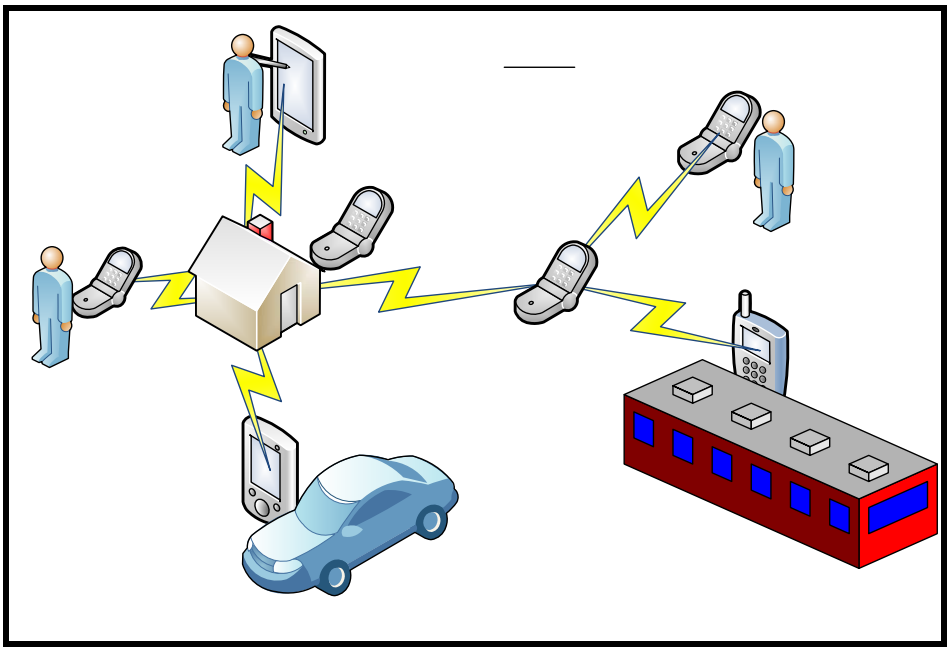


Fig. 4 Exemple MANET

Les MANET poden utilitzar diferents tipus de protocol d'encaminament:

Proactiu: Aquest tipus de protocols mantenen la informació d'encaminament actualitzada, encara que no es facin servir. Sempre que hi hagi un canvi en l'encaminament de un node de la xarxa, aquest l'ha d'informar i la resta, han de actualitzar les taules.

Aquest tipus de protocol provoca una sobrecàrrega de paquets a la xarxa avegades innecessària, però té l'avantatge de que en tot moment es pot saber la disponibilitat dels nodes i la informació d'encaminament.
Alguns exemples de protocols proactius : OLSR, BATMAN, HSR.

Reactiu: A diferencia dels Proactius els protocols Reactius nomes envien la informació d'encaminament en cas de que sigui sol·licitada, i només es guarda en els nodes que la sol·liciten per enviar un paquet. Aquesta informació nomes es guardarà temporalment, en cas de que caigui un enllaç, es recalcularà el camí.

El avantatge d'aquests protocols a diferencia de els proactius es que es molt menys important la càrrega de paquets en la xarxa, però augmenta el retard a l'hora de calcular les rutes.
Alguns exemples de protocols reactius: AODV, DSR, LQSR.

En aquest treball s'ha escollit un protocol reactiu per a la realització de les proves. S'utilitzarà el driver MCL que funciona sobre el protocol LQSR.

Híbrid: Es una combinació entre els dos tipus anteriors, s'intenta adaptar les avantatges dels dos protocols, i disminuir les desavantatges. Alguns camins estaran sempre informades i d'altres es descobriran en el moment de utilitzar-les.
Alguns exemples de protocols híbrids: BABEL, ZHLS, HSLS.

2.2. P2P

Coneguda també com a xarxa punt a punt, es una xarxa en que cada un dels components de la mateixa es un node que pot ser tan servidor com client, inclús servidor i client simultàniament.

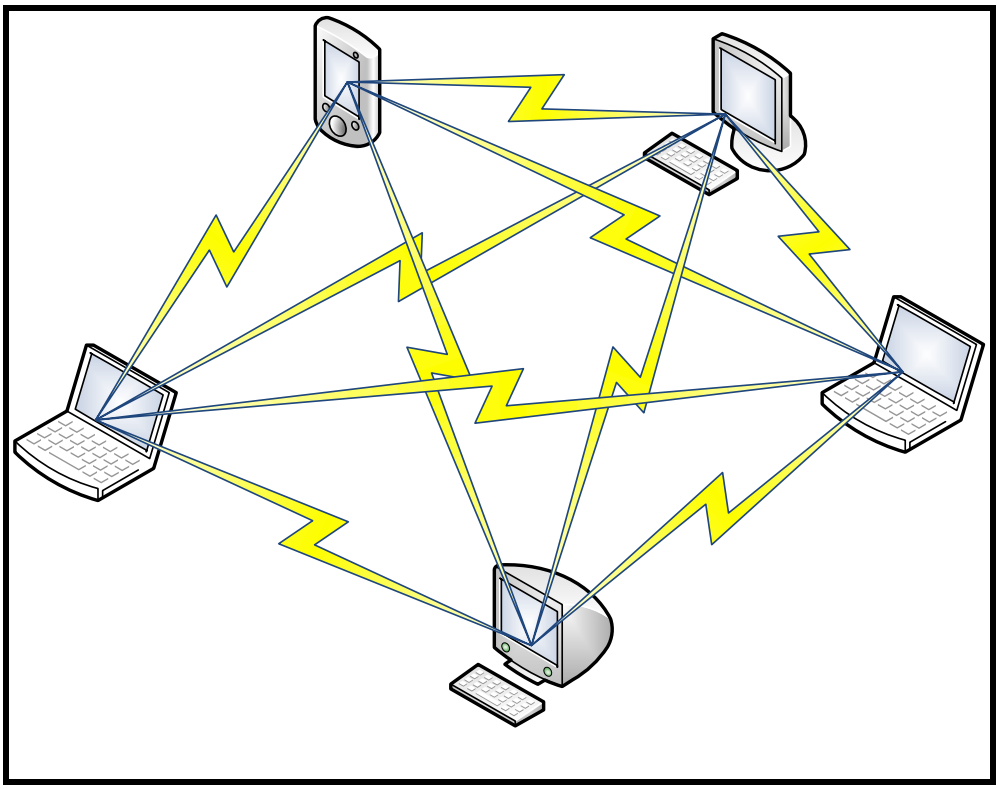


Fig. 5 Exemple P2P

Característiques que hauria de tenir una xarxa P2P:

El ús dels recursos es compartit entre els diferents nodes.
Al ser una xarxa descentralitzada, no es té dependència de un node concret.
La principal característica de una xarxa P2P es que es totalment escalable, es poden introduir nous nodes per a poder incrementar la xarxa, incrementant així també els recursos disponibles.
La pitjor característica de una xarxa P2P es que es difícil implementar-hi un alt nivell de seguretat, a causa de la estructura de la xarxa.

Tipus de xarxes P2P:

La classificació mes utilitzada es la per centralització, hi ha tres tipus:

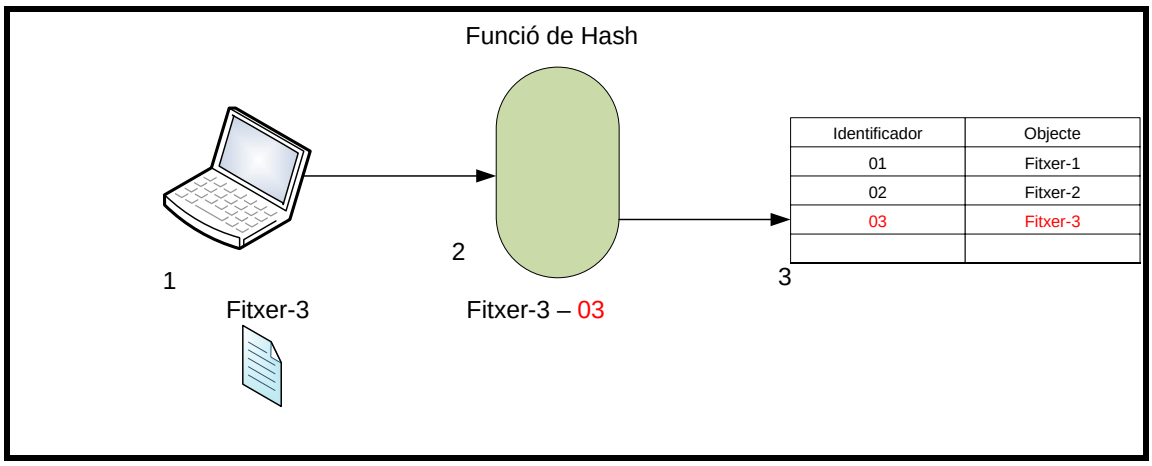
Xarxes centralitzades: en aquest cas existeix un únic servidor que s'encarrega de gestionar tot el contingut al qual es pot accedir, es a

dir, per a fer qualsevol petició d'accés a contingut, depèn totalment del servidor.
Xarxes descentralitzades: aquest tipus de xarxa es la mes habitual, no s'utilitza un servidor, totes les comunicacions es fan node a node directament. Per tant cada node pot ser client i servidor.
Xarxes semi-centralitzades: existeix un servidor central, però actua com a repetidor, no emmagatzema informació, però s'encarrega de gestionar el ample de banda , i els enrutaments.

2.3. DHT

Una taula de Hash es una forma organitzativa. En els sistemes que la utilitzen els usuaris són capaços de assignar un identificador a un objecte. El identificador es calcula a partir de una sèrie càlculs matemàtics referents a diferents característiques del objecte.

Funcionament:
Inserir:

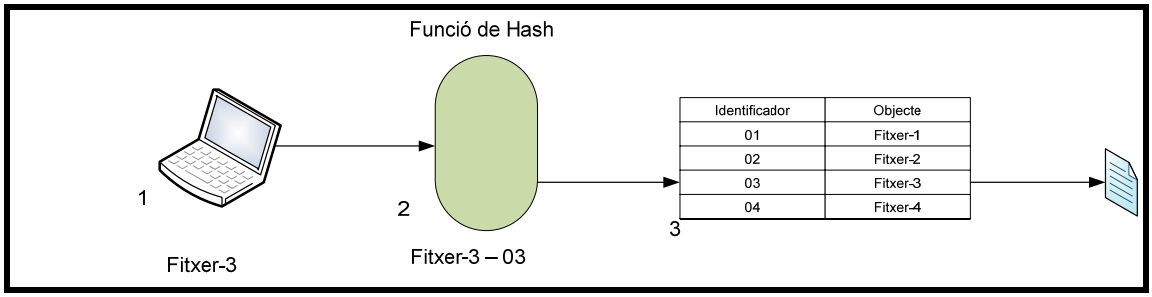


1: El usuari vol inserir un fitxer a la Taula de Hash amb el nom Fitxer-3.

2: Per poder introduir aquest nou element a la taula, primer s'haurà d'obtenir un identificador, per a fer-ho s'utilitzarà la Funció de Hash. La Funció de Hash proporcionarà un identificador al fitxer que estem inserint, mitjançant una sèrie de càlculs matemàtics.

3: S'introdueix el identificador assignat en la taula de Hash. Si el identificador existís provocaria una col·lisió dintre de la taula.

Buscar:



1: El usuari vol buscar un fitxer. Per buscar en una taula de Hash cal saber únicament el nom.

2: La funció de Hash ens proporcionarà el identificador a partir del nom, en aquest cas ens indica que el identificador que esta assignat es el 03.

3: Es buscà el identificador dintre de la taula i se'n obté el objecte assignat.

Les taules de Hash distribuïdes s'utilitzen com a opció quan hi ha un nombre gran de nodes, la finalitat es descentralitzar el sistema i repartir el contingut de la taula entre els diferents nodes. Principalment s'utilitzen per a sistemes on es comparteixen fitxers, són un element essencial en les xarxes P2P i tenen un efecte revolucionari en la descentralització de les xarxes.

Les DHT es basen en 3 característiques importants .La principal es que funciona de forma descentralitzada, es a dir la informació està compartida per els diferents nodes que componen la xarxa, no cal que hi hagi un servidor únic. Una altre característica important es que el sistema de funciona correctament quan hi ha entrada o sortida de nodes en la xarxa, podríem dir que el rendiment no es veu afectat en aquest cas. Finalment també em de tenir en compte que té una gran escalabilitat, es a dir el sistema funciona correctament amb pocs i molts nodes independentment, en aquest cas, es una bona opció per a una xarxa que té tendència a grans canvis de estructura.

