Títol: Sistema Meteorològic Distribuït

Volum: 1

Alumne: Ramon Safont Gutiérrez

Director/Ponent: Toni Cortés Rosselló

Departament: AC

Data: 29 de Juny de 2009

Agraïments

A la meva família

Als tots els meus avis que m'han cuidat i als meus pares, que han donat lloc a la meva existència. En especial vull agrair a les tietes: Conxita, Mercè i a l'Anna el seu suport i també a la meva germana Irene que també s'obre camí en una carrera. I com no a la meva àvia, Pilar, per cuidar-me tot aquest temps.

A gfibers

Per que són el millor regal que m'emporto d'aquesta facultat i els tinc com la meva segona família. Ells són: Alex, Carlos, David, Iván, Oriol, Ruth, Sara, Sergi i Vanesa.

Als meus mestres

La gent del CEPBA: Juanjo, Javi, Xavi, Raül, Rodi, Roger, Àlex, Iván, Jairo, Romàn, David, Montse i Vicenç; més enllà de excel·lents companys: un grup de gent per treure's el barret. Vull agrair especialment en Xavi Martorell i, com no, en Toni Cortés: perquè sense ells aquest projecte mai hagués arribat a cap port. A Xavier Muñoz per tota la seva guia.

Als meus companys de camí

En Jordi Centeno, Noemí i Josebe per caminar junts més enllà d'aquells 300 km's a peu. A Isabel per totes les classes que dóna a dins i a fora de l'aula. A Marta F. Borsot per cada pas fet. A Diana i a Marta perquè hem crescut junts. Gràcies a Firàs i Marta per compartir el bo i millor. Gràcies per donar-me més punts de vista Christophe. A Oliver Ponce per mostrar a tothom com superar-se cada dia.

Índex

1 Introducció

1.1 Sobre el projecte

Aquest projecte de fi de carrera és una iniciativa personal relacionada amb el món de la meteorologia. La idea que hi ha al darrera d'aquest projecte gira al voltant d'un dels problemes habituals en el món de la meteorologia amateur, que és la comunicació de dades meteorològiques que, de forma individual, són obtingudes pels afeccionats.

1.2 Motivació

La meteorologia i la informàtica poc han tingut a veure fins fa no gaire quan, amb la ajuda dels grans centres de càlcul, s'ha començat a usar tècniques diverses per recopilar dades, realitzar càlculs i predir situacions meteorològiques diverses [1].

Des de ja fa força anys, la meva afecció a la meteorologia unida a la meva afecció a la informàtica m'han motivat a desenvolupar petits programes per ajudar a enregistrar dades meteorològiques amb els instruments que tenia aleshores.

El problema que intenta abordar aquest projecte és conjunt a molts dels afeccionats a la meteorologia que durant anys hem estat atents a les dades i les observacions que s'han originat dins de les comunitats d'Internet d'arreu: compartir les dades que cadascú anava enregistrant, tenint en compte la infraestructura disponible.

Tot això ha anat derivant cap a un projecte d'unes dimensions majors i amb unes fites més

ambicioses. Afortunadament, els coneixements que he pogut obtenir d'aquesta carrera sumat a la experiència laboral obtinguda, m'han fet obrir noves vies per treure molt més suc a allò que va començar com un simple entreteniment.

1.3 Objectius del projecte

L'objectiu a assolir és la construcció d'un sistema que tot fent ús de les capacitats de infraestructura existent, faciliti la comunicació de dades meteorològiques entre usuaris. Aquest objectiu passa per:

1. No centralitzar la comunicació de dades meteorològiques.
2. Fer que el sistema sigui ampliable en funcionalitats.
3. Limitar-se a usar la infraestructura existent.

1.4 Descripció

Quan es parla d'un Sistema Meteorològic Distribuït el que es fa referència és a la distribució de les dades que circulen entre els diversos usuaris del sistema. En concret, aquesta distribució de dades es fa via xarxa.

Per tal de facilitar la construcció del sistema es parteix del fet que hi ha dos tipus d'usuaris, bàsicament:

1. Un usuari que donarà les seves dades meteorològiques i les cedirà a la xarxa. Aquest

usuari podrà obtenir les dades usant diversos medis, com per exemple, estacions meteorològiques professionals.

2. Un usuari que llegirà les diverses dades que els anteriors usuaris llencen a la xarxa. Aquest usuari podrà visualitzar les dades enviades de manera conjunta i en un format determinat.

Expressat en un dibuix obtenim un esquema com aquest:

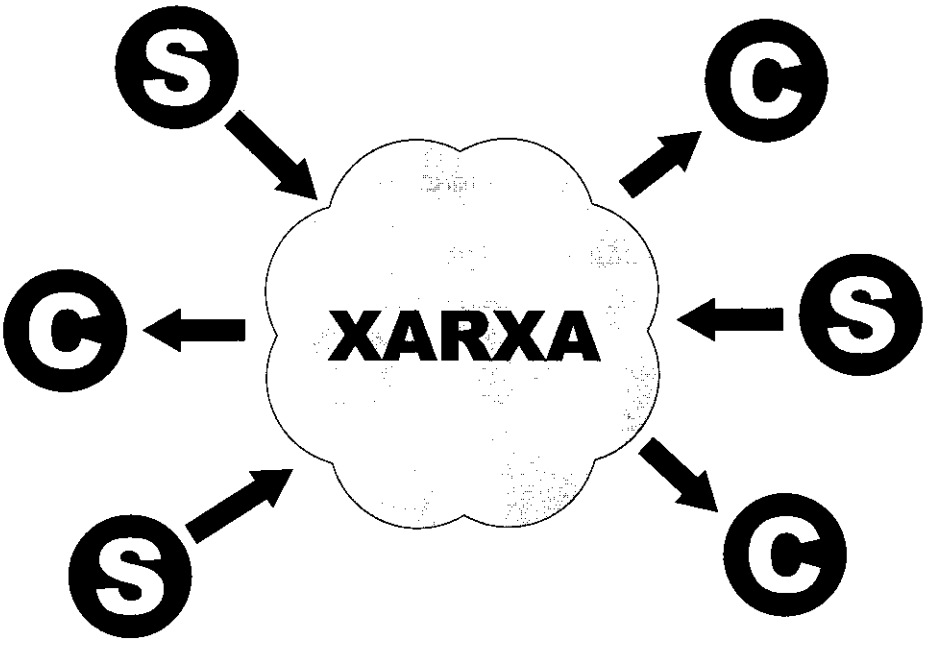


FIGURA 1A

En la figura 1A es pot observar com hi ha usuaris de diversos tipus que obtenen i envien les dades meteorològiques. Els usuaris que envien dades meteorològiques (mencionat prèviament en el punt 1) se'ls ha simbolitzat amb una S, mentre que els usuaris que reben aquestes dades se'ls ha simbolitzat amb una C (mencionats en el punt 2).

Aquest és l'esquema bàsic del sistema sobre el que ha partit el projecte. En els successius capítols es detallarà el disseny i el funcionament del mateix.

1.4.1 Exemple

Un usuari posseeix una estació meteorològica i vol compartir les seves observacions. Aquestes observacions, que envia en temps real, seran visualitzades per altres usuaris, que també estan connectades a aquesta xarxa.

L'usuari posseidor de la estació meteorològica es connecta a la xarxa mitjançant un programa que recollirà les dades de la seva estació i les enviarà a altres usuaris també connectats a la xarxa que tindran un programa amb el que podran visualitzar aquestes dades. Quan s'envia una observació no se li enviarà a tothom que hi hagi a la xarxa, només a un petit grup de usuaris: seran ells qui s'encarregaran de fer arribar la observació a d'altres usuaris fins que es cobreixi la totalitat d'usuaris que té la xarxa.

Aquest funcionament passa per a qualsevol paquet d'informació que circula per la xarxa: els usuaris actuen com a repetidors: reenvien la informació que els arriba a un altre conjunt d'usuaris fins a cobrir la totalitat de la xarxa.

1.5 Abast del projecte

De tot el sistema detallat en aquest projecte, el que s'ha dissenyat és l'esquema bàsic, és a dir, les parts mínimes del sistema. Aquestes parts comprenen l'objectiu inicial del projecte, que és: comunicar dades meteorològiques entre usuaris, els quals, poden tenir un rol de lectura de dades meteorològiques de la xarxa, o d'enviament de les mateixes cap a la xarxa.

Per tal de mostrar el funcionament del sistema s'ha implementat un prototip que permeti veure la manera en que podria treballar una xarxa real.

Cal tenir en compte que això deixa moltes funcionalitats que es poden afegir al projecte original fent més complet aquest sistema. Entre d'altre funcionalitats podríem destacar el filtratge de dades, la publicació de dades via web o l'emmagatzematge d'un històric de dades en una base de dades.

1.6 Desenvolupament global

La metodologia de desenvolupament que s'ha fet servir en aquest projecte és una part d'una metodologia major anomenada RUP [2]. El RUP és un conjunt de bones pràctiques a seguir en el desenvolupament del software, no té un camí concret sinó un cúmul de tècniques i procediments per tal de desenvolupar el software. El propi desenvolupador juga el paper de decidir quines d'aquestes tècniques són convenients per tal de dissenyar

el sistema software.

A més d'usar el RUP també s'han usat altres tipus de diagrames que han estat necessaris per tal de dissenyar el sistema, tot tenint present l'objectiu ser més concís en la descripció del mateix.

1.6.1 Desenvolupament del software

Hi ha molts paradigmes de desenvolupament del software, cadascun d'ells aporta una metodologia de desenvolupament específica.

En el cas que ens ocupa, s'ha seguit el model de desenvolupament iteratiu [3] que és el propi de la metodologia de desenvolupament RUP.

S'ha descartat el model en cascada [4] degut a que el procés que deriva d'ell requereix una especificació, anàlisi i disseny complets, que corresponen a una feina molt exhaustiva per a un prototip. El model en espiral [5] també s'ha descartat perquè va més enllà de l'abast del projecte: no es vol construir un software complet sinó un prototip del mateix.

No es segueix el RUP de manera estricta donat la quantitat d'artefactes (documentació i diversitat de diagrames) a realitzar, factor que hagués alentit molt el desenvolupament.

1.6.1.1 Model iteratiu

Els avantatges trobats que han inclinat la balança cap a aquest model han estat la continua millora del sistema a través de refinar el disseny, la implementació i tests continus.

Un dels motius pels que s'ha seguit el model iteratiu és el fet que no hi ha cap model previ existent d'un software d'aquestes característiques, per tant, cal revisar contínuament la feina feta per tal no perdre l'horitzó tot reevaluant el model construït i refinant el disseny de les parts més complexes del sistema.

El següent gràfic il·lustra aquest model amb les etapes corresponents:

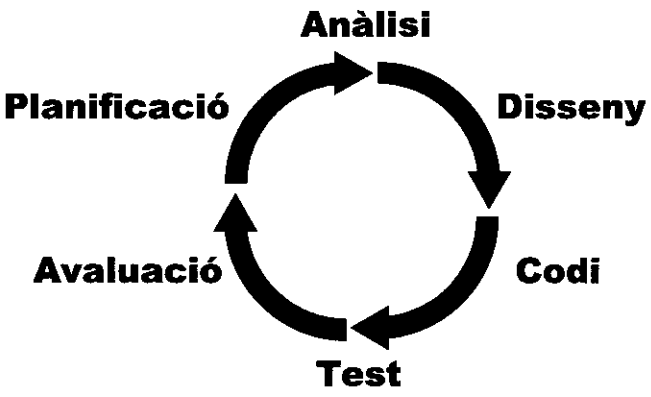


FIGURA 2A

1.7 Metodologia de disseny

Per al disseny del sistema s'ha fet servir altres gràfics que han servit de suport de cara a

modelar el sistema i els seus components, tals com, grafs d'estat o diagrames de xarxa.

1.8 Punt de partida

La primera aproximació cap a un software meteorològic personal no sol ser la comunicació de dades sinó emmagatzemar un històric de les mateixes, és a dir, maneres de guardar les observacions fetes. Aquest escenari és típicament el de l'usuari afeccionat que compta amb instrumentació bàsica.

Les estacions meteorològiques professionals són molt més completes, i solen venir amb possibilitats per a connectar a l'ordinador propi. Amb aquestes es sol proveir de software (propi o aliè) que realitza diverses funcions tals com: guardar un històric, crear uns gràfics per tal de publicar-los via web o el propi fet de visualitzar les dades pròpies, en temps real, al seu ordinador (gràcies a la connexió de que disposen).

Tota aquesta potència que proporciona una estació meteorològica professional, en quant a recollir dades, queda molt enrere en el moment de comunicar aquestes dades entre comunitats d'usuaris. Per tant, l'objectiu principal és cobrir aquest aspecte.

1.9 La meteorologia

No podem passar a muntar un sistema relacionat amb la meteorologia sense prèviament saber sobre quins elements estem treballant: es fa necessari definir un fons que doni

sentit a moltes de les decisions preses i això implica explicar una petita part d'aquesta ciència, sempre orientat al tema del present projecte.

Cal definir doncs la meteorologia [6] com la ciència que estudia els fenòmens meteorològics presents. De l'estudi del temps al llarg d'un època (a més llarg termini) s'encarrega una altra ciència: la climatologia [7].

És essencial definir que aquest projecte es basa en la meteorologia, per tant, el paper que juguen les dades meteorològiques actuals és la base d'aquest projecte.

1.10 Dades meteorològiques

Les dades meteorològiques són els valors que enregistrem amb un aparell de mesura, en un instant i un lloc determinats.

Parlar de dades meteorològiques en aquest projecte és equivalent a parlar d'observacions: totes venen referides al temps real.

És important destacar que no es pot dissociar una observació respecte el temps o l'espai. Per exemple, és molt habitual citar valors extrems de temperatures a l'estiu als espais meteorològics televisius i, generalment, aquests valors solen ser d'una data determinada i es representen sobre un mapa per tal que es pugui saber a quina zona ha esdevingut la

temperatura.

Hi ha diversitat de dades meteorològiques, tals com: la temperatura, la humitat, direcció del vent, força del vent, pluja acumulada, intensitat de pluja, etc. Totes elles són susceptibles de ser enregistrades i per tant interessa comunicar-les per tal d'establir comparatives i anàlisis.

1.11 Conceptes meteorològics

Per tal d'entendre aquest projecte no es requereixen coneixements de meteorologia. Sí que es fan necessaris però, certs conceptes que es veuran més endavant i que han servit per justificar certes decisions de disseny.

Un primer concepte a tenir en compte és la variabilitat del temps: depèn de quines zones del món, el temps pot ésser molt canviant en uns instants. Un exemple molt clar és el clima mediterrani [8], que a l'estiu és molt canviant: una tempesta de tarda pot aparèixer de manera sobtada i deixar quantitats d'aigua molt grans. Serveixi com a exemple un cas personal viscut en agost de l'any 2002 [9] en que van caure en només hora i mitja més de 192 litres per metre quadrat a l'àrea de Barcelona. La intensitat de pluja era tal que es va haver de buidar el pluviòmetre diverses vegades per tal de fer una medició correcta, en menys de mitja hora ja tornava a estar ple. La immediatesa de les notícies meteorològiques en certs climes és un factor decisiu.

En l'apartat de unitats relacionades amb les observacions no hi ha gaire a destacar. És de coneixement comú els graus Centígrads com a mesura de temperatura, l'hectoPascal [10] o unitat equivalent, el mil·libar, com a mesura de pressió o la velocitat del vent en km/h.

En aquest present projecte s'usa una mesura pressió normalitzada, això és coneix com pressió reduïda a nivell de mar o, en anglès, MSLP [11]. El que vol dir això és que la pressió atmosfèrica és més baixa conforme més alts estem: a 1500 metres la pressió és molt menor que a nivell de mar. Per aquest motiu el que es fa és extrapolar la pressió que hi ha en un punt més elevat com si estigués a nivell de mar, de manera que es pot comparar més fàcilment els valors de pressió.

1.12 Nivell amateur: la xarxa

1.12.1 Fòrums d'Internet

Conscients de la necessitat de comunicar dades meteorològiques, Internet ha estat un centre d'on s'ha partit per tal de crear comunitats senceres sobre les que els usuaris han escrit les seves observacions meteorològiques. Es parteix de la base que cada usuari compta amb una connexió a Internet més la instrumentació que posseeixi per obtenir dades meteorològiques.

En un principi el més habitual han estat els fòrums, que encara perduren a dia d'avui. Fòrums com Catmeteo [12] o Meteored [13] tenen moltíssimes dades del temps real

aportades pels diversos usuaris. Part d'aquestes observacions són manuals, preses a simple vista i complementades amb fotografies que il·lustren la situació.

1.12.2 Webs personals

Hi han usuaris que poden permetre's el fet de comprar un espai web on poden publicar les seves observacions de manera contínua. Això requereix una infraestructura extra més enllà de la clàssica connexió d'Internet: passa a ser un pagament mensual o anual per a poder tenir un espai web online, afegit al cost de la connexió a Internet pròpia de cada usuari.

Aquest tipus d'usuari sol ser propietari d'una estació meteorològica professional que pren dades de manera contínua i, amb un software propietari de l'estació, les dades són rebudes per un ordinador connectat a la estació i a Internet. El mateix ordinador s'encarrega de publicar les dades en forma textual o gràfica en el site web que posseeixi l'usuari.

1.13 Nivell professional

Les entitats meteorològiques solen estar relacionades amb el govern, d'on treuen el finançament corresponent per a poder mantenir una infraestructura potent. Això els permet instal·lar estacions meteorològiques automàtiques a més d'una xarxa d'observadors que proveeix amb observacions de manera contínua.

A nivell professional la quantitat de dades observades són molt més diverses, amb un equipament que dóna molta exactitud en cadascuna de les mesures i amb una freqüència molt elevada. Cal tenir en compte que totes les dades observades són recollides i processades per un centre de manera que, més tard, es pugui establir prediccions meteorològiques a curt o mig termini.

No totes les entitats són de caire governamental o públic, també n'hi ha de privades, però tot i així operen d'una manera similar: aporten dades a un centre on es processaran i es publicaran una sèrie de resultats.

1.14 El model centralitzat

Ja sigui de manera amateur, sense infraestructura o bé amb la infraestructura que poden comptar des de una entitat amb fons (pública o privada), el model de comunicació que presenten és un model per on les observacions passen per un sol punt: ja sigui per ser consultades per diversos usuaris, o per ser processades per a un posterior ús en prediccions, les dades es concentren en un lloc. A la següent figura s'expressa aquest esquema:

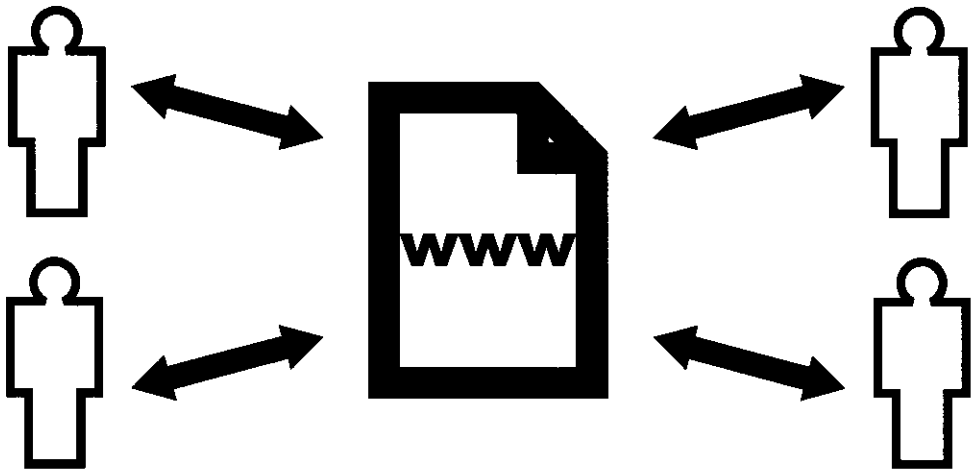


FIGURA 5A

El resultat de les diverses formes en que els usuaris comparteixen dades meteorològiques segueix un model centralitzat. Es parla d'un model clàssic de client/servidor, on el client consulta o afegeix observacions a un sol servidor, que les emmagatzema.

1.15 Motivació pel canvi de model

El model centralitzat, vist en l'anterior capítol, suposa una sèrie d'avantatges però també compta amb una sèrie d'inconvenients. Cal tenir en compte que és un model que ha sorgit d'aquesta manera degut a la naturalesa de la xarxa: en el cas amateur pel fet de ser un lloc centralitzat de discussió o en el cas professional per la necessitat de concentrar totes els dades en un mateix punt per tal de simplificar infraestructura i unificar processos de càlcul.

1.15.1 Desavantatges del model centralitzat

1.15.1.1 Observacions centralitzades

Si es concentren les observacions en un sol punt es corre el risc de poder no afegir una nova observació o consultar les existents en cas de fallada. Seguint la figura 5A, és el cas en que la web pot quedar sense servei degut a problemes tècnics tant de hardware com de software en el node central o en la xarxa d'accés a la mateixa.

El mateix pot passar dins d'una xarxa professional d'observadors meteorològics a on les dades meteorològiques aportades es concentren en un sol punt.

1.15.1.2 Cost de la infraestructura

Partint de la base que la connexió bàsica a Internet avui dia té un cost adquisitiu determinat, el fet d'adquirir un espai en Internet, ja sigui una web o un fòrum, suposa un sobre cost, no només econòmic sinó també de manteniment.

Si s'opta des del punt de vista professional o d'organitzacions la despesa econòmica suposa un cost molt major, més enllà de l'abast de moltes persones.

1.15.1.3 Desplegament de les dades

A nivell amateur, les dades aportades pels usuaris a les webs o els fòrums solen ser en temps real on es publiquen tant aviat l'usuari envia aquestes observacions. A nivell

professional això no sol passar, degut a que les dades s'han de sotmetre a una validació per part de l'ens que les gestiona prèvia publicació en el medi corresponent.

1.16 El P2P

Es pot definir com a P2P [14] un model de connectivitat on no hi ha servidors centralitzats, sinó els mínims necessaris per al funcionament bàsic de la xarxa.

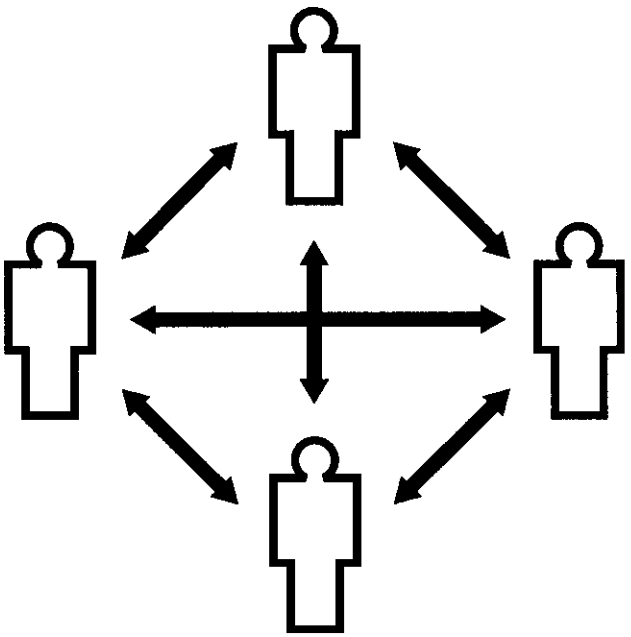


FIGURA 6A

A la anterior figura (6A) es pot apreciar l'esquema indicat: cada node es posa en contacte amb la resta de nodes sense passar per un servei centralitzat. Això també presenta una sèrie de avantatges i inconvenients que exposarem a continuació.

Aquest mateix model de connexió és un model que ja ha estat adoptat per software divers, sobretot el que està relacionat amb l'intercanvi de fitxers. Exemples ben clars de l'ús d'aquesta tecnologia són: Emule, BitTorrent o DirectConnect. Cadascun d'aquests programes suporta un sistema de xarxa determinat, és a dir, un esquema de connexions concret que pot diferir al mostrat en la figura 6A, però que es basa en aquesta estructura.

En aquest tipus d'arquitectura ja no es parla de clients i servidors, sinó que es parla de nodes.

1.16.1 Avantatges i inconvenients

1.16.1.1 Avantatges

Es pot comptar com a avantatge que no es requereix una infraestructura potent per a construir una xarxa P2P, no cal tenir serveis contractats més enllà d'una connexió bàsica per a poder gaudir de la potència que brinden les aplicacions basades en aquest model.

També cal tenir en compte que el fet que no es centralitzi la informació suposa una major robustesa en la comunicació de dades, donat que la informació pot viatjar per diferents camins.

1.16.1.2 Desavantatges

En una xarxa descentralitzada es dóna el problema d'arrencada, és a dir: el lloc d'entrada

per on hem de connectar-nos a la xarxa. En el model centralitzat hi ha un sol punt on hem d'anar a buscar la informació: una web, un fòrum, etc... En aquest model hauríem de tenir algun sistema per saber per quin punt podem entrar per tal de formar part d'aquesta xarxa.

També hi ha la problemàtica de la incoherència (en dades o en la manera en que està organitzada la xarxa). Això no passa en el model centralitzat donat que tots les dades passen pel mateix punt i són distribuïdes idènticament a cadascun dels usuaris.

1.17 Un model híbrid

El model híbrid combina les possibilitats dels dos models cercant els avantatges en tots dos casos. Algunes aplicacions prèviament mencionades en el P2P no són purament descentralitzades sinó que estan basades en models híbrids: és en aquesta línia on s'ha decidit incidir en el projecte donat que aquestes aplicacions s'han mostrat molt efectives en els àmbits on han estat aplicades i poden resultar favorables en aquest cas.

El model híbrid és el model que s'ha aplicat en aquest projecte.

La manera en que es combinaran tots dos models fins a formar la proposta del present projecte és el que es discutirà a continuació.

1.18 Bibliografia i referències web

1: Article sobre la predicció numèrica del temps [Wikipedia]

http://en.wikipedia.org/wiki/Numerical_weather_prediction

2: Article sobre la metodologia RUP [Wikipedia]

http://en.wikipedia.org/wiki/RUP

3: Article sobre el paradigma de desenvolupament iteratiu [Wikipedia]

http://en.wikipedia.org/wiki/Iterative_and_incremental_development

4: Article sobre el paradigma de desenvolupament en cascada [Wikipedia]

http://en.wikipedia.org/wiki/Waterfall_model

5: Article sobre el paradigma de desenvolupament en espiral [Wikipedia]

http://en.wikipedia.org/wiki/Spiral_model

6: Definició de meteorologia [Wikipedia]

http://en.wikipedia.org/wiki/Meteorology

7: Definició de climatologia [Wikipedia]

http://en.wikipedia.org/wiki/Climatology

8: Article sobre el clima mediterrani [Wikipedia]

http://en.wikipedia.org/wiki/Mediterranean_climate

9: Entrada de blog sobre episodis de pluges d'Agost [Comando Tibidabo]

http://www.comandotibidabo.com/index.php/arxiu-resums/63

10: Definició de la unitat de pressió Pascal i els seus equivalents [Wikipedia]

http://en.wikipedia.org/wiki/Pascal_(unit)

11: Definició de "Pressió reduïda a nivell de mar" [Wikipedia]

http://en.wikipedia.org/wiki/Mean_Sea_Level_Pressure

12: Fòrum de meteorologia Catmeteo [Catmeteo]

http://forum.catmeteo.com/

13: Fòrum de meteorologia Meteored [Meteored]

http://foro.meteored.com/

14: Article sobre el "Peer to peer" [Wikipedia]

http://en.wikipedia.org/wiki/Peer-to-peer

2 Disseny del sistema

2.1 Disseny inicial

Previ al present treball, es va dissenyar un altre sistema del qual ha derivat el projecte "Sistema Meteorològic Distribuït".

Aquell disseny inicial contemplava un model de xarxa totalment P2P on cada integrant, d'ara en endavant node, tenia un únic software que s'encarregava de:

1. Obtenir dades de la estació meteorològica pròpia, si s'en disposa.
2. Manipular les connexions amb els altres nodes.
3. Exposar les dades rebudes en un mapa.

Això té l'avantatge que s'unifica en un sol software totes les funcionalitats de la xarxa. És el cas que ens podem trobar quan usem programes d'intercanvi de fitxers com el Emule: tota la gestió la fem des d'una mateixa interfície.

A a figura 7A podem observar una possible topologia per a la xarxa en la seva forma de P2P. Cada node consta de la xarxa consta, com a mínim, d'una pantalla sobre la que podem visualitzar les dades i, en alguns casos, alguns nodes disposaran d'elements de mesura per tal de transmetre les observacions a la xarxa. Aquells nodes que només poden veure les observacions estan simbolitzats amb una pantalla que conté un núvol, en canvi els que sí poden enviar les observacions estan simbolitzats amb un termòmetre. En el dibuix es pot observar com les connexions són punt a punt entre els diversos nodes: els

nodes que poden enviar observacions estan connectats a la resta, en canvi els nodes que només poden veure aquestes observacions no estan connectats amb els nodes del mateix tipus.

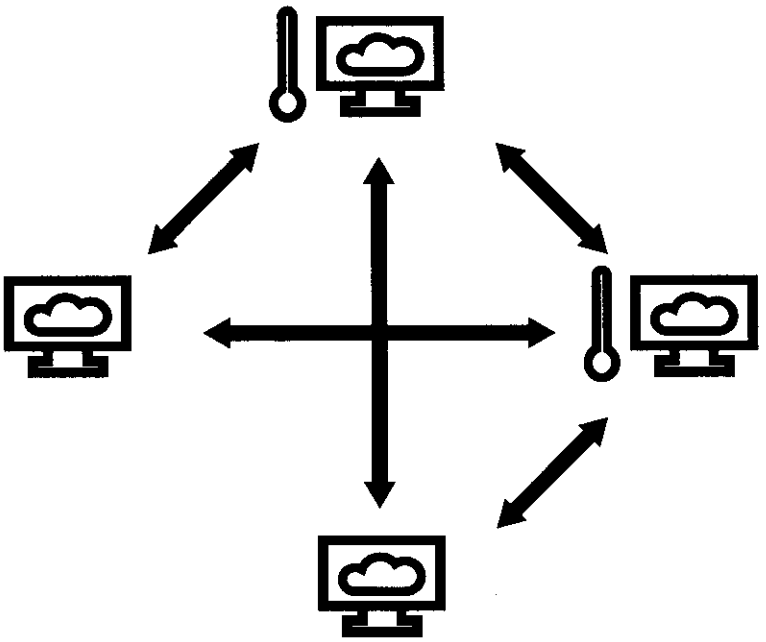


FIGURA 7A

2.1.1 Avantatges del model

Entre els avantatges d'aquest model de comunicació d'observacions es pot observar com es compleix la descentralització de les dades: les dades no es concentren en cap punt sinó que viatgen de manera independent cap a la resta de nodes de la xarxa.

També es dóna el fet que la comunicació de dades és directa, és a dir, no passa per nodes intermedis sinó que va directament de l'observador a la resta d'usuaris: això fa que no hi hagi retards significatius des que es produeix una observació fins que els usuaris la

veuen.

2.1.2 Desavantatges del model

Tot i els avantatges de la descentralització, aquest model no és escalable. Si per cada client hem d'obrir una connexió per enviar les observacions això pot significar un total de connexions molt considerable tan aviat el nombre d'usuaris augmenti. Suposem un miler d'usuaris, el cost dels recursos de les connexions és molt alt per al típic sistema operatiu. Fins i tot, determinats sistemes operatius no permeten obrir moltes connexions en poc segons per restriccions de seguretat [1].

Cal tenir en compte que l'ample de banda en un sol node és molt elevat: si cada node obre tantes connexions com usuaris aleshores localment, en cada node que envii informació, l'ample de banda requerit per enviar una sola observació a tots els nodes serà tant elevat com el nombre d'usuaris. Si fem uns càlculs ràpids, suposem 1024 bytes per informació a 1000 nodes, aleshores tenim 1024000 bytes que en termes de xarxa seran 8192000 bits. Si la connexió habitual d'ADSL té un ample de banda de pujada de 320000 bits per segon això implica diversos segons per enviar la informació a la resta de nodes, sense que l'usuari pugui usar la seva connexió d'Internet per a qualsevol altre ús. Cal recordar, d'altra banda que les observacions són enviades en un interval de segons, pel que podria passar que, tot just mentre enviem una observació haguéssim d'enviar la següent.

El problema de la arrencada continua existint: haurem de fixar un node en tota la xarxa, és a dir publicar la seva adreça, per tal que els altres es connectin allà i poder construir la xarxa a partir d'aquell punt.

2.2 Disseny modular

Per tal de millorar els possibles desavantatges del disseny inicial s'ha pensat en una millora fent servir l'esquema híbrid que es va introduir en el apartat 1.19 d'aquest mateix document.

Quan abans es parlava de un sol node, s'especificava que aquest complia tres funcions: obtenir les observacions, controlar les connexions d'ell amb els altres nodes i mostrar les observacions rebudes.

El plantejament d'aquest nou disseny és basa, precisament, en dividir les funcionalitats d'aquest node en tres aspectes diferents. Separem les tres funcionalitats ens programes diferents de tal manera que ara l'usuari només haurà d'executar allò que necessiti. Un usuari que només vulgui servir dades de la seva estació a la resta d'usuaris només caldrà que usi un sol programa, o aquell usuari que només vulgui visualitzar les dades aportades per altres usuaris també arrencarà un sol programa. Cada programa serà més petit i només contindrà les funcionalitats necessàries per a l'usuari.

La estimació en una xarxa d'aquest tipus és que el nombre d'usuaris que enviïn informació

sigui relativament baix respecte el nombre d'usuaris que visualitzaran les observacions: no més d'un 20% dels usuaris totals degut al cost econòmic que suposa tenir una estació meteorològica amb connexió a l'ordinador. Per a l'usuari que només vol tenir un programa que reculli les observacions les 24 hores del dia és molt més còmode si el mateix programa conté les parts essencials per a realitzar aquesta tasca. Per a l'usuari que no tingui cap estació de on recollir observacions i en canvi si les vulgui visualitzar, usarà un sol programa que reculli les observacions de altres usuaris i permeti visualitzar-les.

La solució en que s'ha basat aquest projecte continua la direcció apuntada en aquest apartat: s'ha optat per separar un únic software monolític en tres mòduls diferents.

Cal ressaltar que serà totalment compatible el fet de usar més d'un programa en la mateixa màquina: el fet que hagi un programa que reculli observacions meteorològiques no priva que pugui coexistir amb un altre programa que mostri aquestes dades, i la de altres usuaris, per pantalla.

Manca per resoldre una tercera funcionalitat que fins ara no hem esmentat i que és essencial: la manera en que es connectaran els nodes entre ells per tal de formar la xarxa. Aquesta funcionalitat també està separada de les altres dues i s'explicarà en els següents apartats.

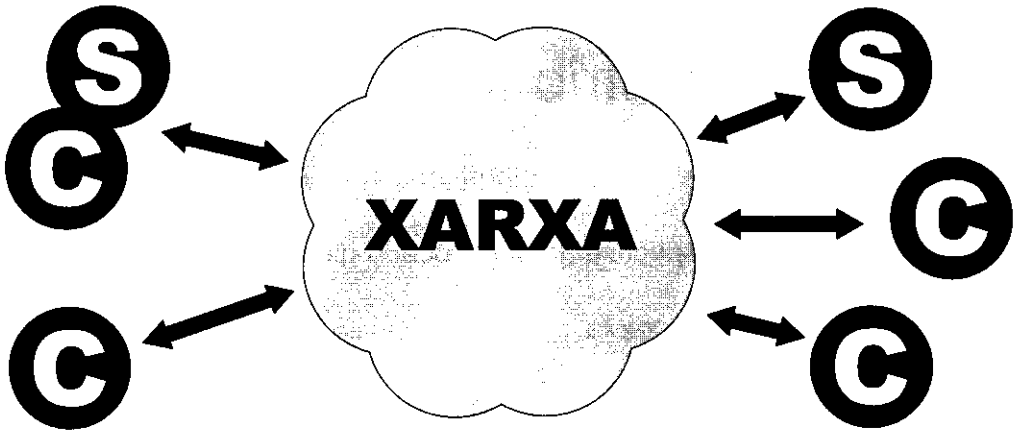


FIGURA 7B

A la figura 7B podem observar com quedaria una xarxa de nodes segons el disseny que s'ha ideat. El tipus de nodes corresponen a:

S : Si executa el programa que rep dades de la estació i les envia a la xarxa.

C : Si executa el programa que pren dades de la xarxa i les mostra per pantalla.

2.3 El control de la xarxa

Tal i com s'ha vist en el punt 2.1 hi ha un problema associat a l'ús del P2P de manera directa aplicat en aquest cas: l'ample de banda local és molt gran per a un determinat node de la xarxa. Cal tenir un major control de la xarxa per tal d'evitar aquests fets.

Aquest problema s'ha adreçat en aquest present projecte fent ús de un model híbrid, com es veurà a continuació, de tal manera que la xarxa tingui un sistema que permeti organitzar-se per tal d'evitar aquestes situacions.

2.3.1 El control distribuït

Quan parlem de control de la xarxa distribuït ens referim a que cada node quan es connecta a la xarxa té una perspectiva (ja sigui local o global) de la manera en que estan connectats la resta de nodes a la xarxa.

Si la perspectiva és local pot passar que no s'optimitzi de manera correcta les connexions per tal de minimitzar l'ample de banda disponible: en tenir un punt de vista parcial, no es pot optimitzar tota la xarxa de manera global.

Si la perspectiva és global, la quantitat d'informació que ha de viatjar per la xarxa informant respecte la seva topologia és molt gran: sí que tindrem una visió sencera de la xarxa i de com organitzar els nodes però al tràfic de dades cal sumar-li el tràfic que comporta la modificació de la topologia de xarxa (cada cop que es connecti un nou usuari la topologia de la xarxa canvia).

2.3.2 El control centralitzat: el node enllaç

En canvi podem establir un determinat node que s'encarregui de ordenar cadascun dels nodes que entri a la xarxa: aquest s'encarregarà d'assignar les connexions del nou node.

Aquest és el mètode que s'ha optat pel present projecte: un node que anomenarem enllaç

i que s'encarregarà de distribuir a la resta de nodes per la xarxa.

Aquest node d'enllaç serà un tercer programa complementari als altres dos, de fet, és el que cobreix el tercer aspecte que s'ha destacat en el disseny inicial (apartat 2.1). Com els altres dos programes, aquest programa també pot ser executat en un mateix node conjuntament amb els altres dos.

En un principi, i per motius de simplicitat, dissenyarem el Sistema Meteorològic Distribuït amb un sol node enllaç per a tota la xarxa, en realitat no té perquè limitar-se a un sol node d'enllaç. La següent figura (7C) mostra l'esquema resultant d'aquesta decisió:

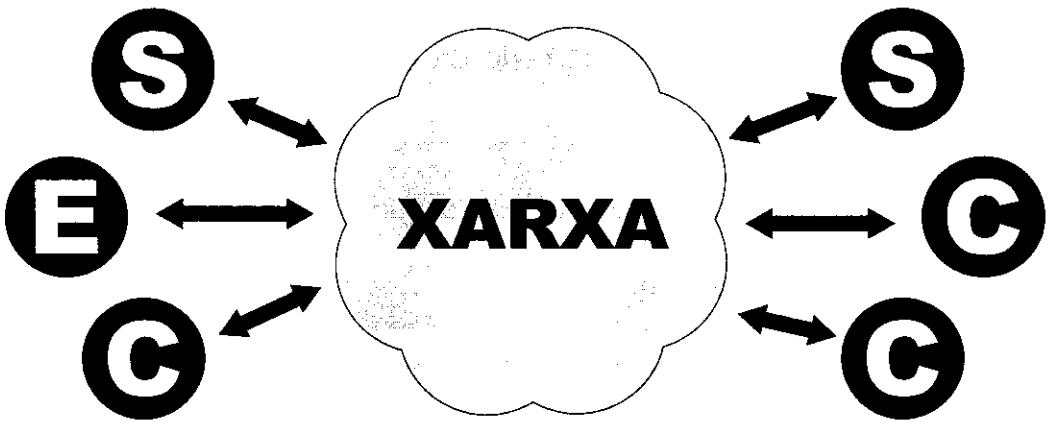


FIGURA 7C

Seguint la figura 7C el node d'enllaç es simbolitza amb una E. En la figura es pot observar com hi ha un sol node d'enllaç (E) respecte els diversos nodes que aporten dades (S) i que els reben i les visualitzen (C).

Arribats a aquest punt es continua tenint un problema sense resoldre: com rebaixar l'ample de banda usar de manera que localment sigui reduït. Per a això cal veure com aquest node d'enllaç ajuda a resoldre aquesta situació, és a dir, es desvetllarà el que conté allò que en les anteriors figures s'ha representat amb un núvol amb la paraula XARXA.

2.4 Disseny intern de la xarxa

Per tal fixar unes bases sobre el que parlarem a continuació definim els següents aspectes.

1. Un node Estació o S és aquell node de xarxa que executa el programa que absorbeix dades de la estació per a llençar-les a la xarxa.
2. Un Client o C és aquell node de xarxa que executa el programa que recull aquestes dades i les visualitza
3. Un node de xarxa Enllaç o E és qui executa el programa que assigna les connexions entre els nodes S i C.
4. El fet que un mateix node executi un, dos o tots tres programes no representa cap distinció pel sistema: es tracten com si fossin nodes separats, és a dir, no han de tenir forçosament cap relació.