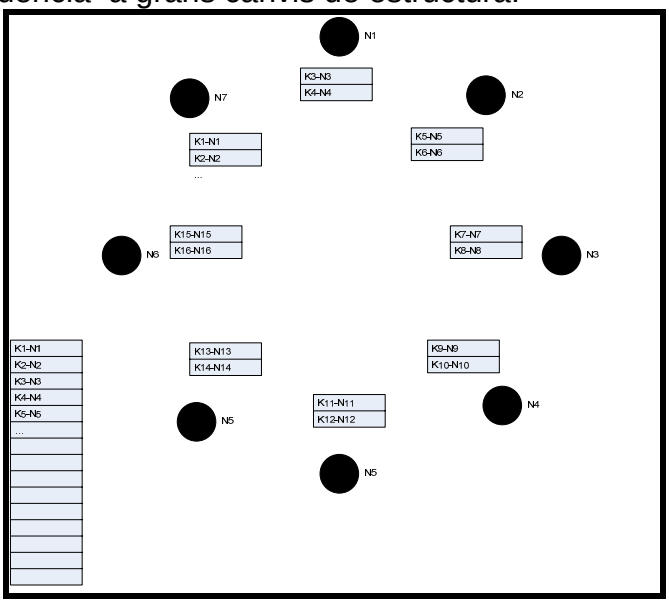


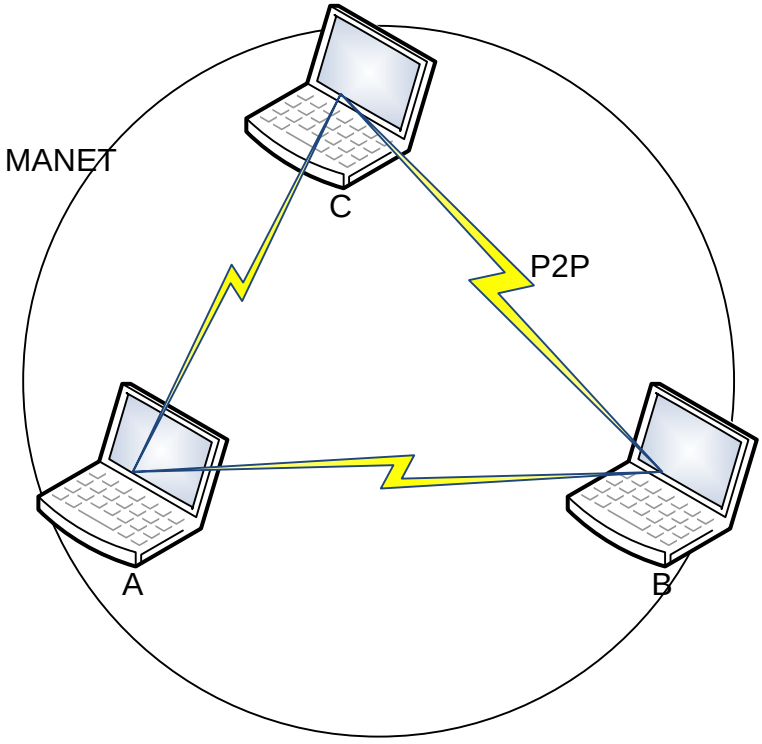
Fig. 6 Funcionament DHT

CAPITOL 3. ENTORN

3.1. Software emprat.

Per dissenyar el sistema que es vol estudiar cal escollir :

- La MANET.
- La xarxa P2P.



- MANET : MCL

En el cas de la MANET, tal com indiquen els requisits del projecte, s'utilitza MCL. Principalment, compleix la característica de utilitzar un protocol ad-hoc, a mes a mes és fàcil de implementar i que introdueix molt poc soroll a la xarxa, i tot i ser un protocol reactiu, no ens afectarà, ja que en les proves realitzades s'han exclòs les proves en moviment.

MCL es un software lliure amb versions per linux i Windows, Aquest protocol s'introdueix en la xarxa entre la capa 2(MAC) i la capa 3(xarxa) de la OSI, es podria dir que estem parlant de una capa intermèdia (2,5). Aquest driver es pot configurar per que funcioni en una xarxa Mesh.

MCL aporta avantatges importants, ja que un cop instal·lat les capes mes altes, funcionen com si estiguessin sobre una xarxa ad-hoc. Per tant no caldrà fer modificacions en la pila de protocols de la xarxa, ja que les funcions com ARP o

DHCP funcionen correctament. Per un altre banda el adaptador virtual MCL pot multiplexar varis adaptadors de xarxa, i no té problemes per a conviure amb altres protocols de xarxa ad-hoc.

- P2P : FreePastry-2.1 alpha 3

S'ha decidit utilitzar FreePastry per varies raons, FreePastry ens aporta una xarxa P2P amb topologia lògica en anell. Per altre banda FreePastry ens introdueix molt soroll a la xarxa causat per els paquets de reconeixement que s'intercanvien els nodes contínuament, això resulta interessant per veure la reacció a les interferències durant la transferència. També cal tenir present que es una implementació de baix nivell i que caldrà modificar i adaptar les classes Java que el formen per poder realitzar les funcions necessàries per el estudi.

FreePastry és una implementació Pastry en Java que compleix les característiques principals de una xarxa P2P, aquest sistema funciona utilitzant una estructura basada en una DHT(Distributed Hash Table).

Es situa per sobre de la capa de transport, defineix una estructura lògica en forma de anell, on els nodes es connecten entre si, la entrada de un nou node en la xarxa, provoca la restauració la resta dels nodes de forma que queden totalment reorganitzats, i s'anuncia als nodes mes pròxims. En cas de eliminar un node de la xarxa els nodes mes propers a l'eliminat tornaran a quedar directament connectats. FreePastry utilitza un conjunt de protocols propis que s'encarreguen de mantenir la xarxa activa.

Tot i ser una xarxa descentralitzada, existeix un node anomenat Bootstrap Node que té la funció de iniciar la xarxa, aquest node es el punt de partida, a on es dirigiran els nodes que vulguin entrar al nou anell. Per entrar en un anell FreePastry caldrà conèixer el Bootstrap node ja que s'haurà de dirigir a ell la petició de entrada, per a obtenir un identificador. Quan el nou node ja està integrat a la xarxa es crearà la taula de rutes per arribar a tots els nodes de la xarxa.

Al generar un node en FreePastry es crea un identificador aleatori hexadecimal que s'encarregarà d'identificar el node dintre de la xarxa on s'ha inclòs.
Quan el node es vol introduir a la xarxa se li assigna un identificador format per 40 caràcters hexadecimals. Aquest identificador servirà per a poder trobar el lloc en la xarxa, ja que es situarà entre el immediatament major i el immediatament menor que hi hagin dintre de la xarxa.

Cada node té un identificador(node-Id) de 128 bits. En una xarxa Pastry de N nodes cadascú pot enviar a un altre node en com a molt $[^{\log}2^b N]$ salts , b es un valor configurat que acostuma a ser 4. Cadascun dels nodes té una taula de Hash que conte $(2^b-1)*[^{\log}2^b N]$ entrades, aquesta taula relaciona cadascun dels identificadors de node amb la ip de la maquina corresponent. Aquestes estan organitzades en $[^{\log}2^b N]$ files de 2^b-1 entrades.

Quan són nodes adjacents en una xarxa la comunicació entre ells es directe. Però quan hi ha 2 o mes salts FreePastry utilitza un algoritme que es basa en la utilització de els identificadors de node.

El que fa FreePastry es transmetre al node adjacent mes proper numèricament, així també actuaran els nodes següents fins arribar al node destinatari.
Aquest mètode permet a la xarxa assegurar-se de que s'arribarà al node final en menys de Log N salts. En la imatge es pot veure el comportament de FreePastry en una transferència del node 09999 al node 12345

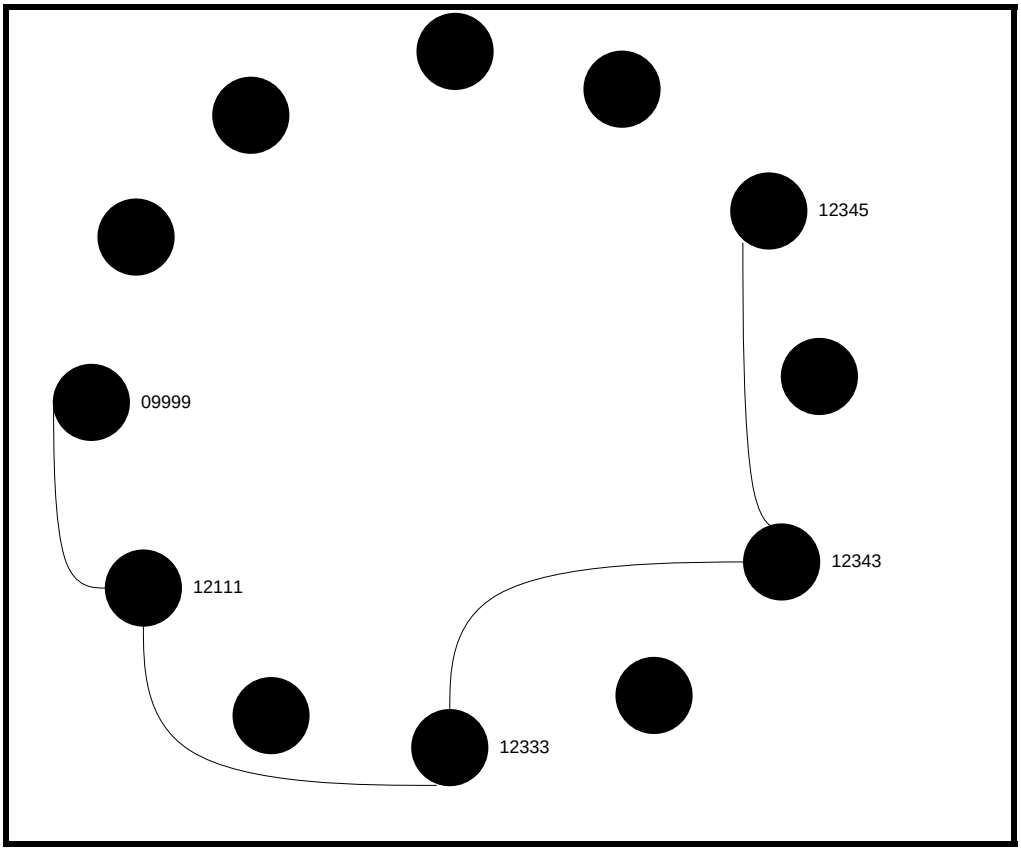


Fig. 7 Exemple búsqueda de nodes en FreePastry

3.2. Instal·lació entorn

3.2.1.- Instal·lació /Configuració FreePastry

FreePastry està fet amb Java, per a poder utilitzar FreePastry Caldrà tenir instal·lada en la maquina una versió actual del JDK, amb la versió 1.5.0_08 o superior n'hi ha suficient.
Aquest el podem descarregar de la pagina http://java.sun.com/javase/index.jsp
.
Quan està instal·lat el JDK tenim dues opcions, o descarregar directament el arxiu ja compilat i preparat per a ser executat, FreePastry-2.1alpha3 , o baixar les classes i compilar-les per poder-les executar després.

Per a la realització del projecte es mes útil descarregar les classes, ja que algunes es tindran de consultar per a poder utilitzar-les correctament, demés algunes classes es modificaran i se'n afegiran de noves per a la aplicació que s'utilitzarà per a fer el estudi.

Per executar-lo haurem de passar 3 paràmetres, el primer es el port que esta utilitzant el node que creem, el segon serà la IP del node bootstraap(principal) i el tercer es el port que utilitza el bootstraap per a la comunicació, aquí es veu un exemple de com creem un nou anell amb 2 nodes.

Llançament node principal:

```
C:\>java -cp FreePastry-2.1alpha3.jar rice.tutorial.lesson1.DistTutorial 9001 19
2.168.1.9 9001
Finished creating new node TLPastryNode[SNH: <0xADE4D6..>/desktop/192.168.1.9:90
01]
```

Llançament segon node:

```
C:\>java -cp FreePastry-2.1alpha3.jar rice.tutorial.lesson1.DistTutorial 9002 19
2.168.1.9 9001
Finished creating new node TLPastryNode[SNH: <0xF41D0C..>/desktop/192.168.1.9:90
02]
```

3.2.2 .- Instal·lació /configuració MCL

La instal·lació de MCL es una instal·lació relativament complexa, ja que cal configurar parametres de la xarxa, i de la configuració wireless. S'ha fet seguint les instruccions annexades en la tesi : *A real-world implementation and parametrization of mobile ad-hoc networks.,* on indica clarament com configurar tots els parametres per que funcioni correctament.

CAPITOL 4. DISSENY APLICACIÓ

Degut a que FreePastry es una xarxa programada amb unes opcions molt bàsiques, ha calgut modificar les funcions originals, i afegir-ne de noves.
La aplicació realitzada en Java esta dissenyada concretament per a poder realitzar el estudi de la publicació i transferència de fitxers utilitzant FreePastry sobre MCL. Aquesta aplicació ens permetrà fer les següents funcions bàsiques:
- Publicar.
- Despublicar.
- Buscar.

A mes a mes afegirem la funció de descarregar a la aplicació.

Totes aquestes opcions estaran disponibles des de qualsevol node de l'anell FreePastry, i cadascun dels nodes podrà publicar els fitxers, despublicar-ne, buscar la ubicació de un fitxer o inclòs descarregar-se'n una copia del original.

Per executar la aplicació utilitzarem la següent comanda :

Java –cp FreePastry-2.alpha3.jar rice.tutorial.lesson3.DistTutorial (port del node) (ip del node principal) (port del node principal)

Exemple:

```
C:\>java -cp FreePastry-2.1alpha3.jar rice.tutorial.lesson1.DistTutorial 9002 19
2.168.1.9 9001
```

Seguidament obtindrem el menú de la aplicació :

```
.-MENU-.
  1) Publicar.
  2) Despublicar.
  3) Buscar.
  4) Descarregar.
  0) Sortir.
  Opcio:
```

4.1. Funcions

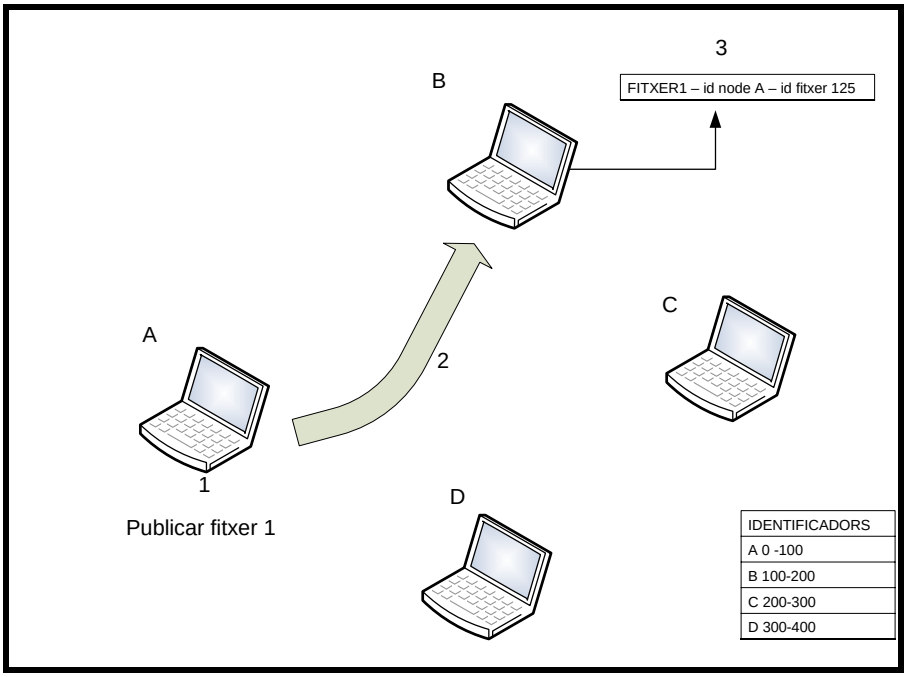
4.1.1.- Publicar.

La funció publicar permetrà ,als usuaris de la aplicació, la opció de publicar un fitxer en la Xarxa MCL, a traves de FreePastry. Al fitxer se li assignarà un identificador relacionat amb el seu nom, que serà únic e irrepetible. Depenent del Identificador assignat s'enviarà a un o altre node per a que sigui gestionat.
Aquest el rebrà el node que gestioni el rang de identificadors on es trobi inclòs, quan el node el rebi l'introduirà en una taula on estarà relacionat el nom del fitxer, el Id del node on esta publicat el fitxer i el identificador assignat al fitxer.

Mai existiran 2 fitxers amb un mateix identificador assignat, ja que el identificador va en funció exclusivament del nom i en cas de existir 2 identificadors iguals existirien 2 fitxers amb el mateix nom.

Una vegada el Fitxer estigui publicat a la xarxa aquest podrà ser descarregat, buscat o despublicat des de els diferents nodes que hi hagi en aquesta.

Funcionament:



1: El node A publica un fitxer que emmagatzema en el seu disc dur. El fitxer indicat rebrà un identificador mitjançant una funció de FreePastry, que proporciona un identificador únic a partir del nom. A mode de exemple direm que el identificador assignat en aquest cas es el 125.

2: El segon pas es enviar el nom del fitxer i el identificador del node que el guarda, que en aquest cas serà el node A. Aquesta informació s'ha de enviar al node que gestiona el identificador assignat al fitxer. En aquest cas es el node B que tal i com es veu en la taula gestiona els identificadors que van des del 100 al 200.
La informació s'enviarà dintre de un paquet tipus "publicar".

3: L'últim pas és introduir les dades en la taula, d'això se'n encarrega el node que gestiona la petició de publicar, en aquest cas el node B. Aquest guardarà un registre d'aquest tipus en la taula.

Nomfitxer	Identificador node	Identificador fitxer
FITXER 1	A	125

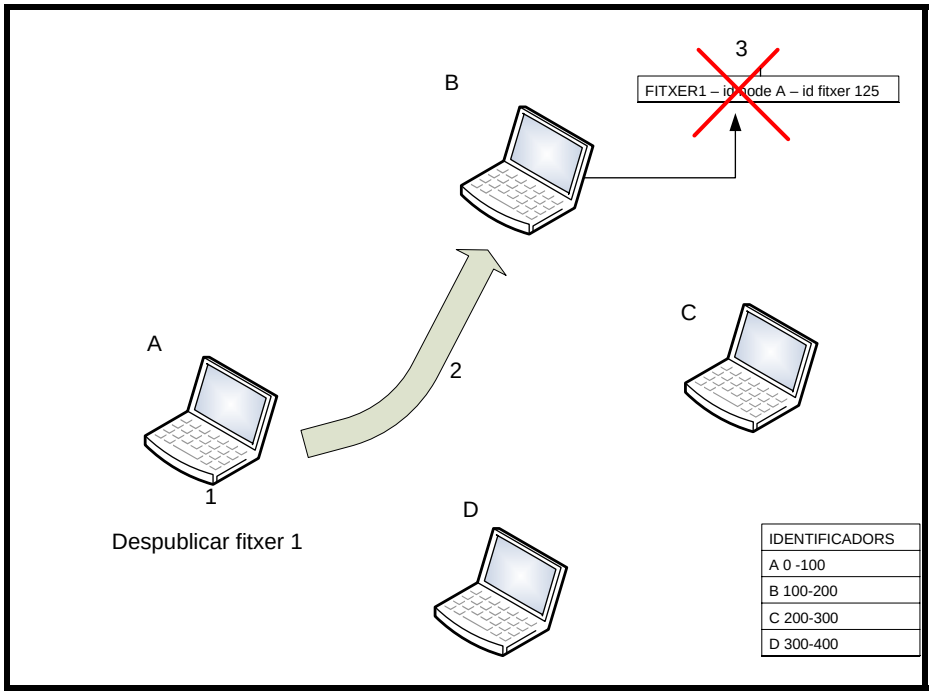
4.1.2.- Despublicar.

La funció despublicar permetrà, com el seu nom indica, extreure el fitxer de la llista de fitxers disponibles per a descarregar. Per a poder despublicar un fitxer nomes haurem de indicar el nom, d'aquest nom de fitxer n'extraurem el identificador, i amb ell sabrem el node que esta gestionant el fitxer i per tant el conte en la seva llista.

El node rebrà un missatge amb el nom del fitxer i el Tipus despublicar indicat, quan el rebem, el buscarem en la llista i l'eliminarem reestructurant altre vegada la resta del array.

Quan un fitxer estigui despublicat ja no es podrà buscar ni descarregar.

Funcionament:



1: El node A despublica que ha estat publicat previament. Per fer-ho s'haurà d'indicar el nom del fitxer, aquest, mitjançant una funció de FreePastry ens retornarà el seu identificador. En aquest cas el identificador es el 125 gestionat per el node B.

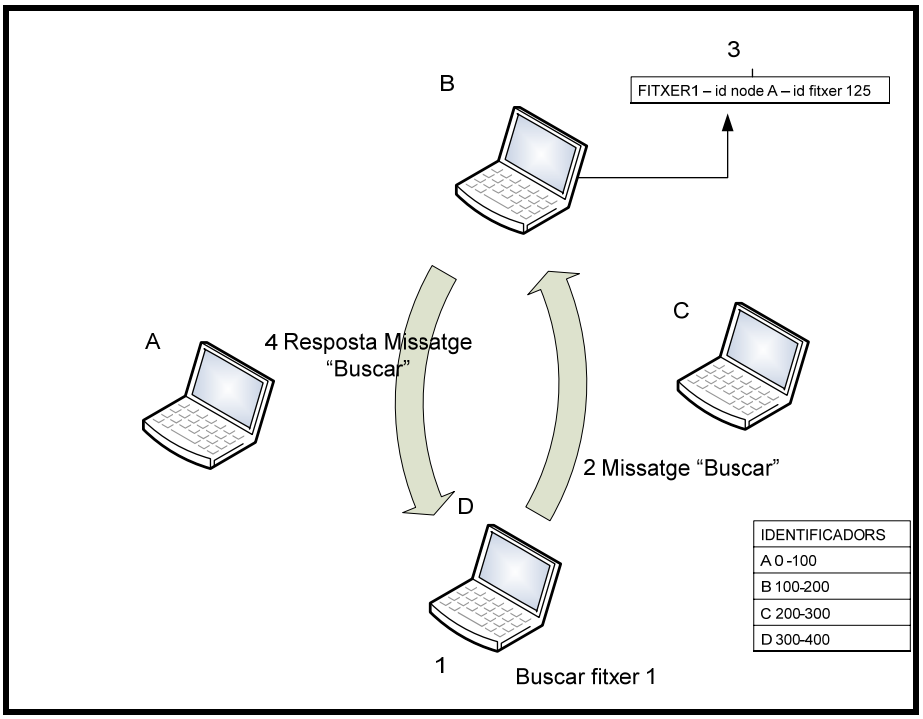
2: S'envia un missatge tipus "Despublicar" al node B, el missatge contindrà el identificador del fitxer a despublicar.

3: Al rebre el missatge el node B buscarà el identificador rebut, i el buscarà en la llista. Si aquest existeix l'eliminarà, i retornarà un missatge indicant la correcte execució.

4.1.3.- Buscar.

Aquesta funció bàsicament ens servirà per saber en quina maquina està situat el fitxer, obtindrem aquestes dades indicant nomes el nom del fitxer, amb ell obtindrem el Id , enviarem el missatge amb el Tipus buscar indicat en el missatge al node que gestioni el Id, i aquest ens retornarà la informació necessària.

Funcionament:



1: El node B busca el Fitxer 1, per fer-ho ha d'introduir el nom en l'aplicació que mitjançant una funció obtindrà el identificador assignat. Obtindrà el identificador 125 que està gestionat per el node B.

2: El node D enviarà un missatge del tipus "Buscar" al node B, el missatge contindrà el identificador del fitxer.

3: El node B busca el identificador dintre la taula i n'obté el identificador del node on està emmagatzemat el fitxer.

Pas 4: El node B respon el missatge "Buscar" indicant el identificador del node que conte el fitxer.

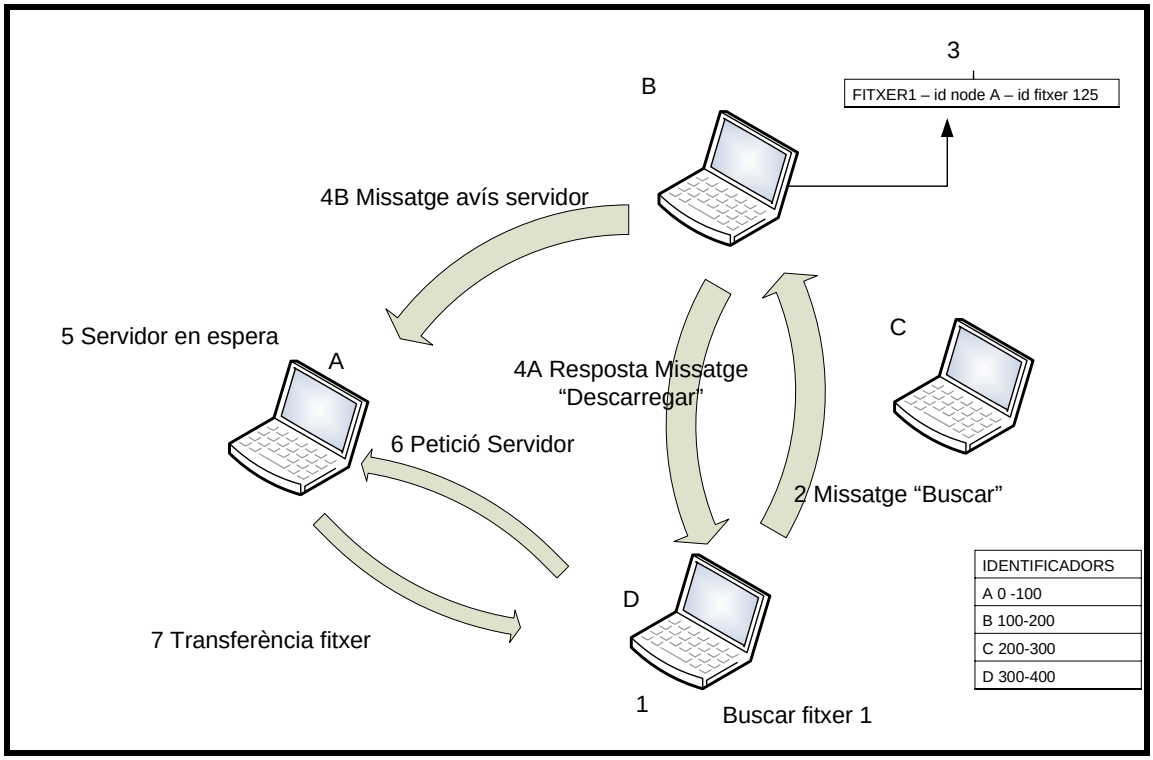
4.1.4.- Descarregar

Quan escollim la opció de descarregar un fitxer, haurem de indicar, tan sols el nom d'aquest. Amb el nom l'extraurem el identificador, i el anirem a buscar a el node que calgui. Quan aquest node rebi el missatge, comprovarà que el fitxer existeixi en la llista, i de ella n'extraurà la ip de el node que té guardat aquest fitxer. Quan tinguem aquestes dades enviarem, un missatge al client del fitxer

amb la adreça del Servidor i un missatge al servidor per indicar-li que es posi en marxa, per una possible petició.