2.4.1 Condicions de disseny

Es vol realitzar un disseny on s'usi el menor ample de banda possible tot i l'increment del nombre de nodes.

Quan s'ha pensat l'esquema de connexions de xarxa que s'explicarà a continuació s'ha tingut en ment que un xarxa pot contenir fins a una desena de milers de nodes.

2.4.2 Connexió dels nodes S

Els nodes S envien informació a la xarxa, per tant, són els susceptibles de ser millorats amb un millor esquema de connexions. En l'esquema P2P es va poder comprovar com no era viable que un node d'aquest tipus enviés informació a la resta de nodes de la xarxa; menys encara quan a la xarxa hi pot haver milers de nodes.

Per millorar aquest aspecte s'ha pensat en el fet que la estació, no té perquè notificar cada observació a tots els nodes que hi hagi a la xarxa. Es pot fer que la estació enviï la informació a un sol node de tipus C i que aquest s'encarregui de llençar-la a la resta de nodes. Això ens resultarà en un esquema com el que segueix:

Les connexions de la figura que estan representades amb fletxes unides amb línia contínua són connexions persistents: estan connectades contínuament mentre el node estigui en actiu. Les connexions no persistents es simbolitzen amb una línia amb fletxes discontínues.

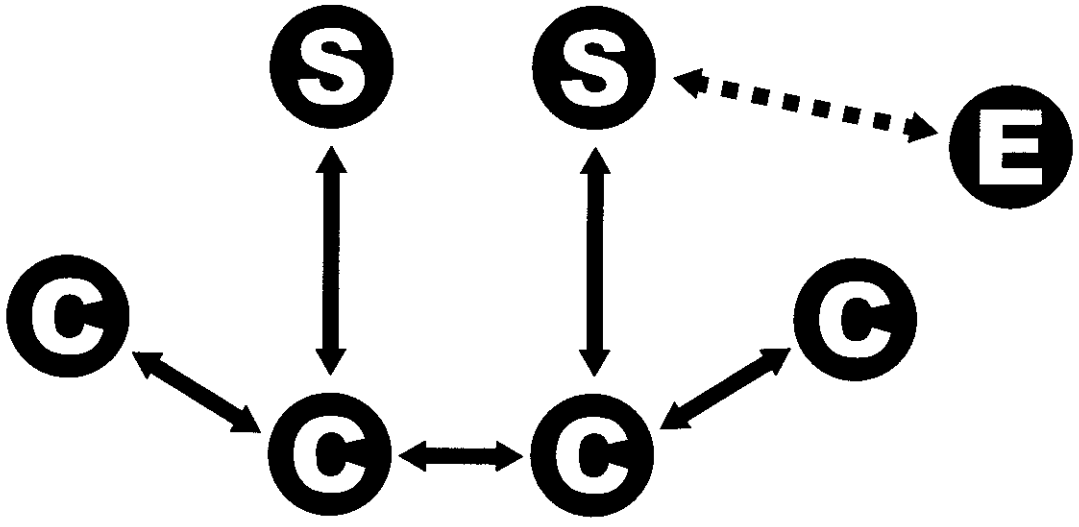


FIGURA 7D

En la figura 7D es pot observar com les estacions (S) tenen una sola connexió amb un altre client (C), mentre que els clients tenen connexions amb altres clients. Les estacions no s'envien dades entre elles, totes van a parar als clients, que enviaran aquestes als seus veïns, és a dir, als nodes clients als que estan connectats.

El problema que pot esdevenir en aquest tipus de connexions és que hi hagi un bucle de connexions entre clients. Això podria resultar en que una observació enviada a un client acabi fent una volta per tots els clients fins que la rebi un que ja ha reenviat aquesta informació. Però d'això s'encarregarà el node enllaç (E) que organitzarà les connexions de tal manera que això no passi.

2.4.3 Connexions dels nodes C

S'ha traslladat el problema de l'ample de banda acumulat a cada node S cap als clients, a qui enviem directament les observacions de manera que entre ells se les han d'enviar. Això suposa dos problemes:

1. La velocitat d'expansió de les dades.
2. L'ample de banda usat.

¿A què es refereix la velocitat d'expansió de les dades? Aquesta ve referida al temps que transcorre entre l'instant que la dada és enviada per un node S, i l'instant en que arriba a l'últim node C.

Si es planteja el fet que cada node C, té un altre node C a qui envia les dades i així successivament fins que arriba al final (el que equivaldria a una cadena), podem establir uns càlculs bàsics que permetin veure quina velocitat d'expansió de les dades hi ha en aquest cas. La figura 7E respon a aquest esquema:

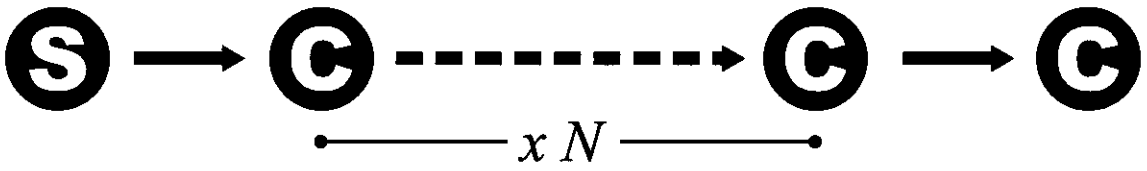


FIGURA 7E

Per a l'objectiu que volem assolir, assumim que N = 9000. Independentment de l'ample de

banda disponible que es suposarà infinit, assumim que la velocitat a la que es transmet una observació entre un node i el següent és de l'ordre de 500 ms. A cada node, tindrem un temps de procés que suposarem de 500 ms. Aquest temps de procés és el temps que triga el programa en notificar la observació de la xarxa, emmagatzemar-la per a la seva posterior visualització i llençar-la a la xarxa de nou cap el següent node.

Així doncs tenim un total 1s per node, multiplicat pels 9000 nodes C, requerim un total de 9000s, és a dir, 150 minuts o el que és el mateix: gairebé 2 hores i mitja.

La velocitat d'expansió de les dades no hauria de suposar un problema habitualment, donat que les temperatures no solen canviar molt ràpidament, ni passa el mateix amb la humitat o la pressió. Però dos hores i mitja no són factibles, menys encara si tenim en compte per posar un exemple, del clima mediterrani que és un clima molt canviant. Requerim un esquema de xarxa una mica més complex que ens permeti ajustar aquest temps fins a rebaixar-lo tant com sigui possible, però a la vegada s'ha de tenir un compromís amb l'ample de banda.

2.4.3.1 El concepte d'associativitat

En veure que una connexió en cadena suposa uns temps d'expansió de dades molt grans, el que es podria fer és que cada node C és connectés a més d'un node C. D'aquesta manera en lloc de passar una observació de un node C a un altre node C, ara passaria a ser d'un sol node C a altres nodes C.

Ara bé, cal tenir molt present que el nombre de nodes als que enviarem la observació no poden ser molt grans perquè si això passés, s'arribarà un altre cop al problema inicial que va esdevenir amb el model P2P: molts nodes connectats suposen un problema d'ample de banda. S'ha d'arribar a un equilibri entre velocitat d'expansió i ample de banda.

Al nombre de nodes C connectats a un sol node C és el que anomenarem associativitat.

Mitjançant el valor de l'associativitat es podrà fixar el balanç més adient entre velocitat d'expansió i ample de banda en funció de la xarxa sobre la que s'estigui treballant.

Suposem una associativitat de 3: cada node C té tres nodes C connectats a ell. De manera que quan arriba una observació per un veí, el node C que la rep la reenvia pels altres dos nodes que resten connectats a ell. La figura 7F resumeix aquesta situació:

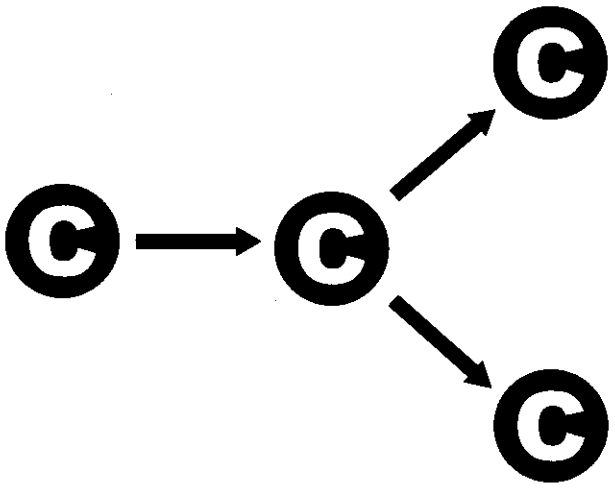


FIGURA 7F

Es pot apreciar com el model de connexió és un arbre, per tant cal tenir molta cura en la seva construcció per tal que aquest quedi ben equilibrat, és a dir, s'ha de tendir a que cada node tingui la màxima associativitat permesa.

A continuació uns càlculs aproximats seguint la figura 7F tal i com s'ha fet en el cas anterior. Posem per cas que hi ha 9000 clients amb una associativitat de 3 i que està completament equilibrat, és a dir, que la major part de nodes C té 3 nodes C connectats. Això implica que a cada pas que la observació recorri l'arbre de connexions es duplicarà el nombre de clients que la llegeixin: aquest creixement és exponencial a on la base és la associativitat - 1.

Si tenim en compte les mateixes dades sobre els retards que en l'anterior càlcul, és a dir, 500 ms de temps de transmissió de dades entre dos nodes C qualssevol i 500 ms de temps de procés de la observació en un node C. La observació entre els seus extrems viatjarà per un total de $\log_2 9000 = 13$ nodes. Això implica 13 segons de temps de viatge entre un punt i un altre: és una millora d'un 76000 % amb un petit canvi.

En el present projecte no s'ha definit un nivell d'associativitat definit, però es considera que un nivell de 3 o 4 són suficients per a grups de 10000 usuaris.

Cal tenir en compte que la organització dels nodes queda reservada exclusivament a la

feina del node d'enllaç, que s'encarregarà d'assignar les connexions entre els nodes client de manera que s'aprofiti de la millor manera la associativitat de la xarxa per tal d'optimitzar ample de banda i velocitat d'expansió.

A continuació un exemple d'un cas complet a petita escala amb associativitat 3:

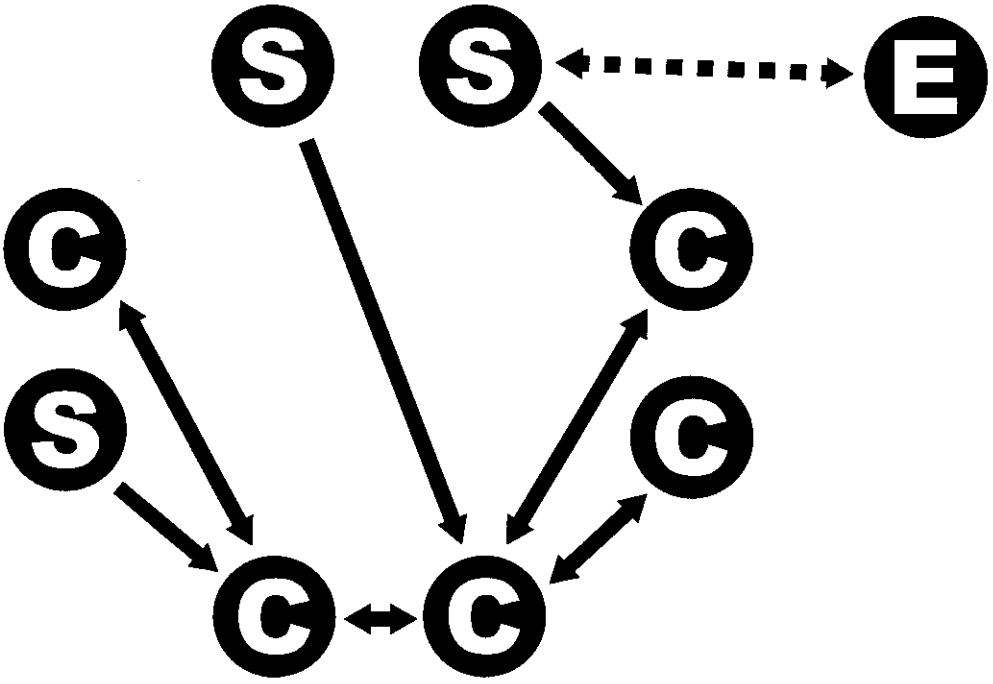


FIGURA 7G

La figura 7G mostra un esquema de connexió plausible amb les condicions indicades prèviament. Com en els anteriors esquemes, les fletxes contínues indiquen connexions permanents mentre que les connexions discontínues indiquen connexions esporàdiques.

En el cas que ens ocupa s'està mostrant l'enllaç com uneix un node S amb la resta de

nodes. Els nodes C d'avall presenten 2 i 3 connexions, mentre que la resta de nodes C tenen una sola. La figura 7G intenta reflectir el que pot passar en la realitat: tot i equilibrar de la millor manera possible un arbre pot esdevenir que els nodes que s'han connectat inicialment desapareguin, de tal manera que la xarxa es dissociï i quedi en un estat no òptim. En aquest cas es mostra un cas pitjor, en el que una observació d'un node S pot travessar 3 nodes fins arribar a l'últim node C.

Els detalls referents al disseny relacionats amb l'arquitectura de cada node es discutiran en els següents capítols.

2.5 Bibliografia i referències web

1: Limitacions d'ús de sockets en Windows XP SP2 en endavant

http://www.speedguide.net/read_articles.php?id=1497

3 Disseny dels nodes

3.1 Introducció

Un cop s'ha vist la manera en que s'ha dissenyat el sistema de manera global i el seu comportament, ara cal definir de quina manera estructurarem els components que integren aquest sistema.

En el següents apartats es descriu la manera en que s'organitzen de manera interna cada node separant en tot cas el comportament en dependència de la funció de realitzen.

Es torna a incidir en el següent aspecte: quan parlem d'un node estem parlant d'un dels tres programes que componen el Sistema meteorològic distribuït independentment que estigui executant-se tots tres en un sol context (un mateix ordinador), o distribuïts al llarg de tota la xarxa (en diversos ordinadors connectats a Internet).

3.2 Perspectives de disseny

El disseny de software cada node hi ha diverses perspectives per tal de desenvolupar aquest software. A continuació es descriuen les perspectives que s'han barallat i la que s'ha decidit.

3.2.1 Perspectiva de xarxa

Si es miren els nodes des de la perspectiva de xarxa, el disseny software de cada node

s'orienta a adaptar-se de la millor manera a la topologia que s'ha decidit (capítol 2).

La perspectiva de xarxa obliga a tenir en compte l'asincronia amb que rebem dades de la xarxa, la qual cosa obliga tenir processos o threads que estiguin escoltant la xarxa contínuament a la espera de dades que s'enviaran entre els diversos nodes. Es reforça molt més el disseny al voltant dels threads que no pas al voltant de l'arquitectura.

Si dividim el software en un model de tres capes, els threads hauran de ser transversals a les tres capes, és a dir, les creuaran donada la interacció de cada node. Això passa degut al fet que cada node usa les dades per a un objectiu diferent: mentre que un node S les observacions estaran en la capa de negoci, en un node C aquestes observacions estaran en la capa de negoci i en la de presentació (cal recordar que aquest node mostrarà dades per pantalla, per tant sí que és necessari que el thread interactuï amb la capa de presentació). La següent figura (8A) mostra aquest fet:

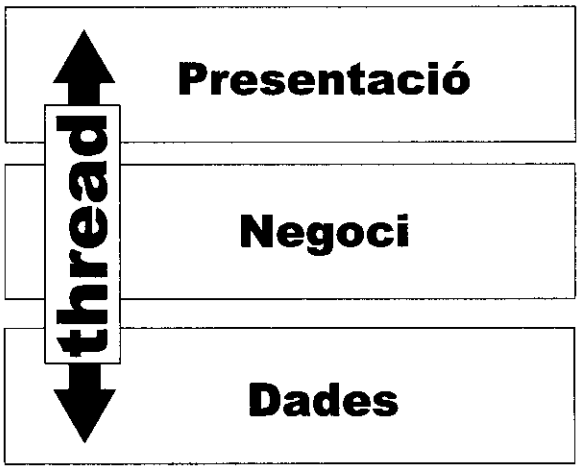


FIGURA 8A

El fet que el thread travessi les capes complica el desenvolupament del software sumat al fet que es farà una mica més difícil el reaprofitament dels mòduls entre diversos tipus de nodes degut a la característica heterogènia del model: cada thread en un determinat tipus de node tindrà un comportament completament diferent.

No obstant no es descarta l'ús dels threads donat que per les comunicacions de xarxa són gairebé imprescindibles.

3.2.2 El model pipeline

El model pipeline està basat en la manera en que un fàbrica elabora un producte de manufactura: el producte es munta i es distribueix seguint una cadena de producció.

Aquesta idea esdevé del fet que la unitat bàsica de tot el Sistema meteorològic distribuït són les observacions. Es basa en el fet que les observacions viatgen al llarg de la xarxa: parteixen dels nodes S, que les generen, i passen per tot un seguit de nodes C que les llegeixen i les passen als seus veïns. L'esquema quedaria com indica la figura 8B:

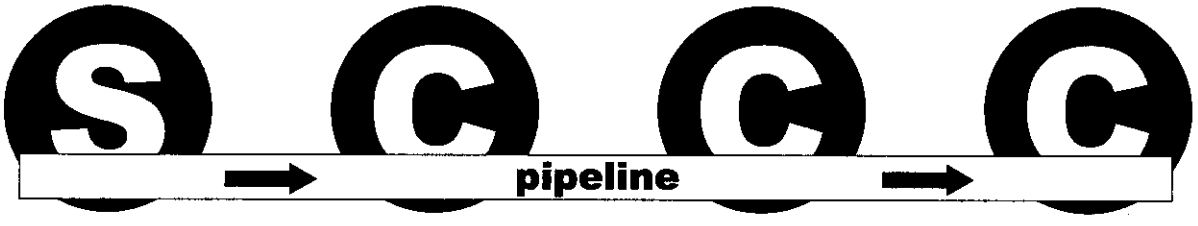


FIGURA 8B

Es pot veure com el pipeline travessa longitudinalment tots els nodes i dura tant com nodes travessi la observació. La explicació de com funcionaria aquest pipeline la cobrim en l'apartat immediatament següent.

Els inconvenients que presenta aquest model és que cal delimitar el pipeline de manera que només travessi una sola capa (la capa de negoci) sense limitar la funcionalitat del sistema: tal i com ens passa amb el threads.

3.2.2.1 Funcionament

Quan es genera aquesta observació, al node S, aquesta es construeix en una estructura de dades i s'insereix dins d'un paquet de dades a la xarxa. Un cop acabada la elaboració del paquet l'enviem a la xarxa.

Un node C rep un paquet de dades d'un altre node, ja sigui de tipus S o C. Un cop rep el paquet envia una còpia del mateix als seus veïns i amb la seva pròpia còpia realitza les tasques assignades al node C, que són les de la visualització de les dades. Per tant transforma el paquet en dades fàcilment interpretables per l'humà i les envia al sistema de visualització que tinguem: ja sigui un fitxer de text, o un mapa amb símbols, etc.

3.3 Disseny dels nodes

Com en altres solucions, l'esquema que s'ha adoptat és un model híbrid: una mescla dels

dos anteriors models.

No es poden tractar connexions sense threads. Si per cada connexió que establim hem d'estar contínuament mirant quina ens aporta dades (consulta per enquesta) i després llençar-les a les altres connexions el que estarem fent és ocupar el processador amb consultes inútils. Per tant crear diversos threads, tants com connexions, que estiguin a l'aguait de les observacions que arriben és una necessitat: aquests threads no consumiran temps de processador fins que alguna dada els arribi per tant serà molt més òptim que un sol thread que estigui consultant connexions contínuament.

Considerant un software sobre una arquitectura de tres capes, usarem el model pipeline dins d'una sola capa, la de negoci. Si es considera que s'han de muntar llibreries per tal de muntar i desmuntar paquets de dades amb observacions, llençar-los a la xarxa i llegir-los de la mateixa, amb un model de pipeline es pot veure com es compartiran molts dels components entre els diversos tipus de nodes. És més fàcil adaptar-los a una sola línia per on passen els paquets que adaptar-los en funció del thread i del node que els llegeixi.

Com es combinaran els threads amb el model pipeline? Es limitarà cada thread a la capa de negoci on cadascun d'aquests només interactuarà amb el pipeline. La capa de negoci serà la que envii les dades o els esdeveniments cap a les altres capes corresponents. D'aquesta manera tot el codi que estigui relacionat amb el pipeline es pot reaprofitar donat que tindrà un comportament molt similar en tots dos tipus de nodes (nodes S i nodes C), mentre que el codi específic de cada node s'afegirà a les capes corresponents: el que

manqui de la capa de negoci més el corresponent a la capa de presentació o de dades.