El servidor rebrà el missatge i es posarà en espera de que se li demani un fitxer. El client rebrà el missatge i enviarà una petició de descarrega al Servidor. Finalment es connectaran servidor i client i es transferirà el fitxer.
El Fitxer s'envia en paquets de 1000 bytes, ja que després de provar varies configuracions es amb la que s'obté millor resultat.
Un cop descarregat el fitxer es paren el client i el servidor i l'aplicació tornarà a mostrar les opcions del menú.

Funcionament:



1: El node B vol descarregar el Fitxer 1, per fer-ho ha d'introduir el nom en l'aplicació que mitjançant una funció obtindrà el identificador assignat. Obtindrà el identificador 125 que està gestionat per el node B.

2: El node D enviarà un missatge del tipus "Descarregar" al node B, el missatge contindrà el identificador del fitxer.

3: El node B busca el identificador dintre la taula i n'obté el identificador del node on està emmagatzemat el fitxer.

4A: El node B respon el missatge "Descarregar" indicant el identificador del node que conte el fitxer i la seva IP. La IP es necessària, per a poder fer la transferència del fitxer, ja que el node sol·licitant li enviarà una petició directament al Servidor, sense utilitzar la xarxa FreePastry.

4B: EL node B també envia un missatge al node que conté el fitxer, per indicar-li que prepari el servidor per a una petició. Aquest al rebre el missatge, llençarà el servidor i esperarà una petició de descàrrega.

5: El node A en rebre el missatge del node B, llençarà el Servidor i esperarà la petició de descarrega.

6: El node D rep el missatge que li indica que existeix el fitxer i la Ip on es troba. Al rebre aquest missatge enviarà directament a la IP del servidor la petició de descarrega.

7: El Servidor (node A) rep la petició i comença la transferència TCP. S'ha de tenir en compte que la transferència es fa fora de la xarxa FreePastry, directament utilitza la xarxa MCL. Quan finalitzi la descarrega, tant el servidor com el client tornaran a el estat normal dintre de FreePastry.

4.2 Classes

En el Anex 1 es pot trobar una descripció de cada una de les classes utiltizades.

4.2.1. Diagrama de Classes.

En aquest apartat es pot veure el diagrama de classes de l'aplicació, en ell es mostren les dependencies de cada una de les classes existents.
Com es pot veure en el gràfic la classe principal es DistTutorial i totes les demès depenen de ella. En el diagrama es mostra també les dependencies entre les demes classes. La classe aplicació depèn directament de la Classe Distutorial, mentre que les classes ServidorFitxer, ClientFitxer i Missatge depenen de la classe aplicació. Per altre banda les classes ClientFitxer i ServidorFitxer tenen una classe que depèn d'elles cadascuna, en el cas de ServidorFitxer es la classe MissTeFitxer, i en el cas de la classe ClientFitxer es MissDonamFitxer.

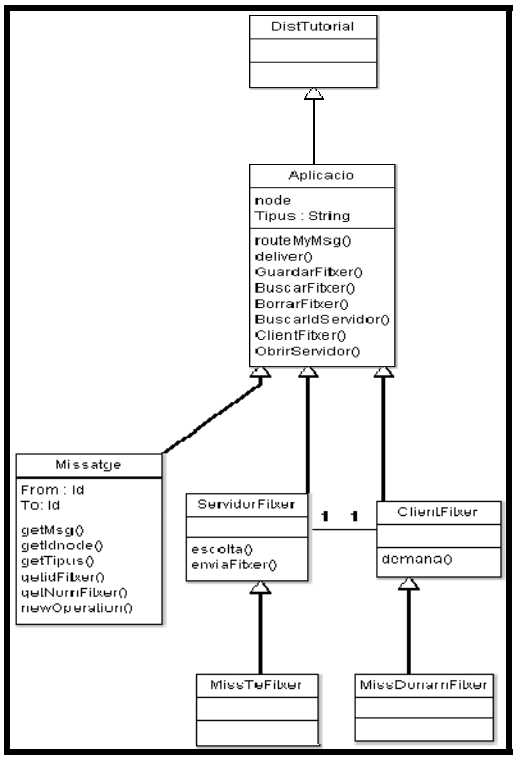


Fig. 8 Diagrama de classes

4.3 Modificacions posteriors

S'han fet algunes modificacions del programa original degudes a el comportament de l'aplicació en l'entorn utilitzat, les principals modificacions són dues.

El primer canvi rellevant en l'aplicació es el de la mida dels paquets enviats, ja que en un principi l'aplicació utilitzava paquets molt petits, de 50 Bytes, i realitzant les primeres proves es va demostrar que calia una mida de paquet mes gran per a poder millorar el throughput en una transferència, ja que amb paquets tan petits es perdia molt temps realitzant lectures de disc. Es van realitzar proves amb diferents mides de paquet i es va arribar a la decisió de utilitzar la mida de 1000 Bytes, amb aquesta mida s'obtenen bons resultats, i si augmentéssim gaire el paquet per sobre de 1200 Bytes provocaria fragmentació en els paquets TCP i això també disminuiria la velocitat de la transferència.

Per altre banda es va detectar un problema en el servidor, ja que quan es feia una transferència de un arxiu de mida gran el programa carregava massa la memòria i provocava un error que no permetia acabar la transferència.

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at rice.tutorial.lesson3.MissTeFitxer.<init>(MissTeFitxer.java:18)
        at rice.tutorial.lesson3.ServidorFitxer.enviaFitxer(ServidorFitxer.java:
78)
        at rice.tutorial.lesson3.ServidorFitxer.escolta(ServidorFitxer.java:41)
        at rice.tutorial.lesson3.ServidorFitxer.main(ServidorFitxer.java:25)
```

Això era degut a que la classe servidor creava un objecte nou per cada paquet que llegia del disc, això provocava que es carregués la memòria en excés. La solució al problema va ser fer un canvi en el codi del servidor, en el que en comptes de generar un objecte nou per cada paquet llegit, es reutilitzaria el mateix cada vegada, d'aquesta manera nomes es reservaria un espai de memòria per un objecte que es aniria modificant.

CAPITOL 5. PROVES I RESULTATS

En aquest apartat s'expliquen les diferents proves realitzades per a completar el estudi, i obtenir les dades necessàries.

Per analitzar completament la xarxa proposada en el projecte caldria realitzar una quantitat molt gran de proves, ja que s'hauria d'analitzar de forma estàtica i en moviment, a mes a mes s'hauria de estudiar per separat el comportament de la xarxa p2p i per la transferència de fitxers. En aquest cas es realitzarà el estudi en una xarxa sense moviment i sobre la transferència de fitxers, ja que s'ha pogut comprovar que la xarxa p2p no té gaire repercussió.

En les proves realitzades, s'utilitzen diferents mides de paquet i diferent número de nodes,amb o sense salts. Aquestes són necessàries per a extreure conclusions vàlides en el estudi realitzat. S'ha de tenir en compte que en tots els escenaris els ordenadors tenen configurada al mínim la potencia de la tarja wifi.

Software utilitzat :

- Iperf 1.7.0

S'utilitzarà Iperf per a generar tràfic tcp similar al de la aplicació i així poder comparar la transferència amb la de la aplicació sobre FreePastry.

- Java Runtime Environment jdk1.6.0_11.

S'utilitza el runtime de Java per a poder executar l'aplicació, i per a poder generar el anell FreePastry, també conté les eines utilitzades per a compilar les classes de Java.

- Windows XP Professional SP-2

Sistema operatiu utilitzat en totes les maquines per a fer les proves.

- Wireshark 1.0

S'utilitza Wireshark per a poder recollir la informació sobre la transferència de paquets entre els diferents nodes que componen la xarxa, i així poder després analitzar-los.

Hardware utilitzat:
El Hardware utilitzar per a realitzar les proves es el següent:

S :
Model: Sony Vaio VGN-FS315H
Tarja Wiffi: Intel® PRO/Wireless 2200BG Network connection
IP: 192.168.2.8

A:
Model: HP Compaq nx 6110
Tarja Wiffi: Intel® PRO/Wireless 2200BG Network connection
IP: 192.168.2.4

B:
Model: HP Compaq nx 6110
Tarja Wiffi: Intel® PRO/Wireless 2200BG Network connection
IP: 192.168.2.1

C:
Model: HP Compaq nx 6110
Tarja Wiffi: Intel® PRO/Wireless 2200BG Network connection
IP: 192.168.2.2

Les proves s'han realitzat en quatre etapes, per a poder avaluar les diferents possibilitats de transferència de fitxers.
Una primera etapa on s'utilitzen 2 nodes pròxims entre ells, on es fan les diferents proves de transferència de fitxers de diferents mides.
Una segona etapa on s'utilitzen 3 nodes, situats a mes distancia, forçant la realització de dos salts per la comunicació entre els nodes mes separats.
Una tercera etapa on s'ha mantingut la xarxa anterior, realitzant proves per observar el comportament, de la entrada de un nou node durant una transferència.
Una etapa final amb 4 nodes, amb la realització de proves forçant tres salts des de el servidor al client C.

5.1.- Escenaris de les proves

5.1.1.- Escenari 1:

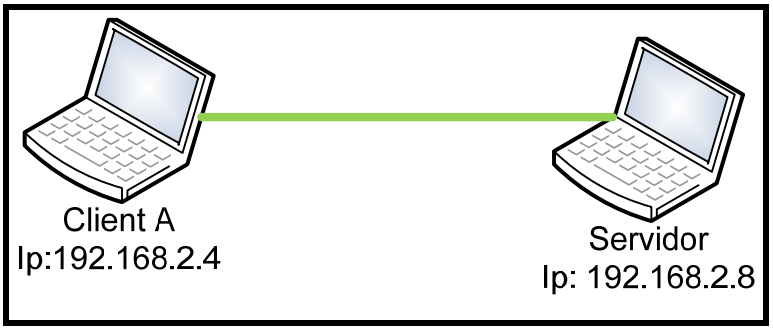
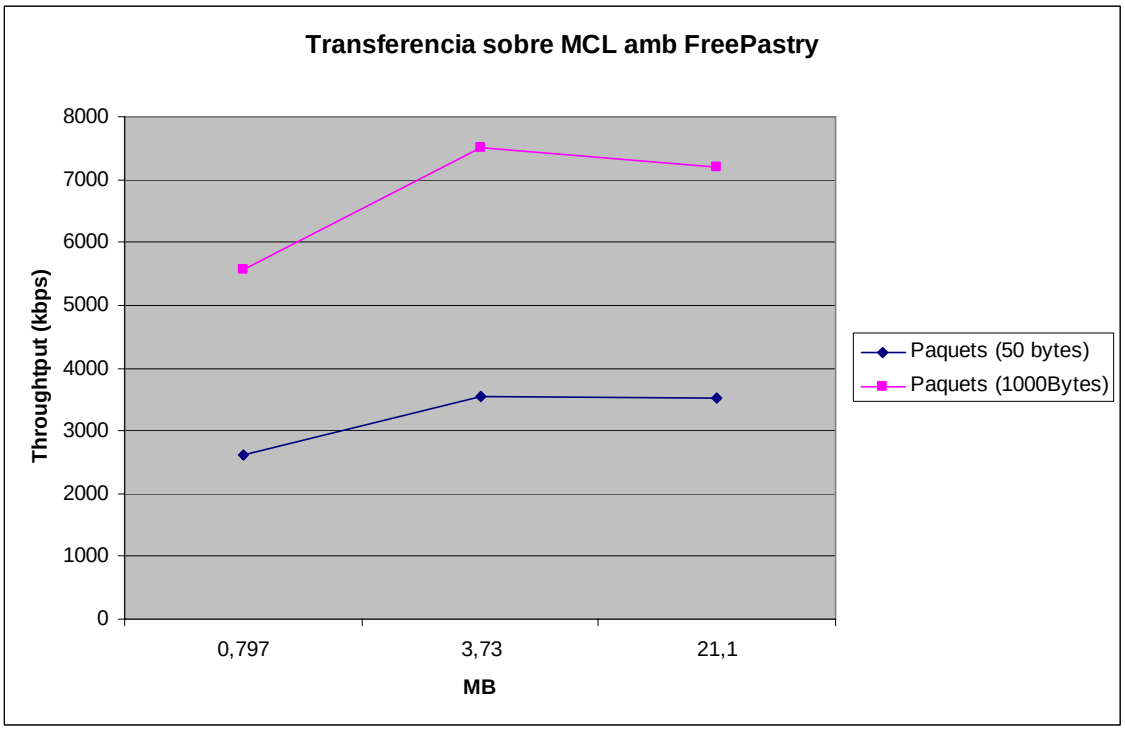


Fig. 9 Escenari 1

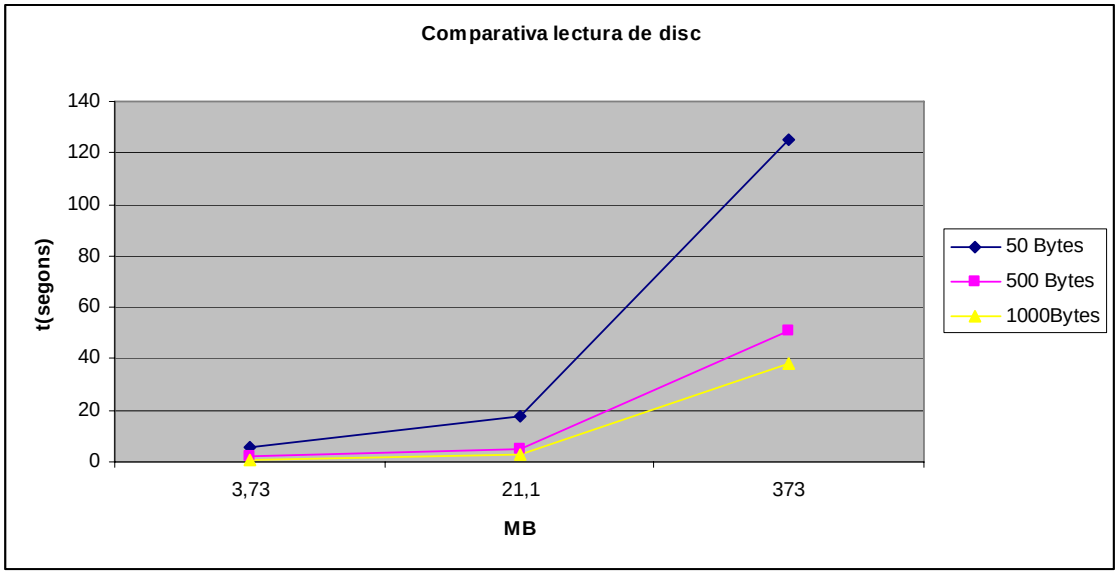
En aquest escenari tenim els 2 nodes molt propers, aproximadament 10 centímetres entre ells, sense cap obstacle entre els nodes, i amb connexió directa. S'han realitzat transferències entre els nodes per veure possibles variacions, i obtenir dades sobre la transferència de fitxers entre ells.

S'han realitzat les proves següents:

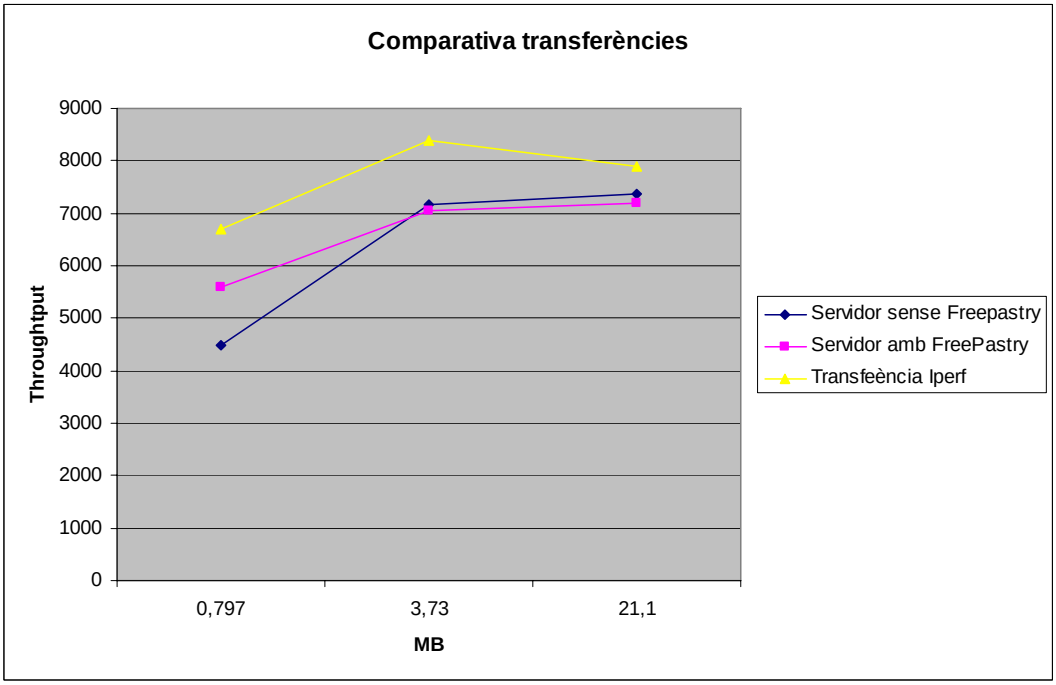
- Transferència de fitxers S→A (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL i FreePastry
- Transferència de fitxers S→B (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL i FreePastry
- Transferència de fitxers S→A (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL sense FreePastry
- Transferència de fitxers S→B (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL sense FreePastry
- Transferència de fitxers S→A (797KB/ 3,73MB/21,1MB) paquets de 1000 Bytes utilitzant Iperf.



Es va decidir fer proves amb dues mides de paquets diferents 50 i 1000 Bytes, es van notar grans diferencies de velocitat de transmissió entre elles com es pot observar el la gràfica, a mes gran la mida del paquet mes velocitat de transferència, i es poden enviar fitxers de mida mes gran.
Com es pot veure al augmentar la mida del paquet, el throughput de la transmissió augmenta considerablement, en el pas de 50 bytes a 1000 bytes el resultat es gairebé el doble de throughtput, això mostra la importància de aquest paràmetre, en la aplicació, cosa que a la hora de dissenyar no se li va donar rellevància deguda. Per tant aquest retard es degut a la lectura de disc, es a dir en el cas de 50 bytes trigarà mes per que haurà de llegir mes vegades del disc.



En la gràfica de comparativa de lectura de disc es demostra el retard que provoca la mida dels paquets a llegir del disc. Es pot veure que a mides mes petites de paquet el retard es mes gran proporcionalment.



Per un altre banda al comparar les transferències amb FreePastry i sense FreePastry com es pot observar no es veuen pràcticament afectades per la implementació ja que no varien gaire els resultats, en canvi si es comparen les dades amb les de la transmissió amb Iperf es pot apreciar que hi ha diferencies en el volum de treball, ja que el Iperf no opera sobre la maquina virtual de Java,

i això permet una millor transmissió. En aquest cas s'utilitzen sempre paquets de 1000 Bytes en una transmissió TCP.

5.1.2.- Escenari 2:

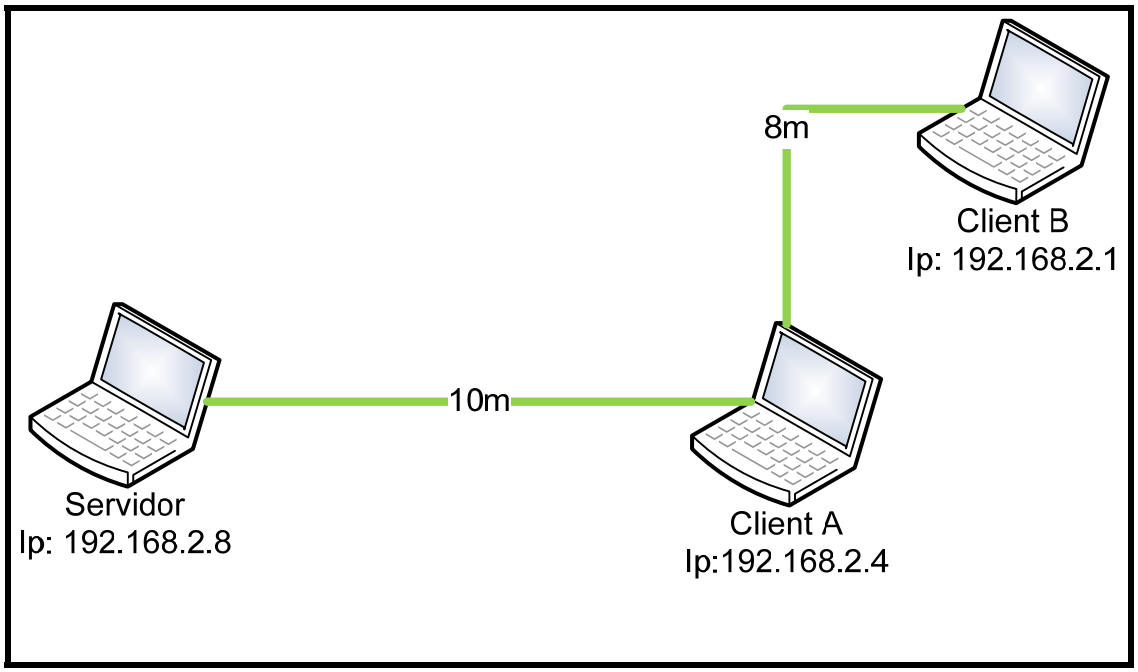


Fig. 10 Escenari 2

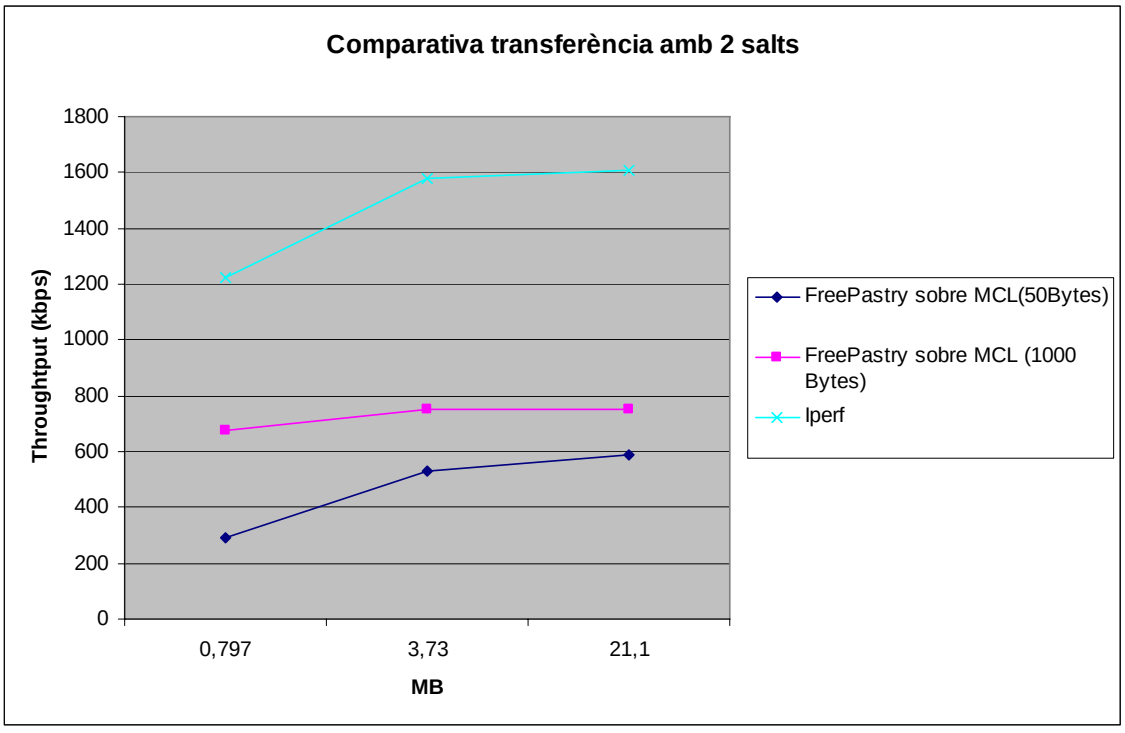
Com es pot veure en aquest nou escenari hi ha tres nodes, s'ha de tenir en compte es que el client B i el servidor no es poden veure directament, tal que per que hi hagi una transferència entre el servidor i el client B els paquets hauran de ser retransmesos per el client A, en la següent captura es pot veure com per arribar des de el servidor a el client A hi ha connexió directe (1 salt) i en canvi per arribar a B es necessita fer 2 salts.
Cal dir que entre el node Servidor i el Client A no existeixen obstacles, en canvi entre el Client B i el Client A existeixen obstacles, ja que el Client B es troba situat dintre de una aula i el client a esta en un passadís, això vol dir que hi ha una paret entre els 2 nodes, que afectarà a la transferència.