La següent figura recull aquesta idea:

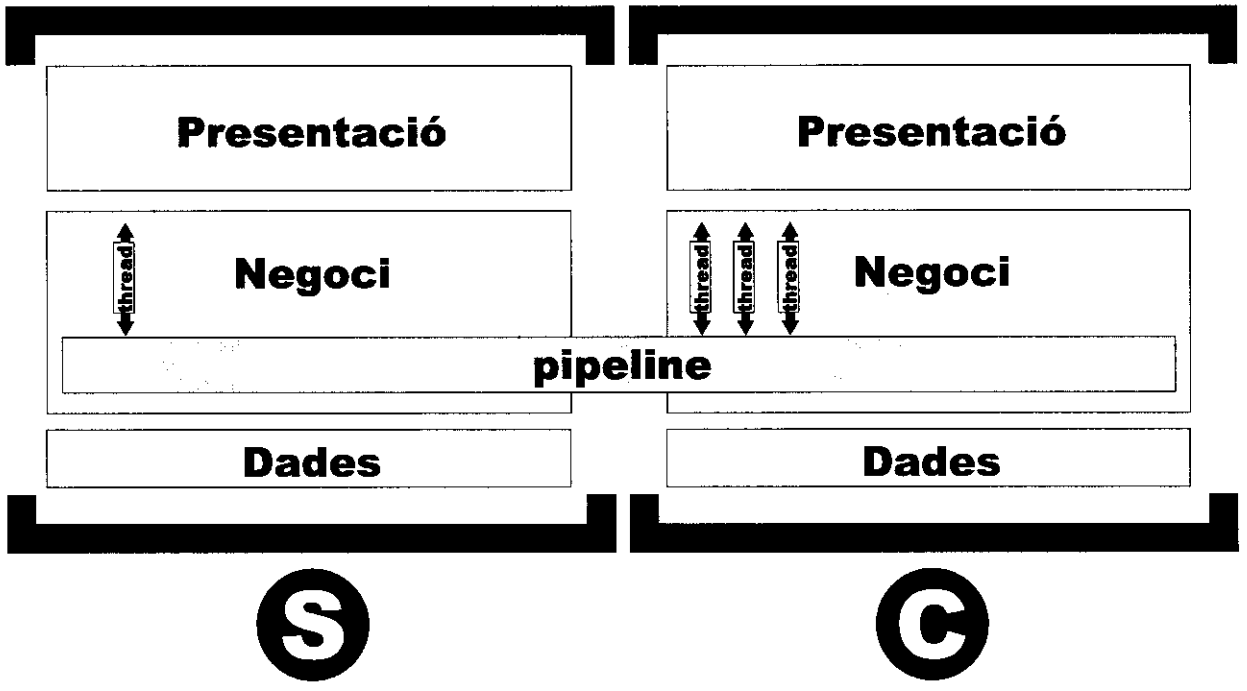


FIGURA 8C

3.3.1 Disseny del node Enllaç

El node enllaç no entra dintre del disseny que s'ha realitzat pels nodes tipus S i C donat que en cap cas aquest node mourà observacions per la xarxa: la tasca d'aquest node és únicament la de organitzar la resta de nodes de la xarxa per tant el seu disseny és diferent.

En aquest node s'ha simplificat molt la feina donat que, a excepció que es vulgui afegir

interfície gràfica o algun tipus de persistència al disc, només caldrà dissenyar sobre la capa de negoci. La tasca d'aquest node és rebre connexions d'altres nodes que vulguin entrar o sortir de la xarxa, i puntualment fer comprovacions diverses: mirar que els nodes no estiguin caiguts, comprovar la integritat de la xarxa i reordenar nodes per tal d'aprofitar millor les característiques de xarxa.

Això últim és especialment important donat que la xarxa pot esdevenir de un arbre equilibrat (millor cas) a una cadena (pitjor cas) i per tant, per optimitzar el sistema caldria reorganitzar els nodes de manera convenient, tot aprofitant al màxim l'associativitat de cada node.

3.4 Extensió del sistema

El model de híbrid amb pipelines i threads s'ha escollit també per un altre motiu que és la possibilitat d'afegir extensions, és a dir, altres tipus de nodes que facin altres tasques.

Es pot donar el cas que ens interessi que les observacions es guardin a disc per després estudiar-les, crear models estadístics o evolucions climatològiques. O bé, que de les observacions en volem crear gràfics: generar unes imatges que després es penjaran d'un servidor web o en un ftp. O fins i tot, podria ser que es volgués afegir un mòdul que ens permetés enviar observacions posades a mà, que no ens permeti mesurar la nostra estació meteorològica, com pot ser el tipus de núvols que es veuen a la zona o bé si hi ha tempesta elèctrica.

Tot això és possible usant el model de pipeline que s'ha descrit anteriorment. A cada node se li podrà afegir abans o després un d'aquests mòduls sense alterar les possibilitats de la xarxa. La figura 8D ho il·lustra:

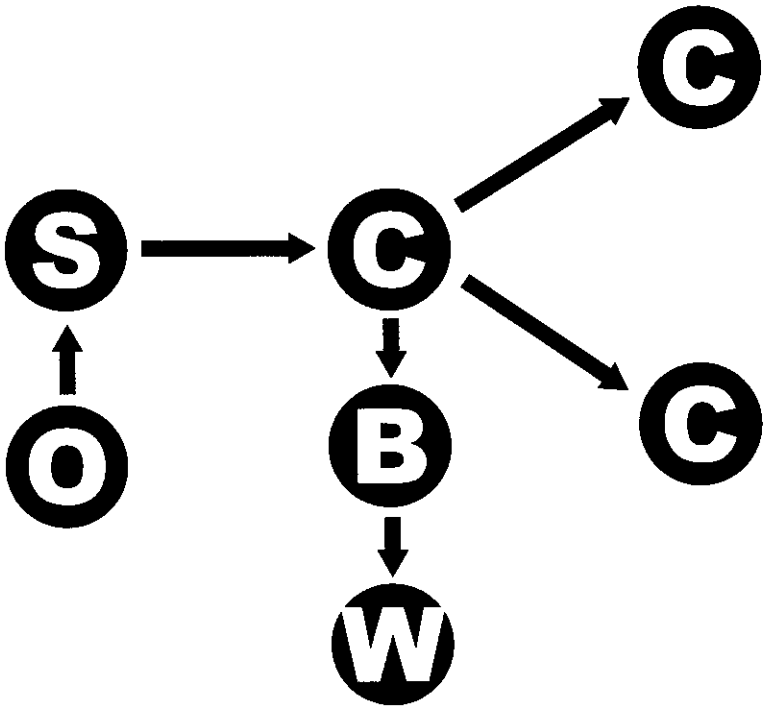


FIGURA 8D

A la figura 8D es pot observar com hi altres tipus de nodes a més dels nodes habituals (tipus S i C) :

Node tipus O : són nodes que permeten afegir observacions manuals.

Node tipus B : permeten emmagatzemar a base de dades les observacions.

Node tipus W : permeten elaborar gràfics o web dinàmics amb les dades.

Aquests tipus de nodes són inventats, però són totalment factibles en aquest model. En resum: es pot afegir qualsevol node per enriquir les observacions davant dels nodes tipus S i qualsevol node per elaborar altres informacions al costat dels nodes tipus C.

4 Mesures de coherència

4.1 Motivació

La coherència en una xarxa d'aquest tipus és essencial. Aquesta ha de cobrir dos cares molt diferents:

1. La coherència de les dades
2. La coherència de la xarxa

4.2 Coherència de les dades

La coherència de les dades ve referida als valors de les observacions meteorològiques que viatgen per xarxa: s'han d'assegurar un mínim de condicions per tal que aquestes siguin lògiques i compleixin una regulació.

4.2.1 Unitats

La regulació de les dades comença en un punt important: les unitats del sistema sobre les que es prenen les dades: així com el més normal és que un valor de temperatura sigui enviat en graus centígrads resulta en un problema considerable. Les estacions meteorològiques dels usuaris europeus estaran configurades en graus Celsius, mentre que les dels usuaris americans solen estar en graus Fahrenheit. Aquest fet pot resultar en que un usuari europeu, que per exemple viu a Granada, envia 32 i un usuari americà rep aquestes dades no les considerin igual. A 32ºC la temperatura és calorosa, mentre que 32ºF és un temperatura freda: equival a 0ºC.

La resolució ha de passar obligatòriament per uns estàndards internacionals dels que parlarem més endavant. Afortunadament hi ha una organització, la Organització Meteorològica Mundial, que ja ha resolt aquest problema: el present projecte segueix les recomanacions d'aquesta organització en termes de unitats de mesura.

El fet que les dades viatgin en unes determinades unitats no prohibeix a l'usuari que les visualitza el fer-ho en les unitats que es desitgi: no hi ha cap impediment en la manera de visualitzar les dades, la restricció és pròpia de l'àmbit de les dades que viatgen per xarxa.

4.2.2 Usuaris

Un altre problema que sol haver amb les observacions provinents d'estacions meteorològiques automàtiques és la manera en que l'usuari les ha instal·lat. Es suposa que l'usuari les ha instal·lat de correctament de tal manera que no hi ha cap impediment físic que doni lectures equivocades.

Exemples d'això poden ser col·locar termòmetres al sol, o penells que tenen algun flanc protegit per una paret i que els pot fer indicar només una direcció o causar turbulències que variïn les direccions del vent, o per exemple posar un pluviòmetre a prop d'un obstacle que evita que entri l'aigua si aquesta cau en una direcció determinada.

Davant d'aquest fet no es pot fer res, donat que molt difícilment es podrà avaluar la

instal·lació d'un usuari. El que sí es pot fer és un filtratge de les dades provinents d'un usuari del qual podem sospitar que té una estació incorrectament instal·lada.

Per a filtrar les dades d'un usuari de forma parcial o total, caldrà poder identificar a aquest usuari: això obliga a prendre mesures d'accés a la xarxa. Ha d'existir un sistema que permeti identificar als usuaris que participen a la xarxa amb les observacions que envien, però també ha d'existir algun mecanisme que no permeti adulterar aquesta informació al llarg del seu viatge pels diferents nodes.

4.2.2.1 Validació per usuari / password

Per tal d'assegurar que els usuaris que envien dades són de confiança es poden optar per diversos esquemes. El més fàcil és un sistema de validació per usuari i password: aprofitant que tots els nodes abans d'entrar al sistema han d'accedir al node d'enllaç per tal que aquest els assigni dins de la xarxa, es pot aprofitar aquest fet per a que el node que vulgui entrar s'identifiqui amb un usuari i un password assignats prèviament (per e-mail o com sigui) de tal manera que si no està en una llista determinada aquest node no podrà entrar a formar part d'aquesta xarxa.

Evidentment, es podran crear xarxes obertes, és a dir, amb connexió anònima. Aquest esquema és molt similar al clàssic esquema d'accés a ftp's públics on certs usuaris tenen privilegis per a pujar fitxers i altres usuaris (els anònims) només podran baixar-se'ls.

Aquest sistema té certs desavantatges considerant que algú es pot fer amb l'usuari /

password de la persona, o bé caçar-los de la xarxa usant un sniffer.

4.2.2.2 Validació per clau pública / privada

De manera similar que l'esquema anterior es pot realitzar un mecanisme de clau pública i privada (tals com l'RSA [1] o ElGamal [2]) de tal manera que s'incrementa la seguretat en la validació de l'usuari: en estar tot encriptat es fa més difícil que una intercepció dels paquets de validació, en el que un usuari podria llegir el parell usuari / password i passar per la persona que els té.

Aquest sistema té algunes vulnerabilitats com els atacs tipus "man in the middle" ja resolts en alguns protocols de validació com SSL [3].

4.2.2.3 Observacions alterades

La possibilitat que un usuari modifiqui el codi de la aplicació per tal d'alterar les observacions que li arriben és un fet que no es pot descartar.

No obstant, aquest és un fet molt més fàcil de detectar donat que el temps, tot i ser variable segueix un patró de comportament. Una alteració de temperatura a la alça o a la baixa de manera considerable, o de pressió atmosfèrica sense alteracions en el vent són indicatius de sospita. Per la pròpia dinàmica atmosfèrica, canvis de pressió molt grans provoquen vents molt potents: és el cas clàssic d'un huracà, el gradient de pressió atmosfèrica és molt gran i els vents que bufen són extrems.

4. Mesures de coherència

També hi ha la possibilitat de caçar al node que està provocant la alteració de les dades posicionant nodes client virtuals en diversos punts de la xarxa. És a dir, es poden col·locar nodes client que no permetin visualitzar les dades sinó que el que fan és connectar-se a diverses zones la xarxa i observar si hi ha canvis de valors, per una mateixa observació d'un determinat node. En el següent gràfic 9A tenim una possible situació:

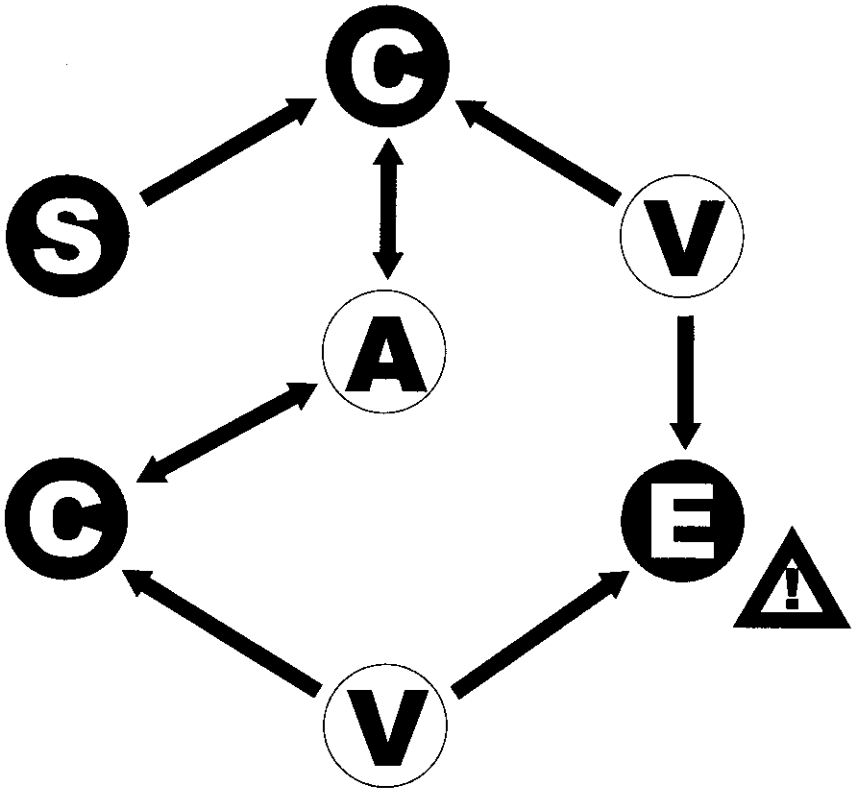


FIGURA 9A

Sent A un node que alteri les dades d'una observació i V el node client virtual que es connectarà a un client C determinat per obtenir dades de les observacions que passin pel client al que està connectat. El sentit de circulació de la informació serà sempre del node

estació S cap el node C, posteriorment passarà per A i acabarà en C. Si es connectessin dos nodes client virtuals als nodes C, els valors que observaran d'una mateixa estació S serà diferent (un node V el veurà sense alterar i l'altre node V alterat). Aquests nodes enviaran la informació al node d'enllaç E que observarà aquesta diferència de valors per una mateixa estació. Canviant els nodes client virtuals de lloc, és a dir el client al que es connecten, es pot acotar el node A que altera les observacions enviades: d'aquesta manera el node enllaç podria impedir que es realitzin connexions a aquest node.

4.3 Coherència de la xarxa

La coherència de la xarxa es pot resoldre fàcilment de la manera en que està dissenyat el sistema. Donat que els node que entren o que surten del sistema han de passar obligadament per un node enllaç, aquest s'encarrega d'assignar aquests nodes a la part de la xarxa on sigui convenient: el node enllaç té la particularitat que coneix tota la topologia de la xarxa i això evita incoherències de connexions.

4.3.1 Desconnexions sense avís

Per errors de xarxa intencionats o no, la topologia de la xarxa pot quedar incoherent. Com que la estructura de la xarxa és en arbre, amb un sol node que no enviï informació hi ha prou per a que tots els que depenen d'ell tampoc la puguin veure.

Una solució és l'ús del timeout. Es poden realitzar proves per a veure si un node respon

periòdicament per part del node enllaç o dels seus veïns: es pot fer d'una manera similar a la comanda "ping". Si un node no respon de manera continuada, aleshores es prendrà una iniciativa que permeti mantenir la coherència de la xarxa, ignorant aquest node i connectant els seus veïns entre sí. Si això passa el node haurà d'entrar de nou en xarxa i demanar una nova identificació.

4.4 Bibliografia i referències web

1: Algorisme de clau pública RSA [Wikipedia]

http://en.wikipedia.org/wiki/RSA

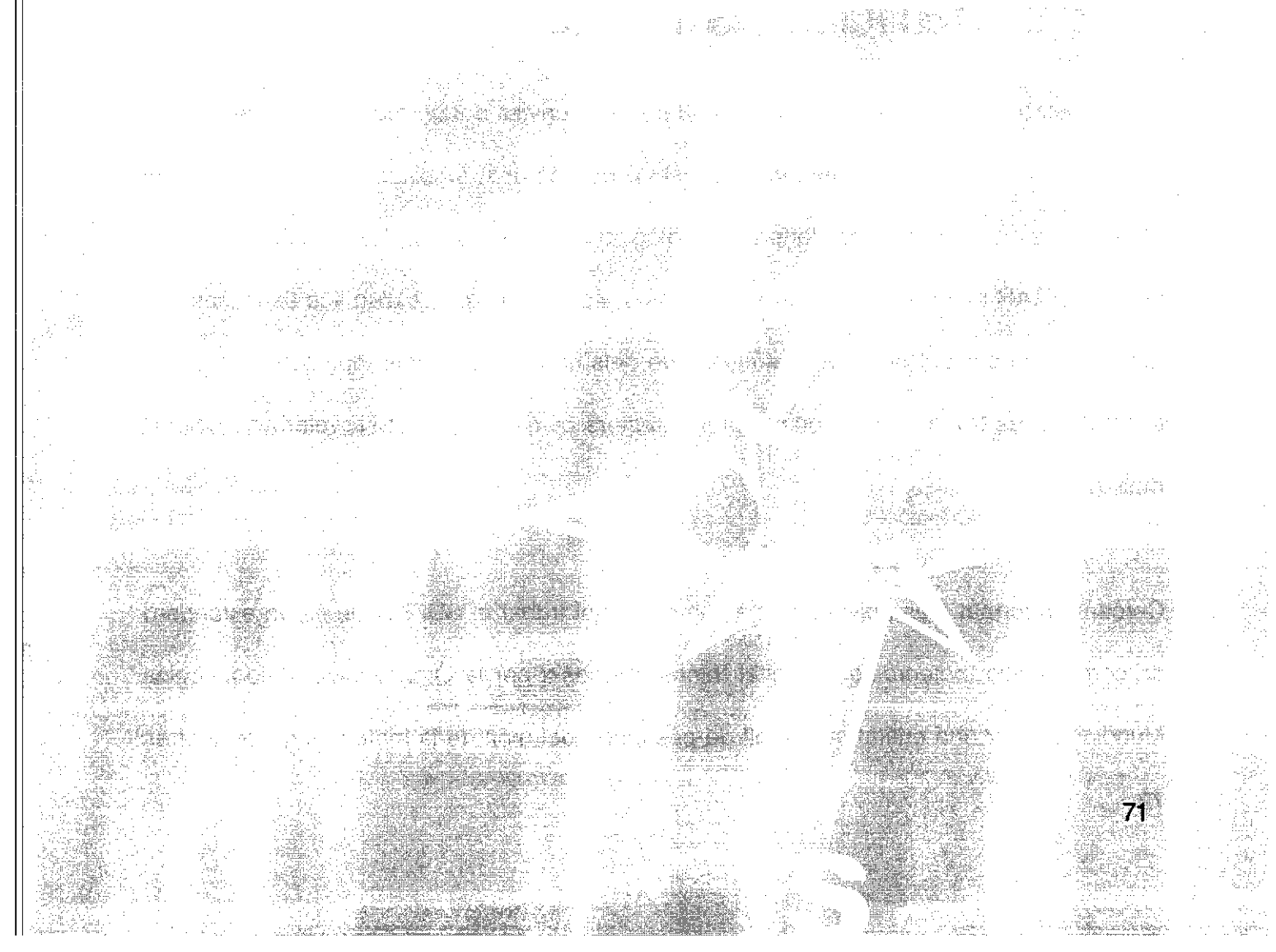
2: Algorisme de clau pública ElGamal [Wikipedia]

http://en.wikipedia.org/wiki/ElGamal_encryption

3: Sistemes de seguretat SSL [Wikipedia]

http://en.wikipedia.org/wiki/Transport_Layer_Security

5 Estàndards



5.1 Necessitat d'estandarditzaci6

Quan es parla de la necessitat d'estandarditzaci6 s'enfoca en la manera en que es pot fer el sistema més interoperable, és a dir, que pugui funcionar amb altres components independents o acoblats al mateix sistema.