```
C:\>mcl lc
VA 1 Link Cache, 3 nodes of 4, Timeout -69s
94-77-09-88-6f-61: Hops 0 DMetric 0 CMetric 0 Prev -1
    From 2 To 1/94-77-09-88-6f-64
        TimeStamp -2s Usage 7 Metric 0.03-7Mbps-1 (1.38ms)
        Queue Drops 0 Failures 0 fraction 0
        ETX:
        TotSentProbes 500 LastProb 0.002
        PktPair:
        PairsSent 9 RepliesRcvd 9
        LastPktPair 1.20ms CurrMin 1.15ms
        NumValid 9 NumInvalid 0
94-77-09-88-6f-64: Hops 1 DMetric 10.56ms CMetric 10.56ms Prev 0
    From 1 To 1/94-77-09-88-6f-68
        TimeStamp 0s Usage 0 Metric 0.67-1Mbps-1 (31.24ms)
    From 1 To 2/94-77-09-88-6f-61
        TimeStamp 0s Usage 8 Metric 0.02-8Mbps-1 (1.21ms)
        RepliesSent 9
        TotRcvdProbes 489 FwdDeliv 24 RevDeliv 28
94-77-09-88-6f-68: Hops 2 DMetric 39.53ms CMetric 0 Prev 1
    From 1 To 1/94-77-09-88-6f-64
        TimeStamp -9s Usage 0 Metric 0.77-900Kbps-1 (58.55ms)
```

En aquest segon escenari, es valoraran dos factors principalment, la transferència entre 2 nodes amb una distancia significativa,en aquest cas uns 10 metres aproximadament, i la transferència entre dos nodes on cal realitzar un salt entremig per un tercer node, en el cas de transferir de el servidor al client B.

S'han realitzat les proves següents:

- Transferència de fitxers S→A (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL i FreePastry
- Transferència de fitxers S→B (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL i FreePastry
- Transferència de fitxers S→A (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL sense FreePastry
- Transferència de fitxers S→B (797KB/ 3,73MB/21,1MB/357MB) paquets de (50B i 1000B) sobre MCL sense FreePastry
- Transferència de fitxers S→B (797KB/ 3,73MB/21,1MB/357MB) paquets de 1000 B utilitzant Iperf



Com es pot apreciar en la gràfica quan tenim 2 salts, la transferència es rellenteix notablement i perd eficiència, ja que degut a la quantitat de paquets perduts que hi ha es realitzen moltes retransmissions. També es pot veure que la diferencia de throughtput amb l'iperf augmenta sensiblement a diferència de quan es una transmissió directe.

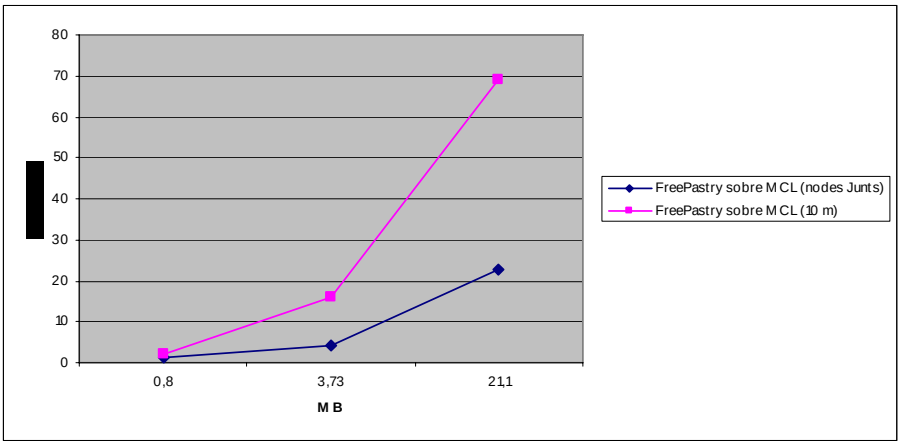
La pèrdua de tants paquets no es només deguda a la intervenció de un node nou sinó també serà deguda a la distancia i el entorn del escenari on hauran

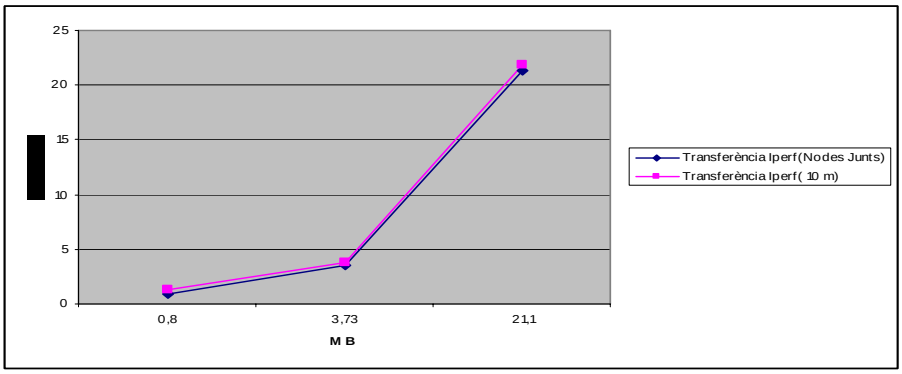
possibles interferències, per la existència de punts d'accés al voltant, i possibles rebots debuts a la situació dels nodes.

També podem veure que la diferència que es pot apreciar en el temps de transferència entre els paquets de 50 bytes i els de 1000 bytes, es deguda a el temps que es perd realitzant el accés a disc.

Hi ha molta diferència de throughtput entre la transferència en Iperf i la transferència sobre FreePastry. Principalment son degudes a que en Iperf s'utilitza tràfic sintètic, mentre que amb FreePastry s'utilitza tràfic real, i l'altre diferència rellevant es que Freepastry utilitzarà la màquina virtual de java , que introduïrà retards, mentre que a l'Iperf no li afectarà.

Gràfiques comparatives nodes junts /nodes separats(10 m)





En aquesta sèrie de resultats es pot veure clar que la distancia entre nodes pot ser molt important a l'hora de fer una transferència amb l'aplicació per FreePastry, ja que com es pot observar, triga mes del doble de temps en alguns casos, això es degut a interferències externes i possibles rebots en l'entorn. A mes distancia mes pèrdues de paquets i com a conseqüència mes retransmissions, això provoca un gran augment de temps per fer una transferència.

5.1.3.- Escenari 3:

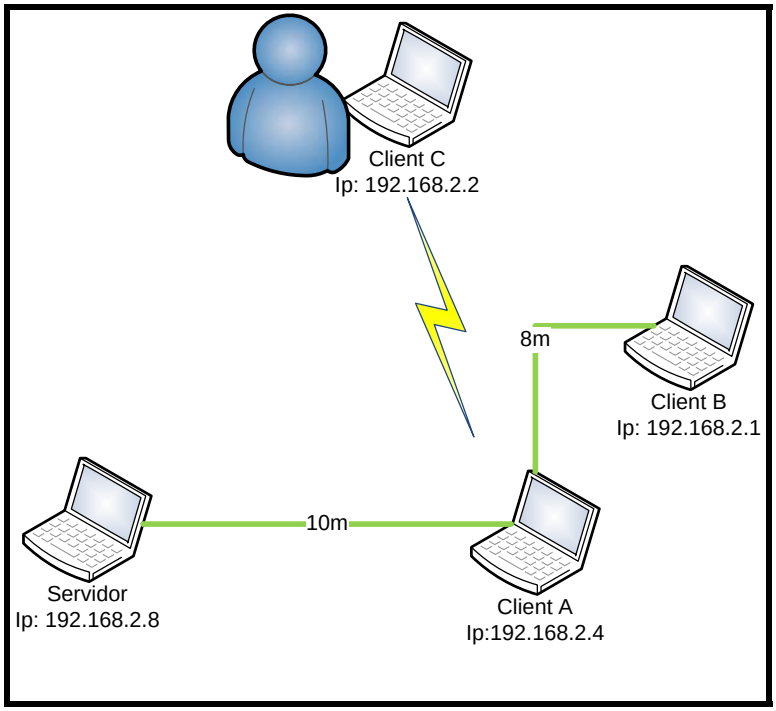


Fig. 11 Escenari 3

En aquest escenari la posició de el servidor del Client A i del Client B són la mateixa que en l'anterior, per tant entre el servidor i A no hi haurà obstacles mentre que entre el Client A i el Client B hi haurà una paret. El Client C en aquest cas es situarà pròxim a la posició de A.

En aquest nou escenari s'intenta avaluar la rellevància de la entrada de un nou node a la xarxa, es vol comprovar com afecta a els nodes que ja estan dintre, mentre fan una transmissió.

Per obtenir els resultats s'introdueix un nou node FreePastry a la xarxa m'entres 2 dels nodes estan realitzant la transferència d'un fitxer. També es realitza la prova de introduir un nou node a la xarxa mcl durant una transferència.

El resultat d'aquesta prova es prou aclaridor, ja que durant una transferència en la xarxa FreePastry no deixa entrar un nou node, i el fa esperar a acabar la transferència, per tant aquest no introduirà cap tipus de soroll a la xarxa. Es quedarà esperant fins que finalitzi el procés i seguidament obtindrà accés a la xarxa.

En el cas de la introducció de un nou node MCL en la xarxa Ad-hoc si que pot entrar a formar part, però el soroll introduït es mínim, MCL nomes envia paquets cada cert temps per comprovar el estat de la xarxa, i si els nodes estan estàtics no hi ha pràcticament soroll introduït per els nodes.

5.1.4.- Escenari 4:

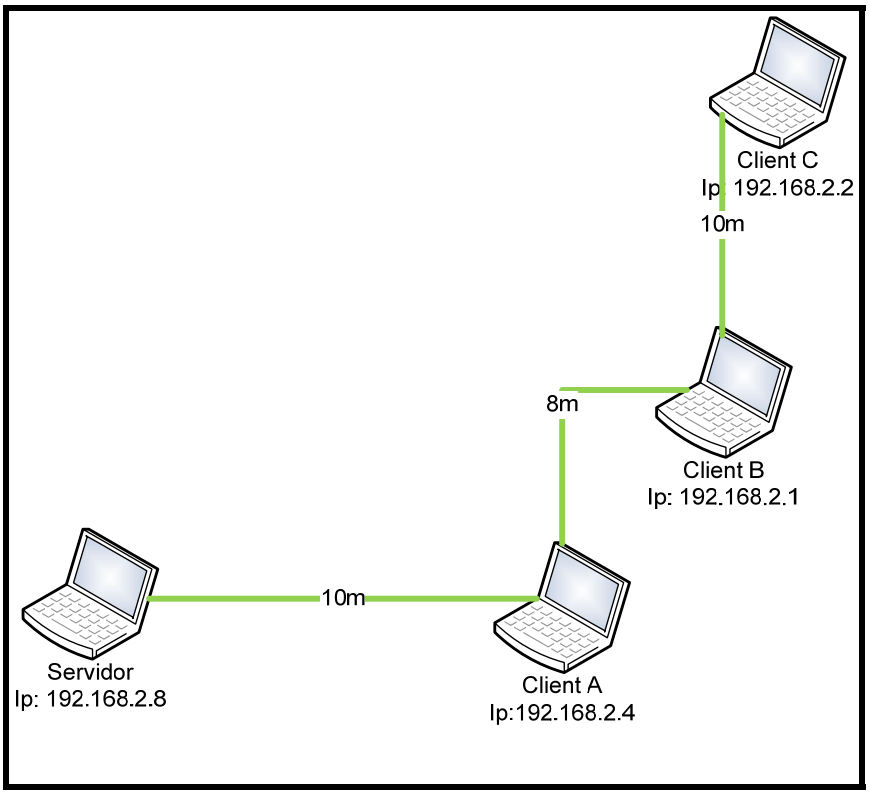
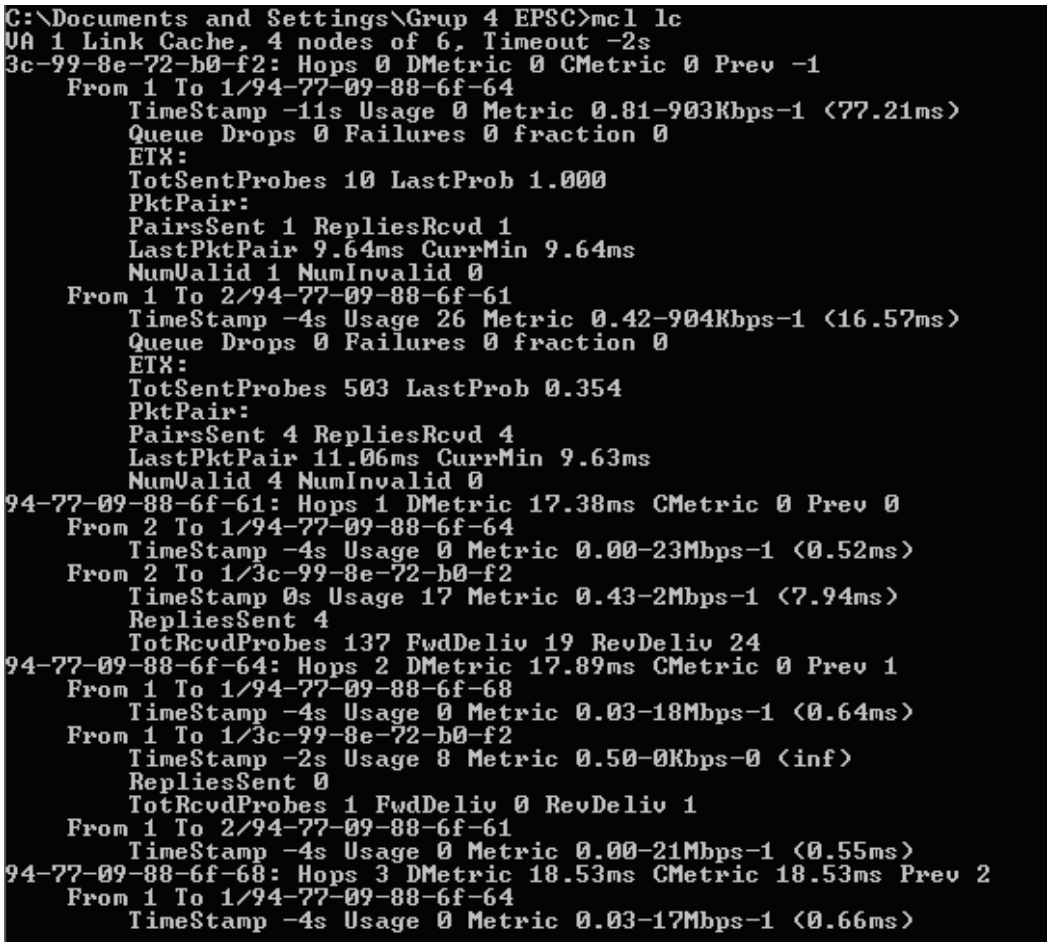


Fig. 12 Escenari 4

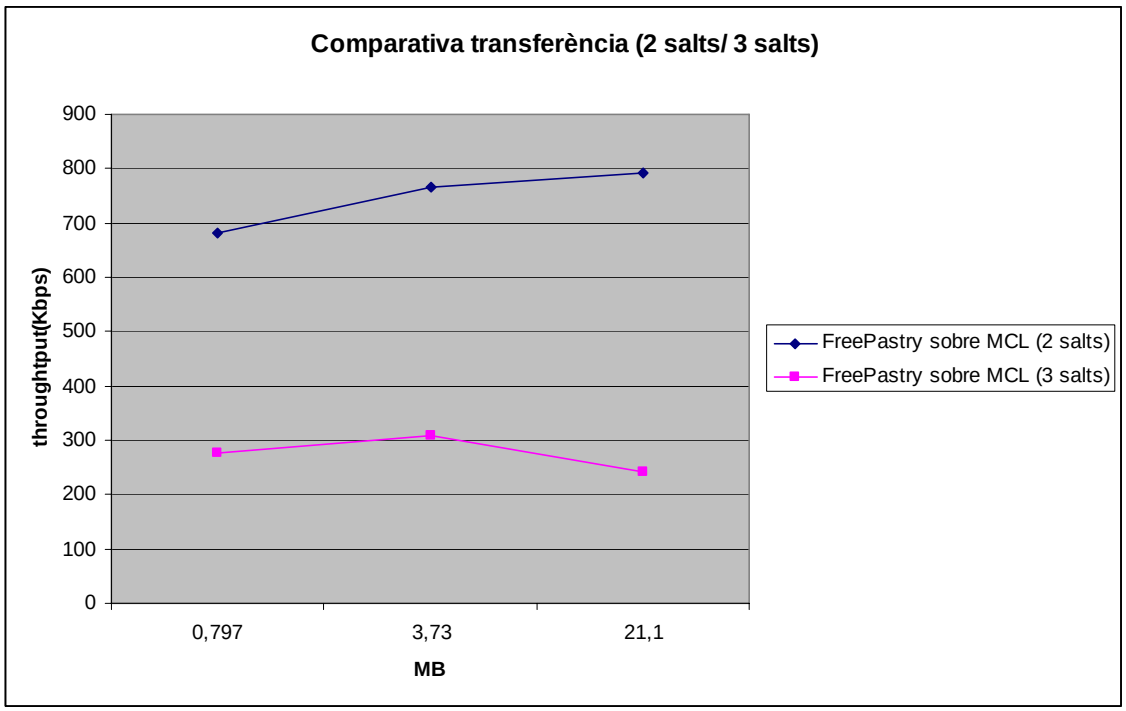
En aquest quart i últim escenari apareix un quart node, el client C, aquest node nomes es comunica directament amb el client B, es a dir per a poder fer una transferència del servidor al client C els paquets hauran de passar per el client A i per el client B sempre, en aquest escenari es vol extreure dades sobre la transferència d'un fitxer passant per 2 nodes entremitjos.
La posició del Servidor,Client A i el Client B es la mateixa mentre que el Client C es situarà a 10 metres de B, i amb 2 finestres tancades com obstacle, ja que estan situats en edificis diferents.



En aquest informe de MCL podem veure que entre el Client C (192.168.2.2) i el Servidor (192.168.2.8) hi ha 3 salts, la captura esta realitzada des de el client C.

S'han realitzat les proves següents:

- Transferència de fitxers S→C (797KB/3,73MB/21,1MB) paquets de (1000B) sobre MCL i FreePastry
- Transferència de fitxers S→C (797KB/3,73MB/21,1MB) paquets de (1000B) sobre MCL sense FreePastry

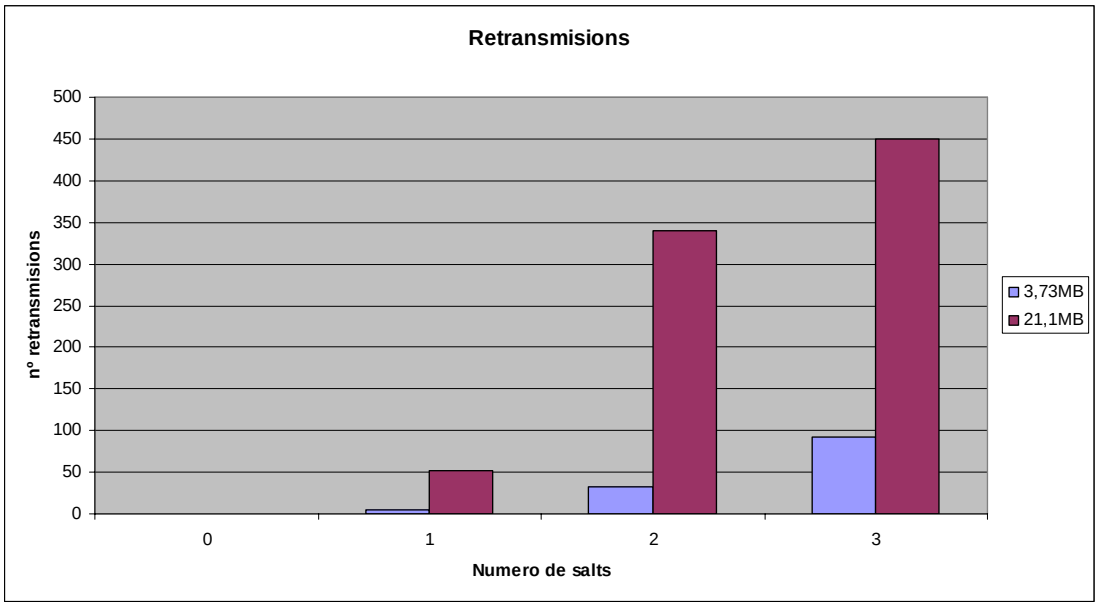


Amb aquestes proves es corrobora la ascendència dels retards en les transmissions en relació a la quantitat de salts en una transferència, i a la distància entre nodes. En la gràfica es pot veure que el throughtput decreix, en comparació a les proves fetes amb 2 salts, degut a la gran quantitat de paquets perduts que hi ha al afegir aquest tercer salt.

5.2. Resultats Globals

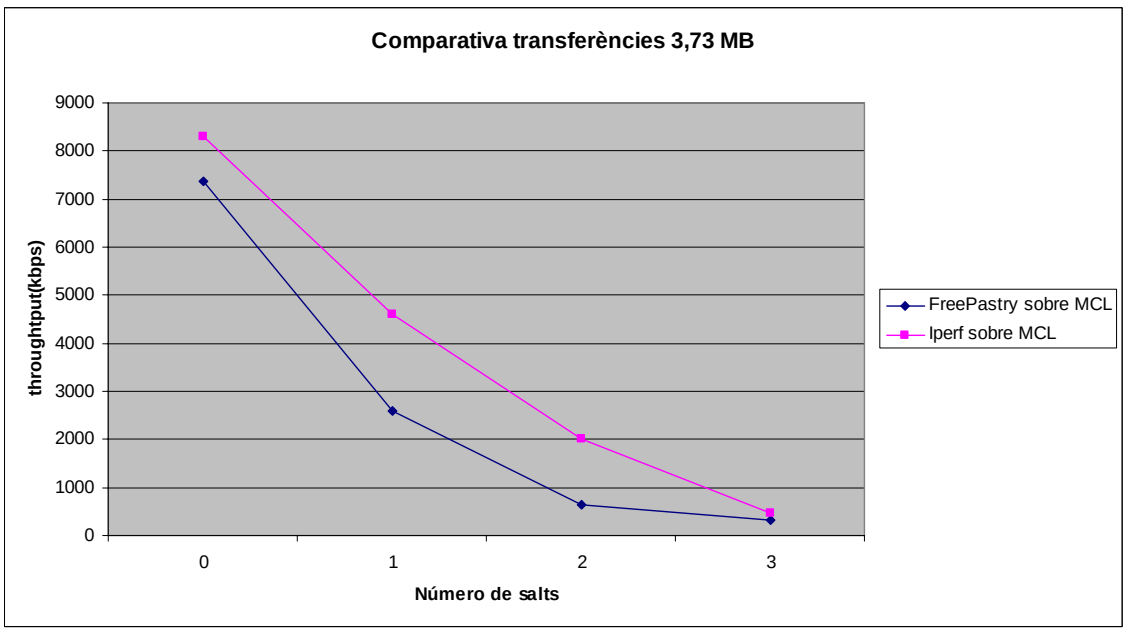
Amb la realització de les proves en els diferents escenaris, es poden comparar els resultats en relació a la quantitat de nodes.