En el present projecte s'han usat certs estàndards que proporcionen una base sòlida sobre la que construir certes parts del sistema. D'aquests estàndards s'ha avaluat la seva evolució, el seu ús i la seva extensió, és a dir, si són usats i reconeguts.

5.2 Estàndards meteorològics

En la meteorologia hi ha diversos estàndards per enviar dades meteorològiques. En la aviació usen un sistema anomenat METAR [1] que és usat àmpliament per a comunicar als pilots les possibles condicions meteorològiques que es poden trobar. Cal considerar que el METAR és un format basat en caràcters, és a dir, el temps d'un lloc es representa mitjançant un conjunt de caràcters cadascun dels quals té un sentit molt determinat. Per exemple, si es troben en el conjunt tres caràcters com aquests "+SN" voldrà dir nevada molt forta.

Cercant un format més universal i més ampli es pot trobar el BUFR, que és un estàndard de comunicació meteorològica digital (binari) proposat per la Organització Meteorològica Mundial. En ser una entitat amb molt de pes dintre del món de la meteorologia aquest és

un estàndard que s'ha estès molt ràpidament i que avui dia gaudeix d'un ús en plenes facultats: prova d'aquest ús és el fet que el BUFR [2] està vora a complir els 30 anys i està en la quarta revisió. En base al BUFR es va crear el CREX, que vindria a ser l'equivalent del METAR però seguint el model BUFR, és a dir un format de números i lletres (no digital) que representa una situació meteorològica de la mateixa manera en que ho pot representar el BUFR de manera digital (binària).

5.2.1 El format BUFR

El format BUFR [3] és un format que ocupa gairebé un centenar de pàgines, aquí només s'exposarà de manera molt senzilla la seva estructura per tal d'avaluar-lo.

Un paquet BUFR s'estructura en sis seccions diferents. Fent una analogia es pot pensar en cada secció com un capítol d'un llibre, on la concatenació de les seccions dóna el paquet BUFR, que en aquest cas seria el llibre. La figura 10A representa aquest fet:

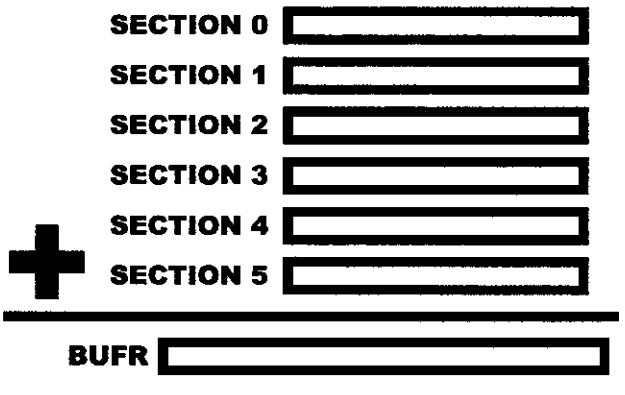


FIGURA 10A

La SECTION 0 conté les dades bàsiques: el tipus de paquet BUFR i la longitud del mateix.

La SECTION 1 conté dades del centre oficial que ha originat les dades i la hora a la que s'han generat.

La SECTION2 és opcional i pot contenir dades codificades sense seguir l'estàndard.

La SECTION 5 només conté els valors "7777" que indiquen final de paquet.

La SECTION 3 i la SECTION 4 són diferents: no tenen un format predefinit. La SECTION 3 contindrà uns codis que indiquen el *tipus* d'observacions que conté el paquet. La SECTION 4 contindrà els *valors* d'aquestes observacions seguint el tipus definit a la anterior SECTION. Per exemple, si a la SECTION 3 es defineix en aquest ordre: "temperatura de bulb sec i humitat relativa" aleshores en la SECTION 4 apareixerà en aquest ordre "23ºC i 65%".

Un dels avantatges de usar el format BUFR és que les unitats venen definides en el estàndard la qual cosa estalvia la incoherència de les unitats en les observacions. El BUFR té com a avantatge que qualsevol meteor existent està descrit per tant es cobreix tot el ventall de possibles meteors estranys que no s'hagin previst inicialment.

El BUFR també té inconvenients. La codificació és molt feixuga donat que requereix unes taules molt grans i la manera de codificar és millorable. Per exemple, el BUFR no contempla dades en format de coma flotant (tipus float). Si hi ha alguna observació susceptible de ser mesurada en decimals obliga a multiplicar-la per una potència de 10 fins que queda com un dada entera. Per exemple, mai s'enviarà una velocitat del vent de 5.2 m/s sinó que s'enviarà 52; si el anemòmetre mesurés amb centèsimes perdríem aquesta precisió.

També s'ha d'afegir la limitació que l'enter que s'envia ha de ser sense signe, és a dir positiu: no podrem enviar una coordenada geogràfica de -20.1545º de latitud, sinó que li sumarem 90º de tal manera que la Antàrtida (a -90º de latitud) quedarà com a 0º.

En conclusió, és preferible perdre una mica de precisió a dissenyar un estàndard completament nou i que cobreixi tots els meteors existents. El cost que suposa el disseny d'un estàndard sumat a la acceptació del mateix fan molt més viable acceptar el BUFR com a format de transport d'observacions.

5.3 Mapes

En el present projecte la meteorologia està relacionada de manera directa a un altre aspecte: els mapes. Es fa necessari un mapa que posicioni les dades de les observacions que envien les diferents estacions.

5. Estàndards

Dintre del món de la cartografia, s'ha anat evolucionant en la estandardització de mapes i serveis relacionats. Tot just en aquests últims anys, la cartografia ha donat pas a un ventall de possibilitats molt gran orientant esforços cap a Internet. Avui dia és molt fàcil comptar amb un carrerer de Barcelona, són moltes les webs que ofereixen serveis cartogràfics globals de manera gratuïta: Google, Microsoft, OpenStreetMap, etc. A més la tipologia dels mateixos a anat cobrint cada vegada més espai: des de mapes topogràfics a mapes de relleu.

L'organisme que s'ha aixecat amb autoritat dins del món de la cartografia ha estat el OGC (Open Geospatial Consortium) que ha proposat estàndards de cartografia per a poder crear sistemes interoperables en aquest aspecte.

Un exemple és el Web Map Service [4] que és un servei basat en el protocol web (HTTP) que, donada una URL que conté una petició d'una zona a mostrar, retorna una imatge en format típic (JPG, GIF, PNG) amb el terreny especificat per la petició.

N'hi ha més estàndards proposats per l'OGC, un format d'ells és el KML creat per Google i que es pot obtenir de algunes de les seves aplicacions tals com Google Earth. O també un altre servei, el WCS [5], que permet un format vectorial: en contrast al WMS el WCS permet declarar línies, punts o talls geogràfics per tal de realitzar un anàlisis geogràfic.

A conseqüència de la existència d'aquests estàndards el nombre de projectes relacionat amb la cartografia a Internet ha incrementat considerablement. Cada vegada hi ha un

major nombre de proveïdors gratuïts de cartografia en diversos formats. En els següents apartats es veurà com s'ha aprofitat aquesta infraestructura per a poder crear una base cartogràfica gratuïta sobre la que poder mostrar imatges.

La necessitat d'accedir a mapes que estiguin allotjats en servidor d'Internet es fa molt necessària: una cartografia bàsica de la Terra pot ocupar 3 GB i això és inviable per a un usuari que només vol tenir una sèrie de símbols sobre un mapa. Per tant, l'ús d'un servei com el WMS permet esquivar aquest problema aconseguint una cartografia molt detallada i sense cost per a l'usuari.

5.4 Bibliografia i referències web

1: Article sobre el METAR [Wikipedia]

http://en.wikipedia.org/wiki/METAR

2: Article sobre el BUFR [Wikipedia]

http://en.wikipedia.org/wiki/BUFR

3: Especificació BUFR [Organització Meteorològica Mundial]

http://www.wmo.int/pages/prog/www/WMOCodes/OperationalCodes.html

4: Article sobre el Web Map Service [Wikipedia]

http://en.wikipedia.org/wiki/Web_Map_Service

5: Article sobre el Web Coverage Service [Wikipedia]

http://en.wikipedia.org/wiki/Web_Coverage_Service

5. Estàndards

6 Prototip

6.1 Motivació

Amb motiu de estudiar més a fons la arquitectura presentada s'ha volgut realitzar un prototip que incorpori una sèrie de funcionalitats bàsiques del projecte. El prototip no pretén ser més que un banc de proves inicial sobre el que es desenvoluparà el projecte.

Fer un prototipus presenta l'avantatge de poder establir un punt de partida sobre el Sistema Meteorològic Distribuït que és desenvoluparà en el futur. En el prototip, tal i com es descriurà en els següents apartats, es fa una avaluació dels artefactes, ja siguin llenguatge, framework o api que són plausibles d'usar en un futur amb les seves avantatges i els seus inconvenients.

6.2 Requeriments

L'abast del projecte compren la realització d'un petit sistema basat en la arquitectura proposada que ha de realitzar diferents funcions:

1. Nodes que generin dades meteorològiques
2. Nodes que visualitzin les dades
3. Un node que gestioni la xarxa.

Les tres funcionalitats descrites s'hauran d'integrar en tres programes per separat tal i com es defineix en el disseny del sistema.

El projecte ha de poder executar-se sobre Windows, però sempre tenint en compte la possibilitat de tenir una adaptació relativament fàcil del sistema a Unix, és a dir, fàcilment portable.

6.2.1 Requeriments funcionals

6.2.1.1 Comunicació de dades entre usuaris

Es vol fer que les dades enregistrades per una estació d'un usuari sigui enviada a d'altres usuaris mitjançant una xarxa.

6.2.1.2 Minimitzar el temps d'expansió

Es vol minimitzar el temps que transcorre entre la obtenció de una mesura meteorològica per una estació meteorològica d'un usuari (com pot ser la temperatura), amb la posterior obtenció d'aquesta dada per la resta d'usuaris.

6.2.1.3 Infraestructura mínima

Es cerca aprofitar la infraestructura actual que pot tenir l'usuari mitjà, sense requerir contractar serveis més enllà dels habituals en les persones que actualment es connecten a la xarxa (les típiques connexions d'accés ADSL), o un hardware concret d'altes prestacions.

6.2.2 Requeriments no funcionals

6.2.2.1 Tipus de xarxa

El sistema s'executarà en xarxa usant protocols habituals d'Internet, tals com TCP o UDP sobre IP. No es limitarà l'ús a una xarxa pública com Internet sinó que també es podrà executar en entorns locals.

6.2.2.2 Estandardització

Sempre que existeixin, prevaldrà l'ús dels estàndards en la comunicació meteorològica. Si existeix un estàndard de comunicació de dades meteorològiques, o de visualització d'aquestes dades, es farà servei d'aquestos.

6.2.2.3 Comportament automàtic

El sistema s'ha de comportar de manera automàtica en tot allò que està relacionat amb la distribució de dades meteorològiques, és a dir, minimitzar la intervenció de l'usuari en aquest aspecte.

6.2.2.4 Geolocalització de dades

Les dades meteorològiques es podran visualitzar sobre un mapa de tal manera que es pugui obtenir una visió instantània àmplia sobre el conjunt de valors que són emesos pels usuaris que disposen d'estacions meteorològiques.

6.3 Avaluació de tecnologies

Prèviament al treball de disseny del sistema software que compondrà el prototip s'han avaluat les diverses alternatives que hi ha en quant a desenvolupament (llenguatges, frameworks o api). La avaluació s'ha tingut en compte sempre tenint la vista més en la direcció que prendrà el projecte en el immediat futur més que en la resolució ràpida i fàcil d'un prototip. S'ha decidit actuar d'aquesta manera per tal d'afrontar des de primer moment les possibles dificultats que es presentin en la realització del sistema de manera completa.

6.3.1 Llenguatges

Dels diversos llenguatges que s'han avaluat s'ha pres una solució híbrida: s'ha usat C++ i Java.

La primera alternativa era realitzar el sistema completament en C. Això presenta un clar avantatge d'eficiència sumat al fet que de cara a un futur és molt més fàcil la interacció amb una estació meteorològica degut a que moltes estacions usen port sèrie o USB i els drivers per usar-les, si mai surten públics, són per a C / C++. El problema és que la orientació a objectes dona molta potència i no usar-la seria desaprofitar-la. D'altra banda la portabilitat de codi C entre sistemes Windows i Unix és complicada.

S'ha avaluat fer tot el projecte en C++, que presenta l'avantatge de la eficiència sumat a la

orientació a objectes i la interfície per a actuar amb les estacions meteorològiques. No obstant el fet de desenvolupar per a Windows ha resultat un problema de cara a la portabilitat.

Java [1] resol el problema de la portabilitat i de la orientació a objectes, en canvi dificulta la programació a baix nivell.

La idea de usar un mateix llenguatge de programació és molt atractiva perquè amb la orientació a objectes el sistema pot compartir el mateix tipus d'objectes entre els diversos programes. que compondran el sistema donant lloc a la reutilització.

La solució ha estat usar diferents llenguatges per a cada tipus de node. Per al node estació, que és qui ha d'actuar amb la estació meteorològica s'ha usat C++. En canvi per al node client, i per al node enllaç s'ha usat Java. La particularitat ha estat que les classes més bàsiques que comparteixen els diferents nodes usen un conjunt de classes equivalent de manera que a nivell conceptual, en el disseny del prototip, es poden tractar igual: només variarà la implementació de les mateixes.

En Windows, el compilador i el conjunt d'eines amb que s'ha implementat el codi de la estació ha estat amb el paquet MinGW [2], que és un entorn minimalista que emula l'entorn bàsic GCC disponible a qualsevol sistema Unix. No fa una emulació tan completa com un altre paquet anomenat CygWin [3] que permet entorns UNIX gairebé complets sobre Windows, fins i tot amb sistema de finestres XWindows.

6.3.2 Sockets

Els sockets no varien substancialment entre una implementació en C++ i una altra en Java. La manera de construir un Socket un Windows és molt menys abstracta que la d'un socket en Java i està molt lligada al sistema.

La part positiva de usar MinGW per al node estació és que es pot treballar directament amb la API de Windows de manera directa i això dóna molt potència. Si es realitza el mateix amb la llibreria de Unix que virtualitza CygWin el codi és més portable però quan es vulgui realitzar la interfície amb la estació en Windows hi haurà més inconvenients. Per tant s'han decidit afrontar aquest inconvenients de bon principi.

6.3.3 Visualització de les dades

La visualització de les dades sobre un mapa ha ocupat una part considerable d'aquest projecte. Des de bon principi s'ha barallat la idea d'usar un sistema d'informació geogràfica (SIG o GIS en anglès) per a dibuixar les observacions sobre mapes.

Per tal de visualitzar les dades per pantalla es va optar inicialment per a una llibreria GIS en C++, C o Java. Es van optar per diverses alternatives com el Mapnik [4], o altre llibreries en que estan basats GIS open source que hi ha a la xarxa com les llibreries GDAL/OGR [5]. La major part d'aquestes llibreries estaven fetes en C++ però eren portables per tant es resolia gran part dels requeriments. El problema que presentaven és

que o no eren massa genèriques i no generaven els gràfics que es requeria o bé estaven en fase de ple desenvolupament i mancava documentació o faltaven certs errors per solucionar.

Posteriorment es va avaluar la conveniència d'usar un GIS complet que integrés un conjunt de llibreries prou gran. En aquest cas es van trobar diverses solucions tals com GeoServer [6] (Java), Quantum GIS [7] (C++) o Grass [8] (C). El problema que presentava és el fet que aquests sistemes són molt complexos de preparar i de instal·lar per l'usuari, sumat al fet que només s'usarà una part molt petita. També es va pensar en afegir una mínima cartografia de la Terra però això resultava en un paquet de més de 500 MB en funció del tipus de cartografia que s'inclogués, xifra que podia arribar als 3GB fàcilment. Impensable per a un simple prototip i difícilment justificable en un visor d'observacions meteorològiques.

La solució, que en un principi es va descartar, ha estat la que més tard s'ha adoptat. L'ús d'una llibreria que integra la capacitat d'absorbir informació cartogràfica d'Internet amb la capacitat d'afegir elements sobre aquest mapa per tal de simbolitzar les observacions. Aquesta solució és OpenLayers [9], una llibreria open source JavaScript força usada arreu que facilita el desenvolupament d'aplicacions web en la que s'hagi de treballar amb cartografia. Aquesta llibreria és capaç d'absorbir informació cartogràfica de diverses fonts (Google Maps, OpenStreetMaps, etc...) i permet integrar fàcilment dades pròpies.

6.4 Entorn local

El prototip que es vol realitzar està orientat a poder-se usar per Internet de manera directa, sempre i quan es configuri correctament les connexions. Però per tal de realitzar el seu desenvolupament de manera ràpida s'ha usat un entorn local, és a dir, llençant tots els nodes en un mateix ordinador.

Això té unes conseqüències considerables en el aspecte de threads i de comunicacions per sockets. Afortunadament la càrrega de CPU per thread s'espera que sigui baixa i això permet fer simulacions amb un mínim nombre de nodes.

6.4.1 Sockets

Les conseqüències d'usar un entorn local per simular tot un sistema complet esdevenen molt lligades al sistema operatiu sobre el que estan programades. Els sistemes Unix, tals com Linux, Solaris o les variants BSD tenen una gestió molt òptima en l'ús dels sockets.

En canvi Windows, a partir de les versions de Windows XP Service Pack 2 apareixen limitacions en molts aspectes. Una d'elles consisteix a limitar el nombre de sockets oberts per software tractant el programa com un si fos "codi maliciós" (virus o trojans, etc...) de manera que a cada cert interval de temps se li permet al programa obrir més sockets.

Les versions de Windows prèvies al Service Pack 2 de Windows XP no inclouen aquesta

limitació. S'ha descartat el suport de sistemes operatius Windows obsolets tals com Windows 98, Millenium o anteriors donat que la gestió de sockets és encara més deficient.

6.5 Bibliografia i referències web

1: Llenguatge de programació Java [Java]

http://www.java.com/en/

2: Entorn de C++ MinGW [Minimalist GNU for Windows]

http://www.mingw.org/

3: Entorn de C++ CygWin [CygWin]

http://www.cygwin.com/

4: Llibreries pel desenvolupament d'aplicacions cartogràfiques [Mapnik]

http://mapnik.org/

5: Llibreria d'abstracció de dades espacials [GDAL]

http://www.gdal.org/

6: Servidor GIS en Java [GeoServer]

http://geoserver.org/display/GEOS/Welcome

7: Servidor GIS en C++ [Quantum GIS]

http://www.qgis.org/

8: Servidor GIS en C [Grass]

http://grass.itc.it/

9: Llibreria de cartografia en JavaScript [OpenLayers]

http://www.openlayers.org/

7 Protocol

7.1 Necessitat d'un protocol

Degut a la seva arquitectura separada en tres tipus de nodes (estació, client i enllaç) i com a aplicació que usa la xarxa, el Sistema Meteorològic Distribuït fa ús de un protocol per tal d'establir un diàleg entre les seves parts.

El protocol s'ha dissenyat de zero donat que encara no hi ha sistemes similars a aquest, no obstant s'ha observat el comportament d'altres protocols coneguts usats en Internet per tal d'establir un punt de partida en el disseny del propi protocol.

7.2 Bases del disseny del protocol

El disseny de l'actual protocol guarda una certa similitud amb els protocols FTP o HTTP. El protocol FTP [1] es basa en una sèrie de comandes que serveixen per a realitzar transferència de fitxers. Mitjançant aquestes comandes es pot navegar en el sistema de fitxers d'un ordinador remot i descarregar-los. El protocol HTTP [2] també té una sèrie de comandes per obtenir fitxers: seguint una sèrie de passos establerts, un ordinador es connecta a un altre i li envia un petició d'un fitxer (generalment un document web (XML, HTML, ...), però pot ser també qualsevol altre tipus de fitxer (ZIP, PNG, OGG, ...).

L'actual prototip implementa un sistema de comandes similar que conformarà el protocol d'interacció entre els nodes.

7.3 Esquema de representació

Prèviament al disseny del protocol s'han adoptat una sèrie de convencions per a dissenyar els diversos grafs d'estats que compondran el protocol.