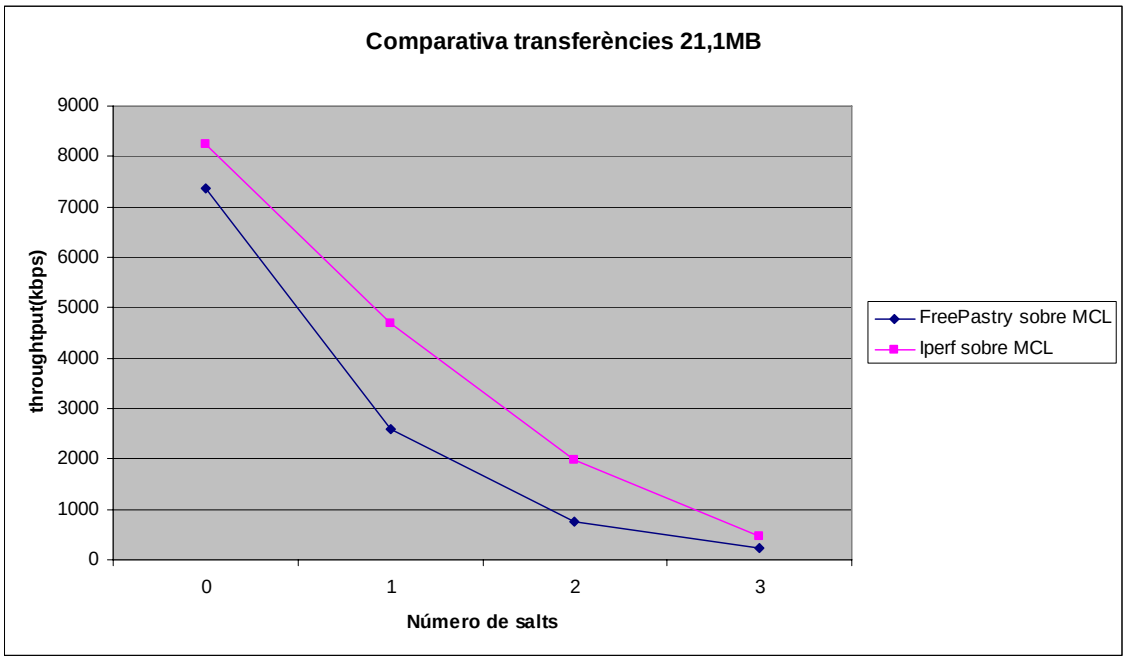
En les següents gràfiques es comparen els resultats dels diferents escenaris que s'han realitzat durant les proves.
En la primera gràfica es veuen les diferents retransmissions que hi ha depenent del numero de salts realitzats, amb els fitxers de 3.73MBytes i 21.1Mbytes.



Com podem veure les retransmissions creixen sensiblement a mesura que s'augmenta el numero de salts que han de realitzar els paquets, com ja s'ha mencionat aquestes retransmissions en gran mesura són directament relacionades amb la distancia entre nodes i el número de salts. El número de retransmissions es aproximat, realitzant una mitja entre les diferents proves realitzades. Aquestes proves són realitzades sempre amb l'aplicació en Java sobre FreePastry.

En les següents gràfiques es compara les transferències de fitxers depenent del numero de salts realitzats, per veure com varia el temps de transmissió.
Les gràfiques són per el cas del fitxer de 3.73 MBytes i el de 21.1MBytes.





En aquestes dues gràfiques es pot veure clarament com evoluciona la transferència d'un fitxer a mesura que augmentem el numero de nodes. La corba decreix considerablement a mesura que afegim nodes, per cada node de mes disminueix considerablement el throughput, també es pot observar que la diferencia entre la transferència sobre FreePastry i la transferència realitzada amb iperf, segueixen la mateixa tendència decreixent, per tant aquesta tendència descendent, es pot dir, que es deguda a MCL, i no a l'aplicació FreePastry.

Amb aquests resultats es pot dir que FreePastry no afecta gaire a la transferència amb el servidor realitzat en Java, ja que el soroll Introduït es nul, FreePastry no envia missatges ni al client ni al servidor de la transferència, fins que aquesta no acaba. Per altre banda això pot provocar, que els altres nodes considerin que el nodes que estan transmeten reben s'hagin desconnectat de el anell,si aquests no envien el missatge corresponent en el temps indicat.

En general després de veure els resultats en totes les proves, es pot dir que la utilització de FreePastry sobre una xarxa adhoc amb MCL seria una opció útil nomes en certs casos. Per que realment es pogués utilitzar com a opció viable la xarxa hauria de constar de proximitat entre els nodes i el mínim número de salts possibles entre ells, ja que com s'ha pogut veure en el estudi, tant la distancia entre nodes com la realització de salts entre servidor i client provoca molta pèrdua de paquets i molta lentitud en la transferència.

El millor escenari per la utilització del projecte proposat es basaria en 2 aspectes fonamentals, en primer lloc la proximitat de tots els nodes de la xarxa, per evitar salts, i distancies massa llargues, ja que en els 2 casos es disminueix molt la eficiència, i per altre banda caldria imposar un ordre a l'hora de realitzar les transferències, ja que no seria útil si els usuaris volguessin transferir alhora.

CAPITOL 6. CONCLUSIONS

Una vegada ja esta completada l'aplicació, funciona correctament, i s'han fet les proves corresponents per fer l'anàlisi, cal extreure un conjunt de conclusions, tot analitzant els resultats obtinguts.

6.1. Conclusions Personals

La realització d'aquest projecte, m'ha prestat l'oportunitat de estudiar amb mes profunditat les xarxes adhoc i el funcionament de les xarxes P2P, cosa que m'ha ajudat a millorar molt el meu coneixement sobre aquest tipus de xarxa.

La idea principal del projecte es fer un estudi sobre una possible utilització de transferències en una MANET en un entorn real, d'aquesta manera també he pogut veure l'ús que tindria en alguns tipus de empresa, que podria utilitzar- se com interfície de transferència de fitxers interna.

Per un altre banda, el tenir que crear una aplicació utilitzant Java, m'ha obligat a realitzar un estudi del llenguatge per a poder programar les diferents funcions, això m'ajuda't a conèixer millor el llenguatge, ja que tenia unes nocions molt bàsiques d'aquest. També he hagut de estudiar la tecnologia Pastry ja que per crear la nostre xarxa P2P hem utilitzat FreePastry, aquesta part per mi ha estat la mes complicada ja que mai havia estudiat un tipus de tecnologia com aquesta realitzada en Java. Però això m'ha ajudat a entendre molts conceptes relacionats amb la tecnologia p2p que potser no m'havien quedat totalment clars durant la carrera, ja que, moltes vegades, s'havien vist de forma poc detallada.

La realització de les proves i l'estudi posterior per mi ha estat la part més gratificant, veient així com, gràcies a la feina realitzada, s'obtenien una sèrie de resultats útils i reals. Ha estat un interessant procés el de recrear els escenaris dissenyats anteriorment, utilitzant el material i el espai necessari.

En general considero la realització del projecte ha estat un procés molt positiu i que m'ha aportat molt personalment. Les diferents fases del treball des de la documentació fins la redacció del treball han estat processos que m'han ensenyat a utilitzar els recursos disponibles, i sobretot la resolució dels problemes durant el treball. Crec que es una experiència important per mi i que en un futur immediat em serà molt útil.

6.2. Comprovació de la planificació del projecte

Es pot dir que la realització del projecte ha estat molt positiva, ja que s'han complert els terminis inicials, i s'han obtingut resultats necessaris per extreure'n conclusions.

La planificació inicial establerta s'ha seguit amb cert rigor, i nomes hi ha algun retard en la primera fase de la creació de l'aplicació, ja que ha calgut dedicar mes temps del esperat per a l'estudi del funcionament de FreePastry, hi ha sorgit alguns problemes durant la programació en Java, deguts a el escàs coneixement del llenguatge, que han requerit la modificació del codi varies vegades. Aquestes modificacions però, han ajudat a entendre millor la funcionalitat del projecte i els problemes reals de la creació de una aplicació dedicada a un entorn com el de una xarxa adhoc.

La realització de les proves, s'ha aconseguit en el temps esperat, tot i que algunes de les proves s'han repetit varies vegades per obtenir més varietat en els resultats. L'execució d'aquestes no ha comportat molts problemes, s'han recreat els escenaris dissenyats i s'han realitzat les diferents proves sobre la xarxa.

L'anàlisi dels resultats ha estat un procés llarg tal i com s'esperava, ja que es tenien de revisar multitud de captures realitzades durant les proves, però s'han obtingut resultats prou clars per treure les conclusions esperades, i així realitzar un estudi amb les dades suficients.

6.3. Objectius assolits.