Per tal de dissenyar el protocol s'han usat diversos grafs d'estats, on cada transició entre els nodes d'un graf d'estats és una comanda que pot ser rebuda o enviada per un determinat node. La següent figura il·lustra 11A aquest fet:

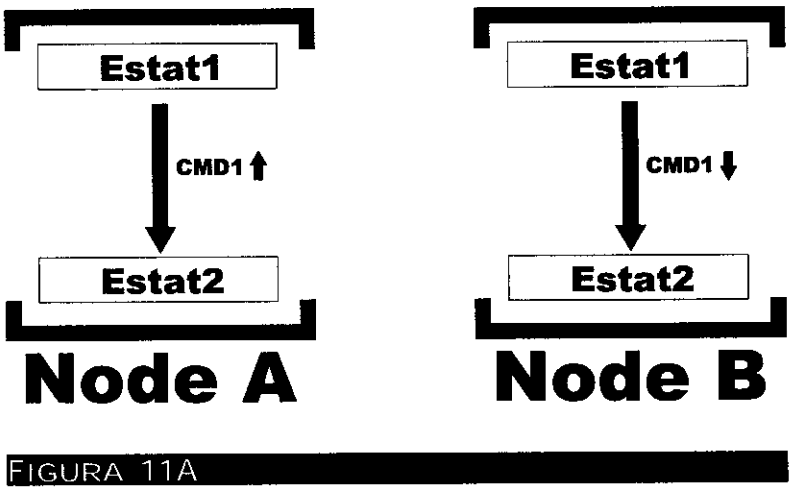


FIGURA 11A

L'exemple de la figura 11A explica la manera en que s'han dissenyat els grafs d'estats de cada node i com s'envien comandes entre un i altre node. A la figura es pot veure com hi ha dos nodes que s'envien comandes: un Node A i un Node B. Tots dos nodes tenen una sola transició d'un Estat1 a un Estat2 que ve indicada per una fletxa que va d'un estat a

l'altre. Al costat de la fletxa s'indica la comanda que s'envia, en aquest cas CMD1 i el sentit de la comanda: una fletxa cap amunt indica una comanda que surt del node, mentre que una fletxa cap avall equival a que la comanda es rep en el node. En l'instant en que la comanda surt per xarxa cap el node destí es realitza la transició entre els estats del Node A. De manera anàloga el mateix moment en que es res la comanda al Node B es passa de l'Estat1 a l'Estat 2.

7.4 Disseny del protocol

El protocol s'ha dissenyat amb la idea de resultar el més simple possible per motius d'eficiència i de simplicitat. El eficiència és justifica per ella mateixa: s'ha de procurar minimitzar la quantitat d'informació a enviar, un estalvi d'uns pocs bytes en cada paquet suposa, a la llarga, un gran estalvi. La simplicitat es fa necessària donat que el protocol a dissenyar és per a un prototip que s'executarà en un entorn local, motiu pel qual s'ha estalviat en codis de correcció i altres mesures de coherència de paquets.

7.4.1 Paquets d'informació

El protocol no només defineix grafs d'estats a seguir sinó que també com s'intercanvia la informació entre els nodes.

Cada transició vista en la figura 11A equival a un paquet d'informació que s'envia (o es rep) des de (o cap a) un node determinat. Cada paquet d'informació es composarà d'una

capçalera comú de mida fixa més un cos de mida variable.

A la capçalera es contindrà la informació de la versió de protocol usat, la comanda que s'inclou en el paquet i la mida del paquet sencer (inclòs la capçalera). El cos del paquet dependrà de la comanda inclosa a la capçalera.

En aquest document no es detallarà el contingut de cada paquet perquè no es considera essencial per entendre el prototip. En canvi sí es detallaran les comandes que contenen cada paquet i que es poden veure reflectides en els grafs d'estats sí inclosos en el present document.

7.4.1.1 Les comandes

La comanda inclosa a la capçalera sempre serà un conjunt de quatre caràcters en majúscula definits en codi ASCII. Un exemple de comandes usades en el protocol són 'AUTH', 'ACCP' o 'ADDS', que són bàsicament acrònims que guarden una similitud de paraules amb anglès i que realitzen accions que tenen relació amb aquestes ('authorization', 'accept', 'reject', 'add station).

7.4.2.2 Els estats

Els estats no guarden cap relació entre els diagrames simplement se'ls ha posat nom per tal de definir de manera més clara l'estat, però no tenen cap conseqüència en el disseny.

7.5 Descripció del protocol

A continuació es descriu el protocol entre els diversos nodes usant la notació vista a la figura 11A separades segons la interacció que requereixin els nodes en cada moment.

7.5.1 Nou node estació

Aquests grafs corresponen a la interacció produïda entre un node estació i un node enllaç i un possible node client quan el node estació entra per primer cop a la xarxa.

Com en anteriors capítols s'ha optat per usar la següent notació per a la descripció dels nodes: un node E equival a node d'enllaç, un node S a node estació i un node C a un node client.

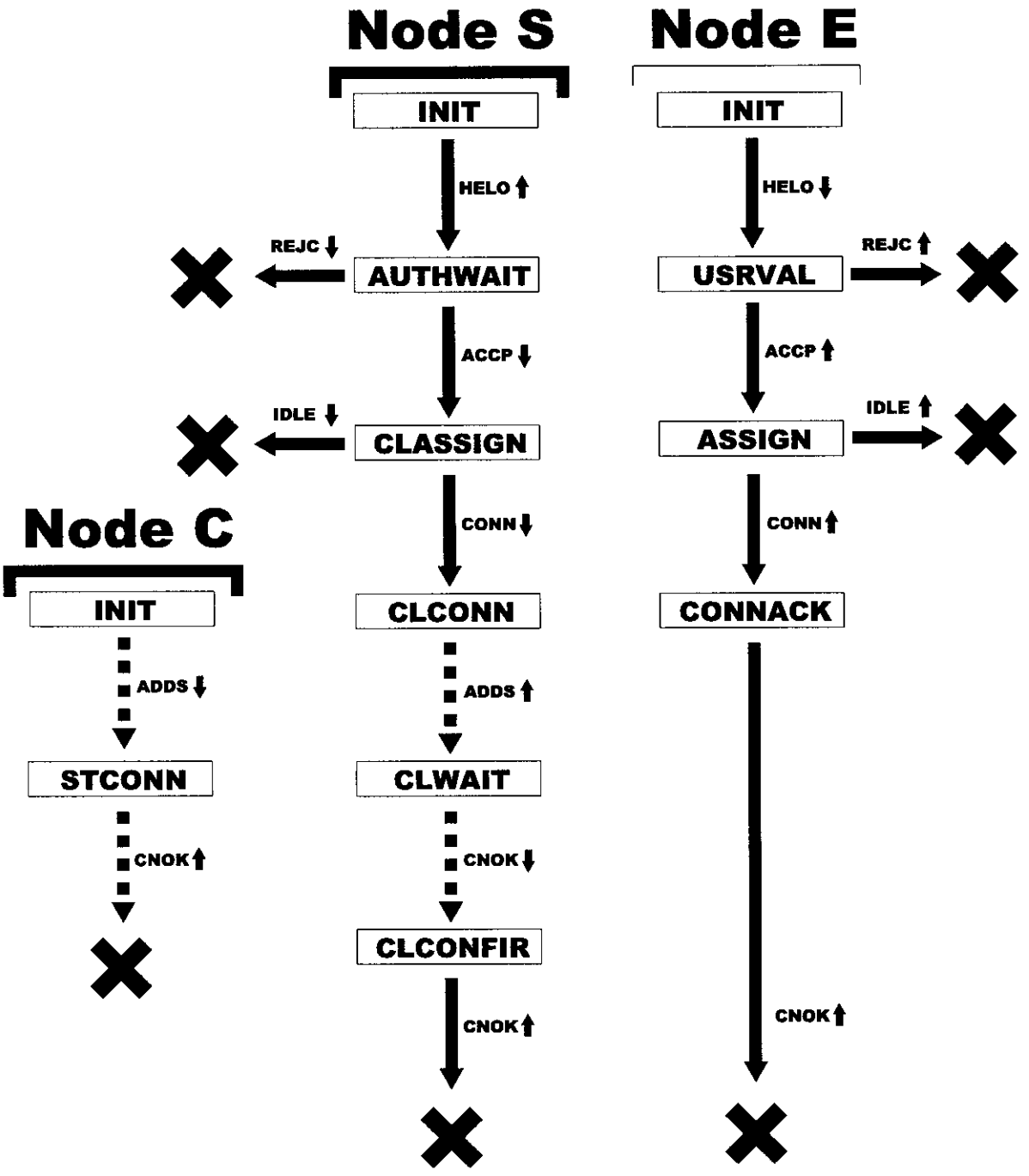


FIGURA 11B

El graf de la figura 11B descriu la situació següent. Un node S qualsevol vol entrar en la xarxa, i llença un paquet HELO que conté dades identificatives de l'usuari (tal i com s'ha

parlat a l'apartat 4.2.2), el node d'enllaç rep el paquet i, si correspon a un usuari reconegut en el sistema, envia un paquet ACCP. En cas contrari envia un paquet REJC i finalitza la conversa entre aquests dos nodes resultant amb que el node S no ha pogut entrar dins de la xarxa.

En cas que es rebi el paquet ACCP aleshores es poden rebre dos tipus de paquets: un IDLE en cas que no hagi cap node al que la estació es pugui connectar o un paquet CONN en cas que la estació sí es pot connectar a un client C. En el cas que el node S rebi IDLE la conversa s'atura.

Si es rep CONN aleshores el node S es connecta al node C indicat dintre del paquet CONN creant una nova connexió cap a aquest node, que s'ha mostrat mitjançant una transició amb línies discontínues. El node C rep la petició de connexió del node S i li respon CNOK. Tan bon punt el node S rep el CNOK aleshores enviarà al node E un altre paquet CNOK per confirmar que s'ha connectat amb el node C indicat.

7.5.2 Nou node client

Es dóna el cas en que un nou node client entra a formar part de la xarxa. Aquest se li assigna a altres nodes com poden ser nodes C o S.

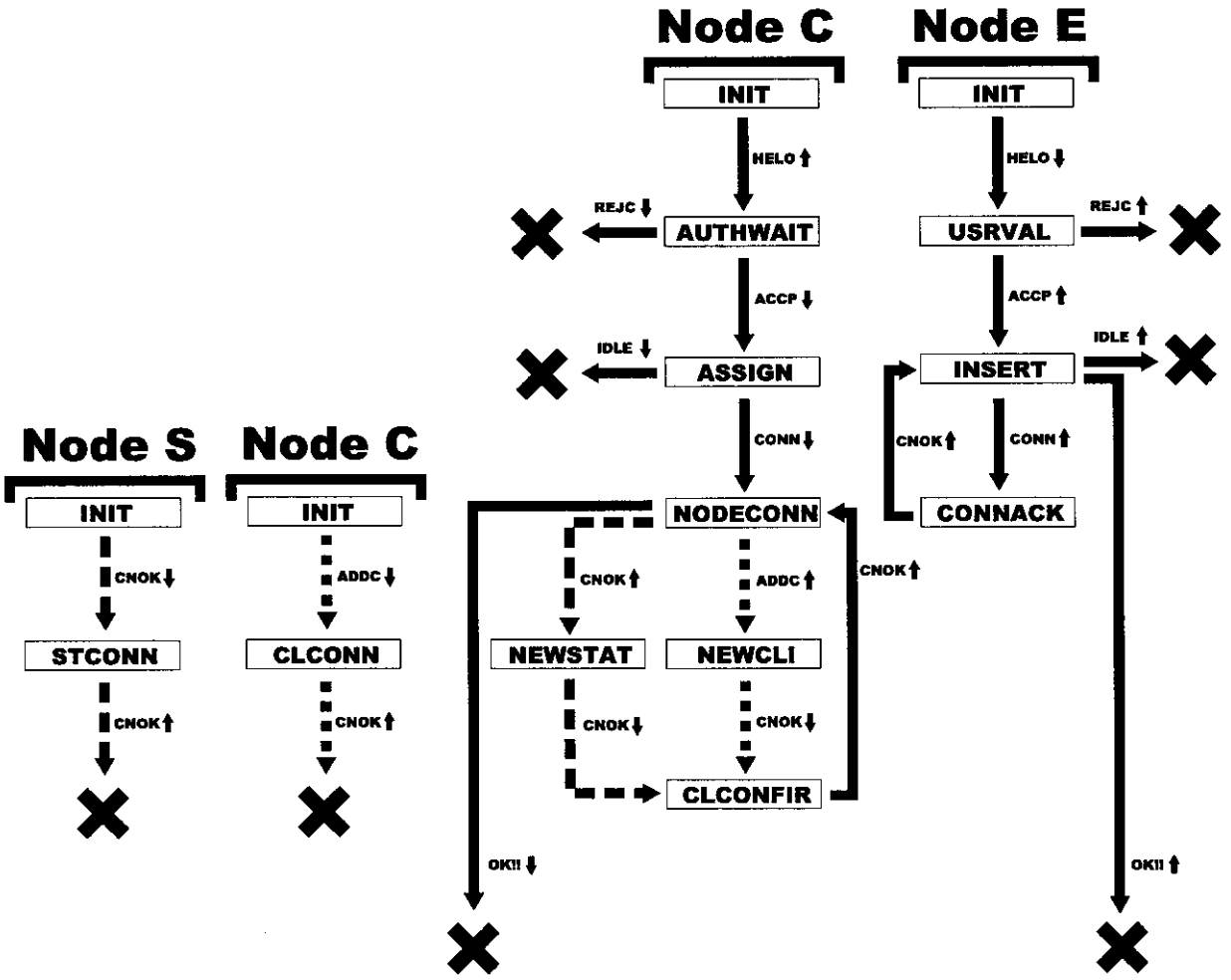


FIGURA 11C

La primera part de la connexió d'un nou node C és similar a la que s'ha vist en el cas anterior, on un nou node S entrava a formar part de la xarxa.

7. Protocol

S'envia un paquet HELO per comprovar si el client forma part de la xarxa i posteriorment se li confirma el seu accés amb el paquet ACCP. De la mateixa manera que passava amb el node S vist en l'anterior apartat si no hi ha cap node més a la xarxa s'envia un paquet IDLE.

Ara bé, si hagués alguna estació o algun client per connectar s'entrarà en un bucle d'assignació. Des del node E s'enviaran paquets CONN al node C i s'esperarà a que el node C faci la seva part. El node C rebrà informació en el paquet CONN dels nodes a connectar. Dintre del paquet CONN s'indica el tipus de client a connectar si és de tipus S o si és de tipus C.

Si és de tipus S, representat en el graf d'estats amb una línia amb guions, aleshores es farà una nova connexió del node C al node S i s'enviarà un CNOK. S'esperarà a que el node S retorni el CNOK que rebrà de nou el client C, moment en que es farà el mateix del node C cap el node E.

De manera anàloga quan es vulgui connectar el node C a un altre node C veí, s'enviarà un paquet ADDC al veí i s'esperarà a rebre d'ell un paquet CNOK. De la mateixa forma que el cas anterior el node C envia un paquet CNOK confirmant que ha establert connexió amb el veí.

Un cop s'acabi de connectar amb tots els clients aleshores el node E enviarà un paquet OK!! indicant que ja no hi ha més connexions a realitzar.

7.5.3 Desconnexió d'un node estació

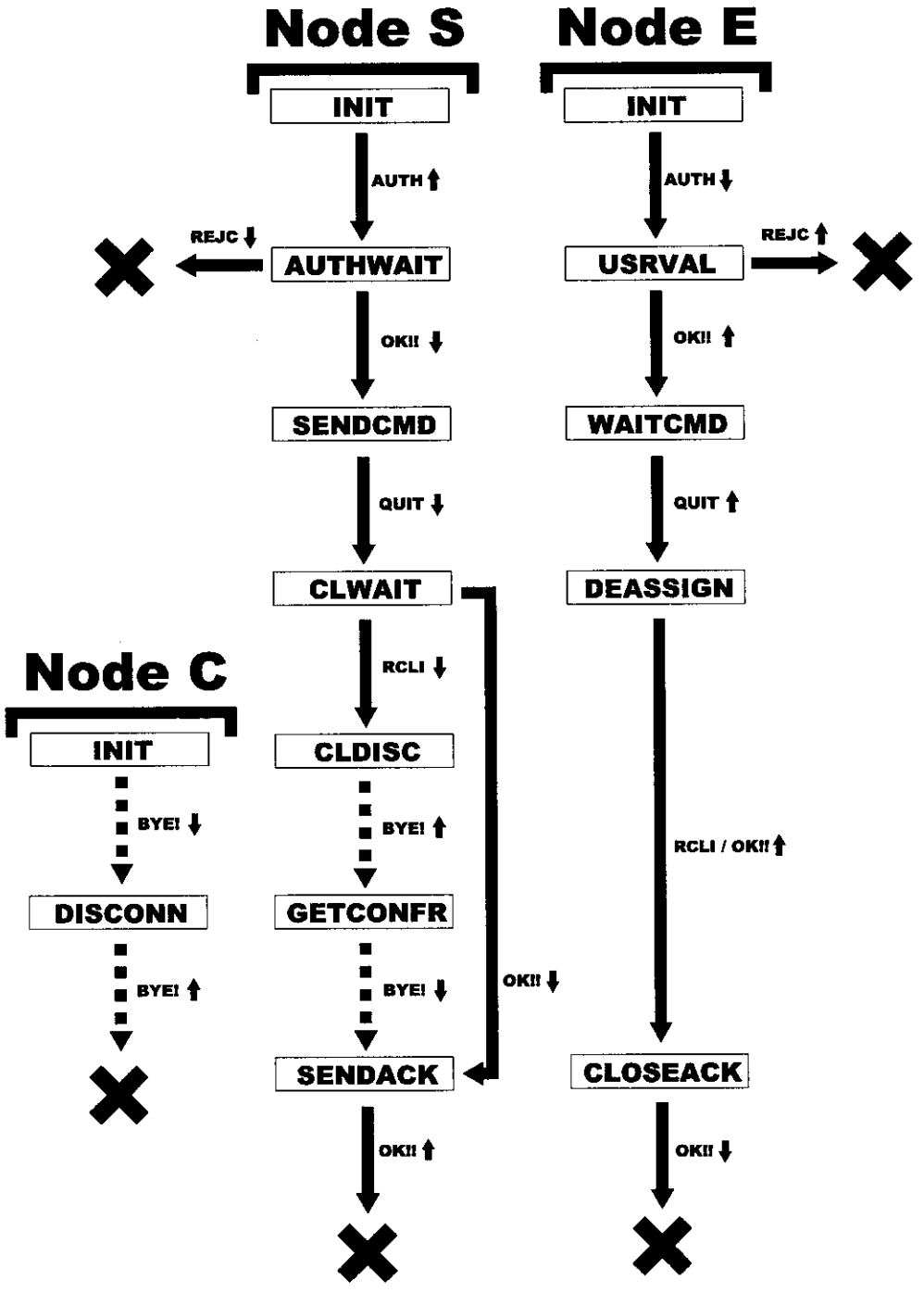


FIGURA 11D

7. Protocol

Si un node estació marxa de la xarxa, aquest ho comunicarà amb el node d'enllaç i desconnectarà el node client al que estava associat.

Quan es desconnecta una estació que ha estat prèviament inserida en el sistema, el principi de la interacció entre ambdós nodes canvia. Ara ja no s'envia un paquet HELO sinó que s'envia un paquet AUTH amb un token d'autenticació que se li ha atorgat al client un cop vol entrar a la xarxa. Aquesta pot ser de qualsevol tipus, per exemple, un valor hash determinat, que és el que s'ha usat en el prototip.

Si el node és reconegut dins de la xarxa aleshores s'enviarà un paquet OK!! i es continuarà amb la interacció. En cas contrari, tal i com passa en els anteriors casos, s'envia un paquet REJC.

Si el node S rep OK!! aleshores enviarà un paquet QUIT de manera que el node enllaç, que coneix a qui està connectat enviarà un paquet RCLI en as que estigui connectat a un client o un paquet OK!! en cas que la estació no s'hagi connectat a cap client.

El paquet RCLI provoca que el node S envii un paquet BYE al node C i aquest li respongui de la mateixa manera moment en que trenquen la seva connexió. Per confirmar aquest fet el node S envia. finalment, un paquet OK!! que serveix tant si el node S estava connectat a una estació com si no ho estava.

7.5.4 Desconnexió d'un node client

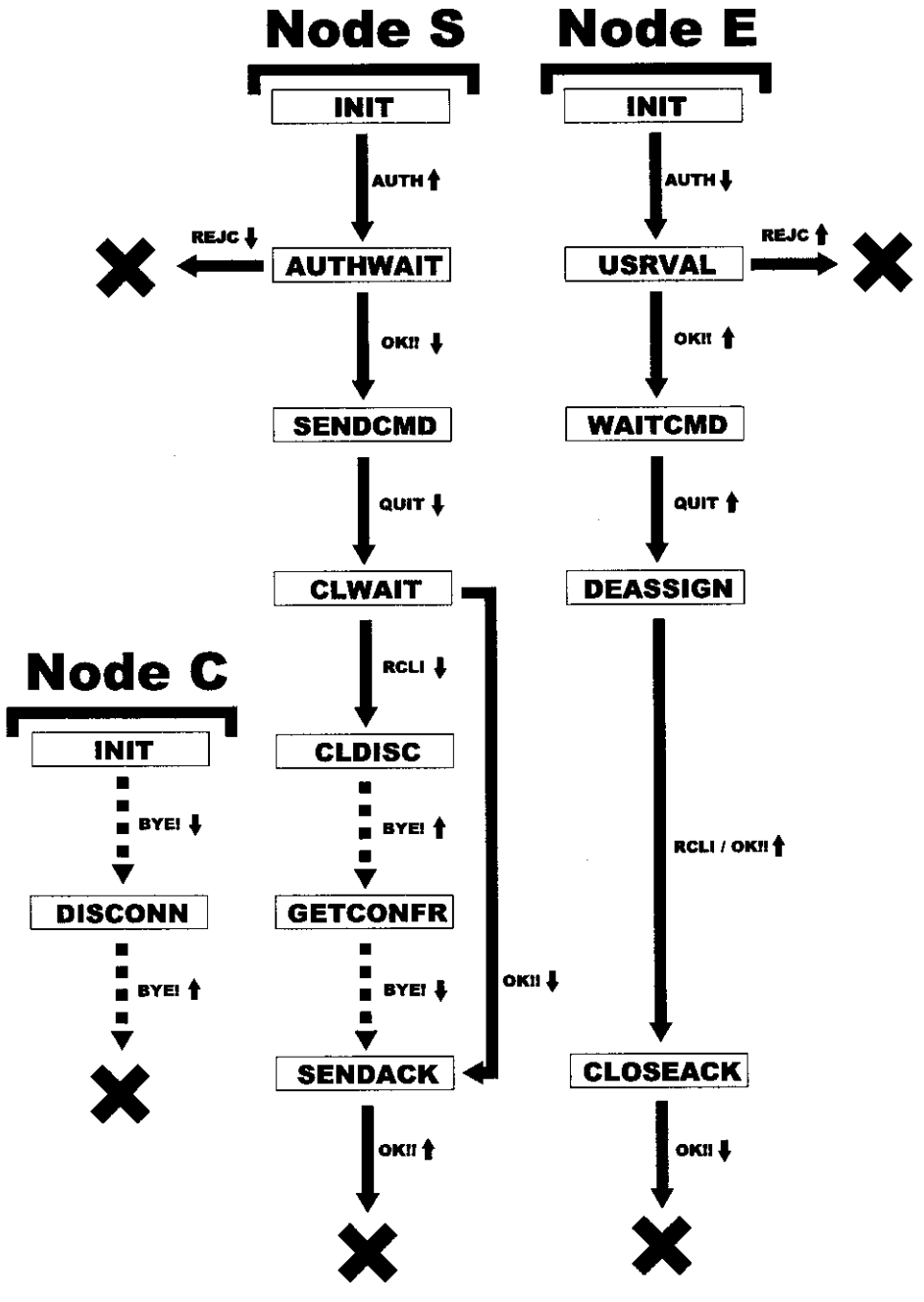


FIGURA 11E

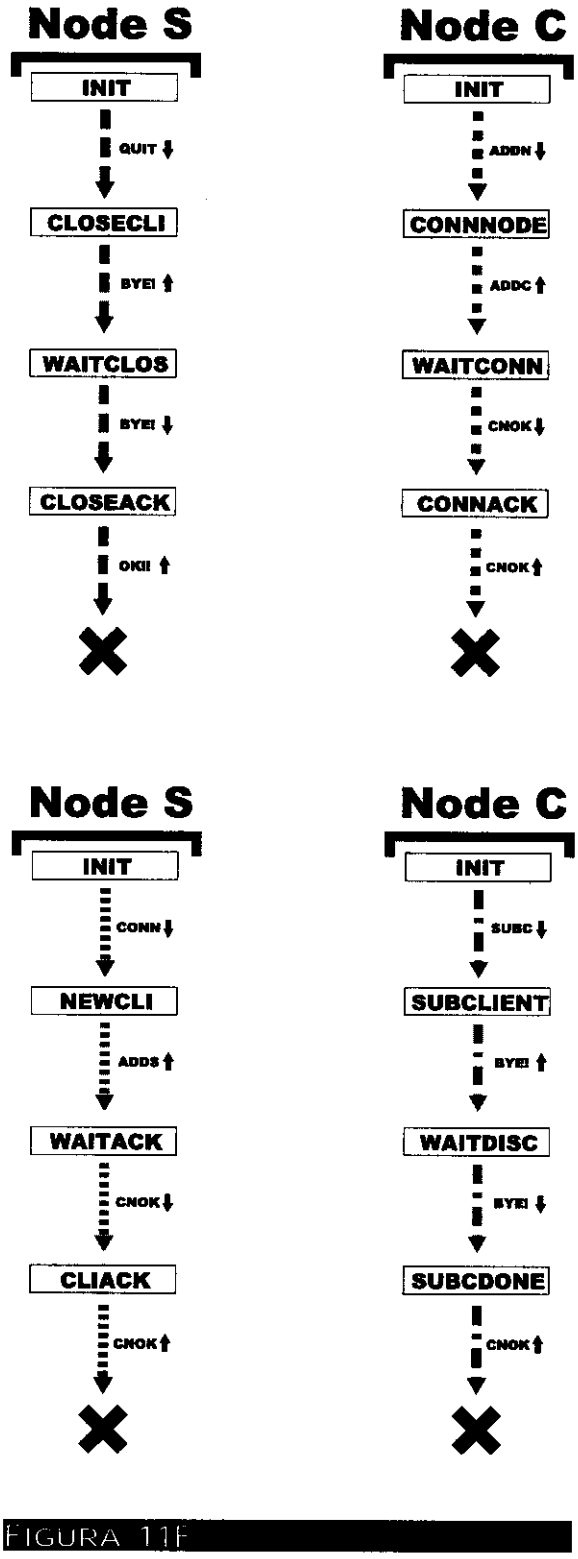

Node S
INIT
QUIT
CLOSECLI
BYE!
WAITCLOS
BYE!
CLOSEACK
OK!!
Node C
INIT
ADDN
CONNNODE
ADDC
WAITCONN
CNOK
CONNACK
CNOK
Node S
INIT
CONN
NEWCLI
ADDS
WAITACK
CNOK
CLIACK
CNOK
Node C
INIT
SUBC
SUBCLIENT
BYE!
WAITDISC
BYE!
SUBCDONE
CNOK

FIGURA 11F

De manera anàloga a l'anterior aquí es descriu la manera en que el client es desconnecta dels nodes S i C als que està connectat.

L'inici de la interacció comença igual que la desconnexió d'una estació: usant les comandes AUTH i retorna comandes OK!! o REJC en funció de si s'ha reconegut, per part de l'enllaç, que el node forma part de la xarxa.

Posteriorment s'envia la comanda QUIT, que és exactament la mateixa que s'ha fet servir en la desconnexió de la estació. Com que el node enllaç coneix el tipus de node en el mateix moment en que s'autentifica no hi ha cap problema per discernir entre quin camí cal seguir.

En aquest punt el node C que ha demanat la desconnexió espera a rebre una comanda BYE! per part del node d'enllaç. El node E es posa a treballar pel seu compte en un bucle de reassignació de connexions i fins que no acaba de realitzar-les no envia la comanda BYE! que el node C espera.

Aquest bucle de reassignació de connexions es divideix en dues parts: la reassignació d'estacions i la reassignació de clients.

7.5.4.1 Reassignació d'estacions

La reassignació d'estacions correspon al bucle dret representat en el graf d'estats del node E. Primer es desconnecta la estació, realitzant una nova connexió a la mateixa (la línia de guionets que correspon al graf superior esquerra de la figura 11F) i enviant una comanda QUIT a la estació connectada al node C, aleshores la estació envia per la connexió que té amb el node C una comanda BYE! que li és resposta pel node C (la part en gris d'aquest graf). Posteriorment s'avisa al node E que la estació s'ha desconnectat del client amb una comanda OK!!.

Posteriorment, sempre que hagi algun client disponible es reassignarà la estació a aquest altre client. Aquesta part correspon al graf d'estats inferior esquerra de la figura 11F. S'envia un paquet CONN a la estació amb el node al que es vol que s'associï i aquesta crea una nova connexió al node C (simbolitzada en gris). La estació enviarà un paquet ADDS al node C i esperarà la confirmació del node (CNOK). Això provocarà enviar un CNOK del node S al node E per tal de notificar la nova connexió.

Si no existís cap node C a qui associar l'estació aquests dos passos s'ignoren.

7.5.4.2 Reassignació de clients

La manera en que es reassignen els clients és anàloga a la de les estacions.

El node enllaç coneix la llista de veïns del node C i procedirà a connectar els seus veïns entre ells prèviament a desconnectar-los el node C. Això vol dir que momentàniament la

associativitat d'un node serà major que la que s'ha assignat al node E. En ser un increment d'una sola unitat fa que l'impacte sobre l'ample de banda no sigui significatiu.

Si considerem una llista de veïns del node C, el primer que es farà serà associar el primer veí de C amb el segon veí de C, el segon amb el tercer, etc... Per tal d'associar-ho el node enllaç enviarà una comanda ADDN a un client C1 (graf d'estat superior dreta de la figura 11F). Aquest crearà una connexió a un altre veí de C (s'anomenarà C2) a qui el vulguem connectar, de manera que aquest enviarà un ADDC per tal de crear un nou vincle entre C1 i C2. Per tal de confirmar la connexió el node C2 enviarà un CNOK a C1 que a la vegada confirmarà al node enllaç amb un altre paquet CNOK.

Si per qualsevol motiu no tingués cap veí aquest node s'ignoraran aquests passos.

Un cop tenim els veïns connectats entre ells s'anirà desconnectant el node C dels veïns. Això es realitza mitjançant la comanda SUBC (graf d'estats inferior dreta), que s'envia al node C de manera que aquest realitzarà una nova connexió (en gris) cap al veí C1 tancant la connexió amb la comanda BYE!. El node C1 respondrà amb una comada simètrica i això permetrà que el node C confirmi la seva desconnexió al node d'enllaç amb un paquet CNOK.

7.5.4.3 En resum

Cal destacar que en cap cas se li diu al node C que tregui ell mateix les connexions cap els altres nodes: sempre es fa de manera inversa, és a dir, són els altres nodes els que es

reassignen cap a altres clients i decideixen tancar la connexió amb el node C que vol sortir de la xarxa.

El motiu per fer això és assegurar-se tant com sigui possible que la xarxa quedi coherent abans que marxi aquest node: el node E, que és qui coneix la topologia de la xarxa és qui manarà a tots els nodes connectats al client que vol marxar la seva reassignació, de tal manera que coneix en tot moment la nova topologia de la xarxa, pas a pas.

7.6 Els paquets DATA

Els paquets DATA són els únics que no afecten a la topologia de la xarxa ni requereixen cap confirmació. En ells viatgen les observacions codificades en format BUFR. Són, per tant, paquets de mida i contingut molt variable.

7.7 Bibliografia i referències web

1: Article sobre el protocol FTP [Wikipedia]

http://en.wikipedia.org/wiki/FTP

2: Article sobre el protocol HTTP [Wikipedia]

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

8 Implementació

8.1 Metodologia

La implementació del prototip s'ha fet de la manera més senzilla possible i començant des de avall cap a dalt. És a dir, començant per dissenyar les classes bàsiques sobre les que es sustentarà els programes i unint-les per a crear estructures més complexes que permetin modelitzar els grafs.

8.1.1 Disseny d'avall cap a dalt

Un altre motiu per fer la implementació d'avall cap a dalt ha estat la reutilització de components. S'ha volgut crear components que siguin fàcilment usables entre les diverses parts que componen el Sistema Meteorològic Distribuït independentment del llenguatge de programació en la que calgui implementar aquest component. Per tant primer s'ha dissenyat aquests components i posteriorment s'han afegit les característiques pròpies de cada tipus de node.

8.2 Diagrama UML bàsic

El diagrama de classes UML bàsic conté la llista de classes que comparteixen entre les diferents parts que componen el sistema.

En la figura 12A es mostra aquest diagrama:

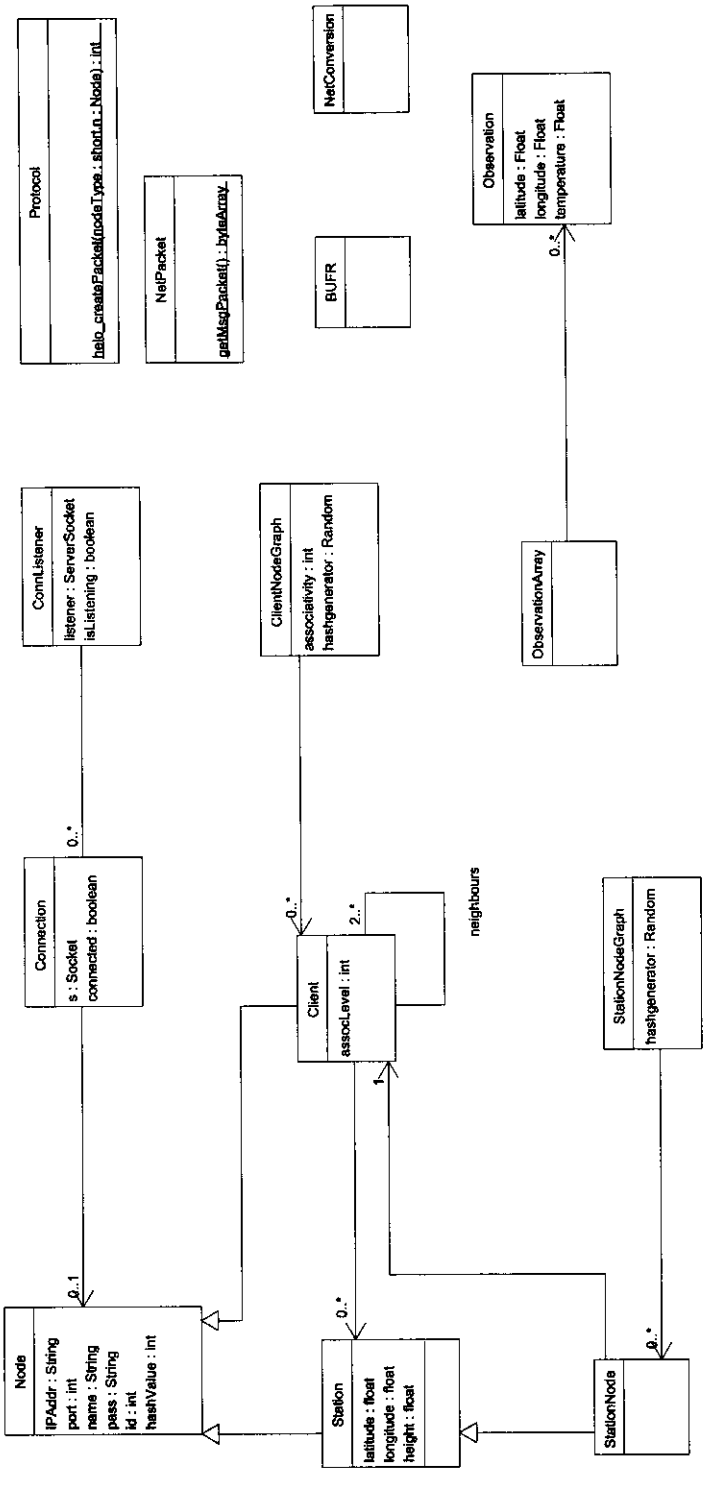

Protocol
hello_createPacket(nodeType : short,n : Node) : int
NetPacket
getMsgPacket() : byteArray
NetConversion
BUFR
Observation
latitude : Float
longitude : Float
temperature : Float
ConnListener
listener : ServerSocket
isListening : boolean
ClientNodeGraph
associativity : int
hashgenerator : Random
ObservationArray
Connection
s : Socket
connected : boolean
Client
assocLevel : int
neighbours
StationNodeGraph
hashgenerator : Random
Node
IPAddr : String
port : int
name : String
pass : String
id : int
hashValue : int
Station
latitude : float
longitude : float
height : float
StationNode
0..*
0..1
1
2..*

FIGURA 12A

8.2.1 Classes bàsiques

En aquest diagrama de classes es pot apreciar com la classe bàsica sobre la que s'ha muntat moltes altres és la classe Node. Per sobre d'aquesta classe s'afegeix una classe Station que modelitza un node de tipus S, que es qui proveirà de dades. D'aquí els atributs de localització geogràfica que posseeix.

8.2.2 Modelització del graf

La manera en que es modelitza el graf de nodes de la xarxa en UML es mitjançant l'ús de classes que es comporten com un array i usant les associacions recursives.

La classe Client té una associació recursiva amb ella mateixa anomenada "neighbours" que no són més que el nombre de veïns que estan connectats a un node. El fet que existeixi un atribut dins de la classe Client indicant la associativitat és amb motiu de que possibles implementacions futures contemplin associativitats dinàmiques: és a dir, diferent màxim nombre de veïns segons el node.

La classe ClientArray és qui modelitza el nombre total de clients que hi ha al sistema. No serveix per a guardar la seva topologia.

La classe StationNode és la que modelitza un node S en la xarxa, la seva associació "0..1" amb client indica si la estació està connectada o no a un node client. Tal i com passa amb la classe Client, també hi ha una classe anomenada StationNodeGaph que hostatja el

conjunt d'estacions que hi ha al sistema.

8.2.3 Les observacions

Les observacions s'emmagatzemen en un objecte Observation. Per motius de claredat no s'han descrit tots els atributs d'aquesta classe, però el que contindrà són un conjunt d'atributs que modelitzaran una sola observació. En la implementació del prototip s'assigna un objecte Observation per cada estació i s'actualitzen els seus atributs cada cop que es rep una observació provinent de la estació associada.

Per a seguir amb el mateix patró de disseny s'ha implementat una classe ObservationArray que conté les observacions de cadascun dels nodes estació que envia paquets amb observacions a la xarxa.

8.2.4 Les classes d'utilitat

Classes com BUFR, NetPacket, Protocol o NetConversion són classes d'utilitat i estan pensades per a contenir mètodes estàtics que ajudaran a fer una tasca determinada. En el cas de BUFR en la part Java serà llegir paquets BUFR i decodificar-los deixant les seves dades dins de la classe Observation.

NetPacket, Protocol o NetConversion ajuden a crear paquets (cas de la classe Protocol) o obtenir-los (cas de la classe NetPacket).

NetConversion és una classe auxiliar necessària per a transformar valors que venen en arrays de bytes al format natiu del llenguatge de programació. Per exemple, la mateixa codificació de un float en C++ és fa canviant el pes de cadascun dels bytes en Java, d'aquí que es requerís una classe que resolgués aquest tipus de conversions.

8.3 El node estació

A la implementació del node estació només van caldre crear unes classes bàsiques a afegir a les classes base de que es disposen.

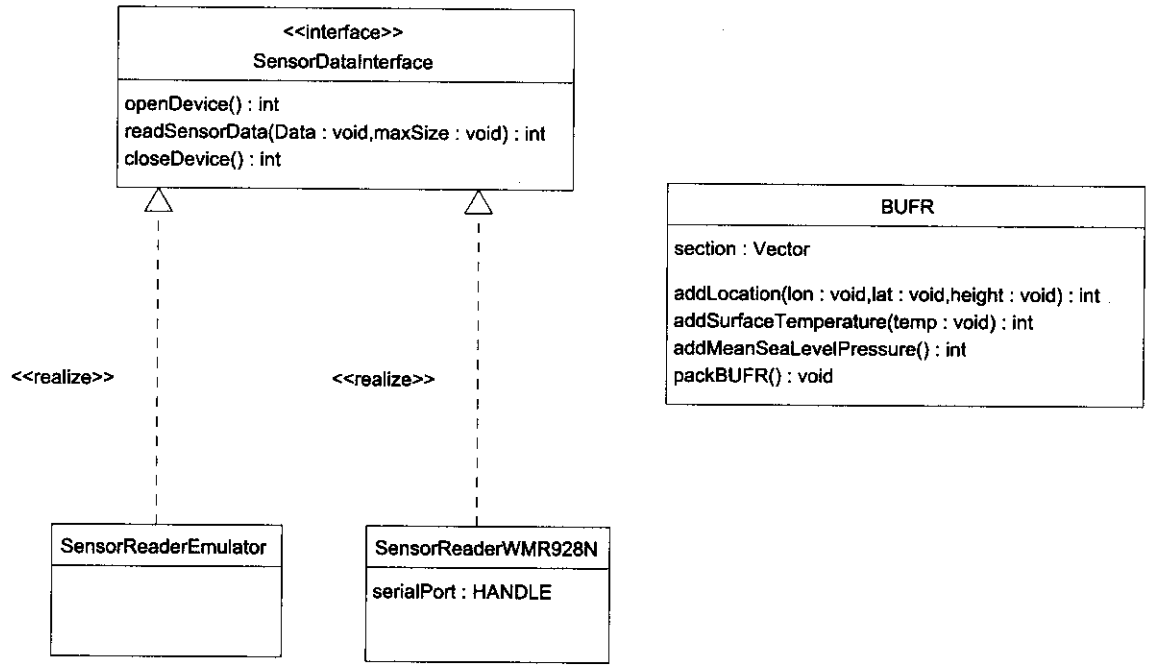


FIGURA 12B

S'ha implementat una interfície que permeti accedir de manera uniforme a qualsevol dels dispositius i que sigui independent de la arquitectura. Per al prototip s'ha fet servir la classe SensorReaderEmulator que emula la manera en que una estació retorna dades: a un cert interval de temps retorna una observació que envia per l'interface al que està connectat. Degut a que disposo d'una estació meteorològica professional Oregon Scientific WMR928 es va programar un test per a posar a prova la interfície fent una implementació que llegís certes dades usant els mètodes descrits. El resultat va ser positiu, tot i que s'haurien d'estudiar altres interfícies (com les USB), més habituals a dia d'avui.