L'objectiu principal era fer un estudi sobre el funcionament de una xarxa P2P sobre una determinada xarxa, que en aquest cas es tractava de una MANET, i obtenir les dades necessàries per poder avaluarla. En aquest projecte, doncs, s'ha complert el objectiu primari, ja que s'ha aconseguit uns resultats que ens permeten obtenir una sèrie de conclusions, sobre el cas estudiat.

Aquest objectiu principal, es dividia en petites parts que s'han anat complint per separat, aquests corresponien amb les diferents fases del projecte, el objectiu de la primera fase era crear l'aplicació, una aplicació amb la qual es poques gestionar la transferència la publicació i la localització de fitxers. Aquest objectiu s'ha complert, ja que la aplicació realitzada contempla totes aquestes accions.

L'objectiu relacionat amb la segona fase era obtenir les dades necessàries de les proves realitzades per a realitzar un estudi satisfactori del qual s'obtingués informació rellevant. Aquest objectiu també s'ha complert, la informació obtinguda ha estat suficient per extreure'n conclusions, tot i que hi ha consciencia que el estudi podria continuar i obtenir més dades, realitzant més proves. De totes maneres les dades obtingudes són suficients per el cas estudiat.

El tercer objectiu o objectiu de la tercera fase, era utilitzar els resultats de la anterior fase per estudiar el comportament de tràfic TCP en una xarxa adhoc. Aquest estudi s'ha mostrat en forma de gràfiques, per poder veure clarament els resultats i les comparatives de les proves realitzades.

6.4. Possibles millores.

Aquest projecte neix de la idea de estudiar tràfic TCP en una xarxa adhoc, és un tema que per analitzar completament, s'haurien de dedicar molts recursos i temps, per tant, s'ha tingut d'acotar en base a uns conceptes més concrets, això provoca que durant l'execució del projecte es trobin moltes possibles millores o ampliacions, que servirien per tenir resultats molt més extensos, cosa que pel temps i els recursos que caldria dedicar són inviables. Aquestes són algunes de les millores que es podrien implementar en un futur en el projecte. Es podrien dividir en dues parts, les possibles millores de l'aplicació creada i les millores sobre les proves que s'han realitzat.

6.4.1.-Millores de l'aplicació.

En el apartat de millores per l'aplicació, es pot dir que n'hi ha algunes que podrien ser importants, la primera seria la de poder realitzar varies transmissions simultànies sobre la xarxa, això permetria la transferència de varis fitxers a l'hora entre els nodes, i permetria un estudi més extens sobre el tràfic TCP, per a realitzar aquesta millora hauríem d'utilitzar threads o processos que ens permetéssin llençar varis servidors o clients en cada node. Seria interessant veure la resposta de la xarxa amb varies transferències simultànies actives.

Per una altra part es podria implementar el buffering en l'aplicació, d'aquesta manera podríem reduir el temps de transferència ja que reduiríem el temps de lectura de disc. També es podria incloure la opció de transferir en UDP, això permetria la comparació dels dos tipus de tràfic IP,i veure quin dels 2 sistemes és més efectiu per aquest tipus de xarxes. Són tres millores que es podrien realitzar amb una modificació del codi relativament senzilla.

6.4.2.- Millores de les proves.

També es podrien afegir proves a les realitzades per treure'n altres conclusions, com per exemple es podria afegir mobilitat als nodes i veure com afecta al tràfic (reenviament, throughput, etc.) així es podria veure com afecta el protocol reactiu a la transferència, ja que en el moment de moure el node s'hauran d'enviar missatges per a poder reorganitzar l'arquitectura de la xarxa. Fent aquesta prova veuríem el pés d'aquests missatges que s'envien els nodes i com afecten a la transferència en curs.

Una altre prova adequada podria ser testar altres protocols per a crear la xarxa adhoc en substitució de MCL, com ara podria ser BATMAN, això ens permetria veure quin protocol afegeix mes flux de missatges a la xarxa, i quin es el protocol que ens introdueix menys soroll, es a dir, es més efectiu a l'hora de suportar tràfic TCP (transferir fitxers).

Finalment es podrien seguir fent proves amb més salts i més nodes per comprovar al degradació de les prestacions de la xarxa, i veure el límit de les possibilitats de la proposta realitzada.

CAPITOL 7. BIBLIOGRAFIA

[1] FreePastry, http://freepastry.org/FreePastry/

[2] MCL, http://research.microsoft.com/en-us/projects/mesh/

[3] Java, http://java.sun.com/javase/downloads/index.jsp

[4] wikipedia, http://es.wikipedia.org/wiki/Wikipedia:Portada

[5] MASTER THESIS, A real-world implementation and parametrization of mobile ad-hoc networks.

[6] Xarxes Ad-hoc, http://www.oreillynet.com/pub/a/wireless/2004/01/22/wirelessmesh.html

CAPITOL 8. ANEXOS

A.1. Classes de l'aplicació.

DistTutorial.Class
La classe principal es la DistTutorial.Class aquesta classe conte el Main i s'encarrega de generar el node dintre de l'anell FreePastry, en cas de que no existeixi el anell, es generarà el node com a node principal, es a dir generarem el primer node de un anell. Es genera un objecte del tipus aplicació que després ens servirà per a poder enviar els diferents missatges.

Aplicació.Class

En aquesta classe es troben les diferents funcions del programa, des de aquí enviarem i rebrem els diferents tipus de missatges. I els tractarem en cas de rebre'ls. Quan volem enviar un nou missatge, generarem un objecte del tipus Missatge i li introduirem les dades necessàries com el nom del fitxer, la operació que volem fer, el identificador destí, identificador origen.

Quan rebem un missatge, la classe Aplicació s'encarrega de comprovar el tipus de missatge que es, un cop definit depenent d'això executarà una funció o una altre.

Missatge.Class
Es la classe que defineix el missatge que enviarem,ens indica quins elements formen el missatge que hem enviat o em rebut, i des de ella es pot extreure tota la informació que conté el missatge.

ServidorFitxer.Class
Es la classe que genera un servidor per e poder transferir un fitxer que està disponible en la maquina, aquest servidor es crida quan es reclama un fitxer a la maquina. Quan l'aplicació rep un missatge del tipus "petició" automàticament es genera un servidor que esperarà a rebre la demanda de transferència per part del client.

ClientFitxer.Class

Es el client del fitxer, es cridarà a aquesta classe quan es demani la transferència de un fitxer que es troba en un altre maquina.

MissDonamFitxer.Class
Es el tipus de missatge que s'utilitza per a rebre el fitxer transferit.

MissTeFitxer.Class
Aquesta classe indica el tipus de missatge que enviem, en ell definirem la mida de paquet que llegim cada vegada que enviem.