També s'ha implementat una classe BUFR que conté un conjunt de mètodes que codifica els valors que se li passen dins d'un paquet BUFR. Conforme es criden als mètodes per afegir dades, "addSurfaceTemperature" per exemple, s'afegeixen les dades en el paquet fins que es crida a packBUFR que retorna un buffer amb el conjunt de dades codificades en format BUFR i llest per enviar per xarxa.

No s'ha tingut cap inconvenient en el fet que es pugui contenir diferents valors per a una mateixa categoria, és a dir, es pot trucar a la funció addSurfaceTemperature tantes vegades com sigui necessari i amb tants valors com calgui, que cadascun d'aquests valors s'anirà afegint al paquet BUFR sense sobreescriure'n cap. l'estàndard BUFR permet aquest fet.

8.4 El node enllaç

El node d'enllaç només té dues classes: una per al programa principal anomenada Link i una altra per cada connexió anomenada LinkThread.

La classe Link és un bucle que espera connexions. Per cada connexió que rep, en aquest cas d'un node S o d'un node C, crea un nou thread que realitza la seva tasca tot seguint el graf d'estats descrit dintre del capítol de Protocol.

La classe LinkThread interactua amb les classes StationNodeGraph i ClientNodeGraph donat que aquestes tenen el conjunt de nodes que hi ha al sistema i la seva topologia, i l'únic node que coneix la topologia de la xarxa és el node d'enllaç.

8.5 El node client

El node client té una model de threads similar al node. Hi ha una classe Client que escoltarà peticions i es connectarà amb els veïns i les estacions que corresponguin. Per a cada connexió amb qualsevol altre node que no sigui l'enllaç s'usarà un thread nou.

Cada thread està modelat per la classe ClientThread. La responsabilitat d'aquesta classe és repartir els paquets de nodes DATA amb altres nodes i obtenir les observacions de cada paquet data per a emmagatzemar-les en un objecte Observation corresponent a cada node S que les envia.

8.5.1 Model de threading del node client

Cada cop que una estació es connecta a un client, o quan un client es connecta a un altre client es crea un thread. Cada thread manté una connexió a un node, per tant si es volen disseminar dades caldrà que tots els threads coneguin totes les connexions als altres nodes.

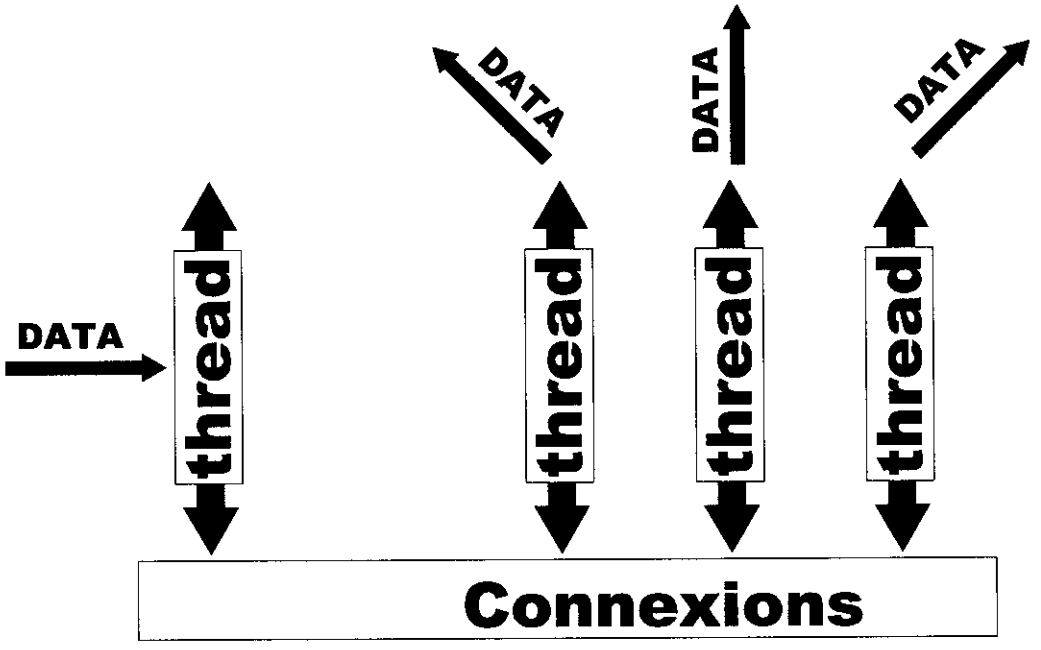


FIGURA 12D

La figura 12D il·lustra aquest fet. Els diversos threads comparteixen un objecte que conté totes les connexions a cadascun dels nodes. Quan un dels threads rep un paquet DATA es comproven quantes connexions, que no sigui la mateixa connexió per on ha vingut el paquet DATA, hi ha disponibles en el moment en que es rep el paquet. Com que el canal

de sortida no és bloquejant un mateix thread accedirà al conjunt de connexions dels altres threads i podrà enviar la seva informació per aquest canal.

8.5.2 La visualització de les dades

La visualització de les observacions es basa en bolcar tota la informació continguda en els objectes Node a un format.

S'ha dit prèviament que finalment s'ha escollit OpenLayers per a realitzar la representació de les observacions sobre un mapa. Cal destacar que OpenLayers és una llibreria javascript i que, per tant, s'ha d'executar en un navegador. Per tant l'objectiu a tenir present ha estat què s'ha de generar per tal que apareguin les dades per pantalla sobre un mapa.

OpenLayers no està encara molt desenvolupada en l'aspecte de mostrar text sobre un mapa sinó que està més orientada a mostrar imatges. D'aquí han sorgit alguns problemes que es comentaran més endavant.

Per tal de generar un mapa amb valors finalment s'ha optat per generar un fitxer javascript usant plantilles, és a dir, petits fragments de codi que concatenats creen el mapa amb els valors. Per tant únicament caldrà usar un fitxer en format HTML que inclogui la llibreria javascript amb el fitxer generat i refrescant el navegador es podrà veure l'últim resultat generat pel node client. En la implementació del prototip es genera aquest fitxer cada cop

que es rep un paquet DATA amb observacions.

La visualització que s'obtindrà està basada en mostrar cinc categories diferents, que són les que s'han implementat per a aquest prototip. Aquestes són: temperatura, humitat, pressió reduïda a nivell de mar, vent i intensitat de precipitació.

8.6 Problemes trobats

A continuació es descriuen els problemes que s'han trobat en relació amb el desenvolupament de prototip classificats en les categories corresponents.

8.6.1 Problemes amb la xarxa

Un dels problemes principals ha estat la mescla entre C++ i Java. La diferència de velocitat d'execució entre ambdós llenguatges ha estat un factor molt important: sobretot en l'aspecte de comunicació de xarxa.

Quan un dels nodes S (implementat en C++) envia informació a un node C (implementat en Java) i tanca la via de comunicació era molt habitual que el node C indiqués error: a aquest node C li arribava abans l'esdeveniment de connexió tancada que no pas la informació motiu pel qual s'ha hagut d'optar per introduir un temps d'espera en algunes comunicacions entre nodes C++ i Java.

El propi fet de solucionar problemes amb el protocol requereix l'ús de programes de xarxa avançats, com sniffers, per tal d'examinar els paquets que viatgen per la xarxa. Com tota la xarxa es munta de manera local i la quantitat de paquets és tan gran com major és la xarxa, l'anàlisi de les causes d'un error quan hi ha un nombre de nodes elevat és dificultós. Per tant s'ha de dissenyar, implementar i provar amb molta cura en models petits de tal manera que la extrapolació no afegeixi problemes més endavant.

També cal comentar que el sistema de protecció de trojans del Windows XP no ha posat les coses fàcils quan s'usen de manera intensiva molts sockets: cal obrir-los poc a poc de manera que el sistema no quedi identificat com codi maliciós.

8.6.2 Problemes amb els threads

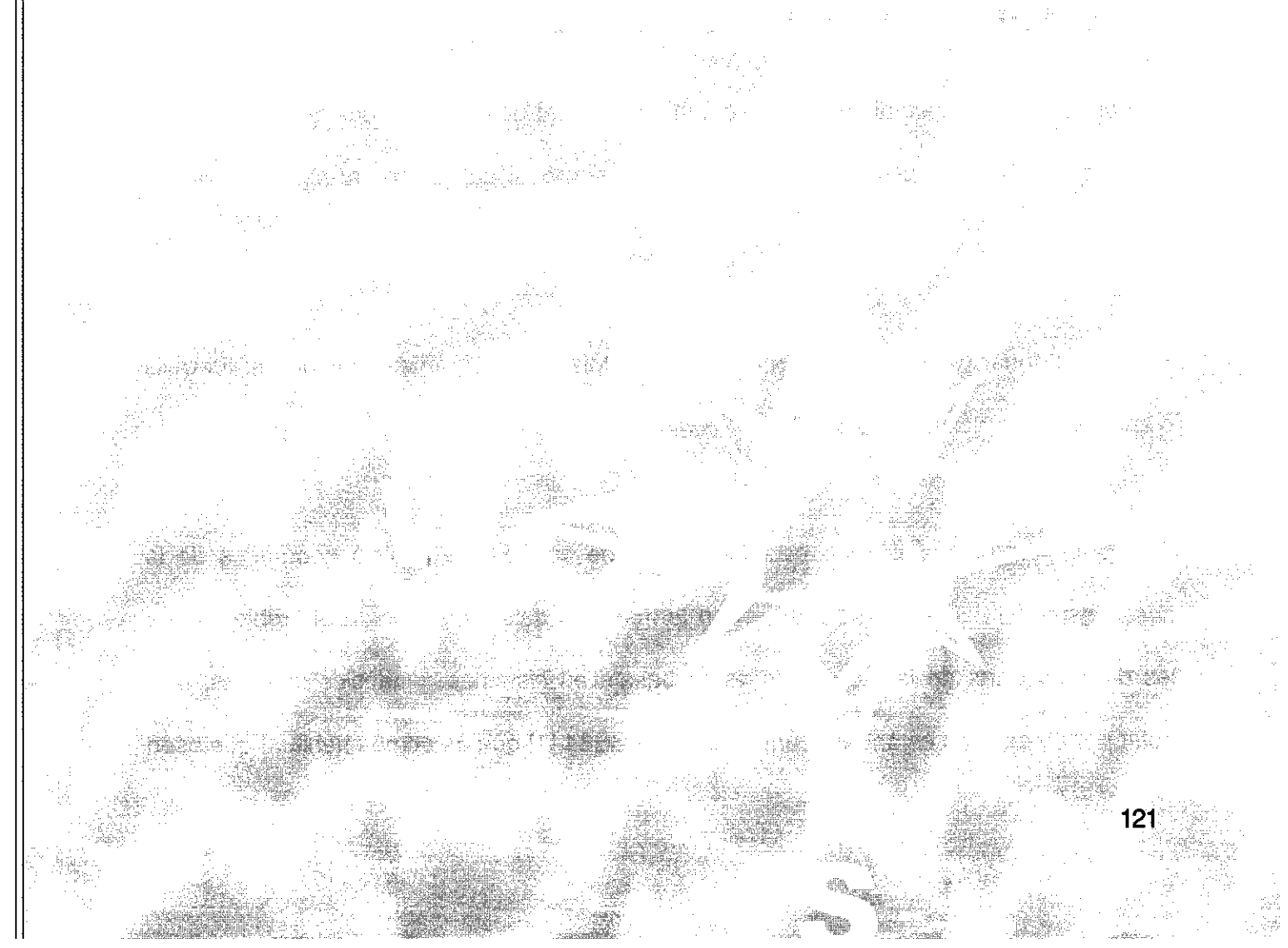
Degut al fet que les proves s'han fet de manera local el elevat nombre de threads presents en el sistema en les proves amb una gran nombre de nodes suposa una problemàtica extra. Les condicions de carrera són molt habituals i cal programar amb molta cura protegint el codi amb mutex de tal manera que les operacions s'executin en l'ordre establert.

8.6.3 Problemes amb la visualització

OpenLayers permet fer de manera molt simple coses realment complexes. El problema de l'OpenLayers esdevé quan s'usen funcionalitats que s'han fet en les últimes versions de la llibreria. Un dels casos on s'ha trobat problemes és en el fet que la llibreria que ofereixen

per a descarregar en la seva plana principal no permet fer determinades accions que sí permeten fer la llibreria que es pot descarregar en certes versions de desenvolupament. Per tant un dels problemes esdevé la elecció de la llibreria OpenLayers, fins i tot dintre de una mateixa versió (concretament la 2.7) segons quina s'usi els valors no apareixeran en el mapa.

9 Resultats



9.1 Funcionalitats aconseguides

S'ha aconseguit realitzar tota la implementació del prototip exposada anteriorment a excepció d'un cas, que per qüestió de temps no s'ha implementat. El cas en concret és la reassignació d'una estació meteorològica a un altre client quan un client marxa de la xarxa.

Aquest fet no ha impedit obtenir una mínima funcionalitat que ha permès avaluar alguns aspectes, sobretot els relacionats amb la implementació.

9.2 Com usar el prototip

El prototip té tres executables. Un executable en format natiu (.exe) i dos que són classes de Java. Prèviament a usar el prototip de manera local cal assegurar-nos de diverses coses.

Primer cal donar permisos per a poder executar tots tres programes, configurant firewalls o antivirus o desactivant-los si cal.

Segon, si arrenquem el node estació haurem de conèixer la posició geogràfica de la nostra estació i aquesta ha d'estar en un format molt determinat. Degut al problema que les diverses parts de la cartografia estan basats en sistemes geodèsics diferents, s'ha optat per un sistema estàndard i universal: el WGS84 [1] que és el més habitual que usen

els GPS.

Si no es disposa d'una GPS però es vol obtenir les coordenades es pot optar a usar el Google Earth [2] que dona les dades directament per a ser usades en el programa del node estació, tant d'longitud i de latitud com de altura,

9.2.1 Línia de comandes

A continuació es descriuen els paràmetres que s'usen en cadascun dels programes.

9.2.1.1 Node S

L'executable natiu, el del node S, té set paràmetres:

1. Longitud on està ubicada la estació en graus seguint el sistema WGS84.
2. Latitud també en el sistema WGS84
3. Altura en mestres sobre el nivell de mar.
4. Adreça IP local
5. Port TCP local
6. Adreça IP del node enllaç
7. Port TCP del node enllaç

9.2.1.2 El node enllaç

Només requereix dos paràmetres:

1. El port per on escoltarem les peticions

2. La associativitat màxima permesa per node

9.2.1.3 El node client

El node client requereix quatre paràmetres:

1. La adreça IP local
2. El port TCP local
3. La adreça de l'enllaç
4. El port de l'enllaç

9.3 Traces del prototip

A continuació s'ofereixen traces del funcionament del prototip. S'ha pres com a mostra una xarxa mínima, amb un node d'enllaç, un client i una estació que s'ha ubicat a Barcelona.

9.3.1 Node d'enllaç

Primer cal arrencar el node d'enllaç

```
java -cp .;..\javalib Link 20200 3

SMD Link Node v0.1

  + startLinkNode :: init
  + Obrint un listener
    + Creant un nou ServerSocket en el port 20200
    + S'ha obert el port : 20200
  + Creant graf de nodes de clients (asoc = 3 )
    + S'ha creat el graf de nodes 'Client'
    + S'ha creat el graf de nodes 'Station'
    + S'ha creat el hook pel Ctrl+C
  + Esperant a una connexio
    + Connectat! Configurant connexio a 127.0.0.1:1609
    + Nova connexio
```

```
  + Thread de connexio
  + Inicialitzant els parametres
  + Esperant a una connexio
  + INIT : Thread de connexio
  + Funcio newNode
    + HELO : Client
  + newClientNode
    + Creant un nou node client
    + Inserint el node client
  + Afegint un nou node al graf 127.0.0.1:20300
    + Cercant el primer espai lliure
    + No hi ha espai lliure, cercant posicio
    + Posicio retornada : 0
    + Cercant vei disponible
  + Cercant un vei lliure
    + No s'ha trobat un vei lliure
    + Afegint el node a la estructura
    + El node no s'associa a cap vei
    + ACCP amb id : 0 hash : facace50
    + Connectant als veins
    + No te veins: enviant paquet IDLE
    + IDLE enviat correctament
    + Connectat! Configurant connexio a 127.0.0.1:1610
    + Nova connexio
  + Thread de connexio
  + Inicialitzant els parametres
  + Esperant a una connexio
  + INIT : Thread de connexio
  + Funcio newNode
    + HELO : Station
  + newStationNode
127.0.0.1:20100
    + Afegint estacio al graf de nodes
  + Afegint una estacio a la llista d'estacions
    + Cercant index del array
    + Posicio retornada : 0
    + Afegint el node dins de l'array
    + Inserit correctament
    + ACCP amb id : 0 hash : facace50
    + ACCP enviat
    + Assignant un client
    + Connectant a un client de la xarxa
    + Trobat el client id : 0
    + Correctament unida la estacio i el client
  + Assignant la estacio al node client
    + Afegint la associacio amb el client id : 0
  + Assignant al client id : 0 a un node estacio
    + Afegint el node estacio id : 0
    + Connectat! Configurant connexio a 127.0.0.1:1612
    + Nova connexio
  + Thread de connexio
  + Inicialitzant els parametres
  + Esperant a una connexio
  + INIT : Thread de connexio
  + AUTH : s'esta autentificant un node
    + Node estacio autentificat
    + Enviant paquet de confirmacio OK!!
    + En espera de ordres del node connectat
    + Rebut QUIT
```

```
+ quitStation
  + Esborrant el client associat
  + Esperant OK!! per part de la estacio
  + OK!! Rebut
  + Esborrant la estacio
+ Treient el node estacio de l'array
  + Obtenint la estacio associada
  + Esborrant el client associat
+ Treient el client associat a la estacio
  + Esborrant el client associat id : 0
+ Trient la estacio del client id : 0
  + Es desassignara el node estacio id : 0 d'aquest client
  + Esborrant la estacio de l'array
- quitStation
  + Connectat! Configurant connexio a 127.0.0.1:1614
  + Nova connexio
+ Thread de connexio
+ Inicialitzant els parametres
+ Esperant a una connexio
+ INIT : Thread de connexio
+ AUTH : s'esta autentificant un node
  + Node client autentificat
  + Enviant paquet de confirmacio OK!!
  + En espera de ordres del node connectat
  + Rebut QUIT
+ quitClient (id : 0)
  + No hi ha cap estacio connectada
  + No hi ha cap client connectat
  + Eliminant el client
+ Esborrant un node del graf de clients
  + S'ha trobat el client amb id : 0
  + El node no te cap vei associat
  + Eliminant el node del graf
  + Esborrant la estacio de l'array
- quitStation
  + Connectat! Configurant connexio a 127.0.0.1:1614
  + Nova connexio
+ Thread de connexio
+ Inicialitzant els parametres
+ Esperant a una connexio
+ INIT : Thread de connexio
+ AUTH : s'esta autentificant un node
  + Node client autentificat
  + Enviant paquet de confirmacio OK!!
  + En espera de ordres del node connectat
  + Rebut QUIT
+ quitClient (id : 0)
  + No hi ha cap estacio connectada
  + No hi ha cap client connectat
  + Eliminant el client
+ Esborrant un node del graf de clients
  + S'ha trobat el client amb id : 0
  + El node no te cap vei associat
  + Eliminant el node del graf
```

9.3.2 Node estació

En arrencar la estació el programa indica això:

```
station.exe 2.168047 41.389275 38 127.0.0.1 20100 127.0.0.1 20200

SMD Station Node v0.1

  + HELO :: protocol
    + HELO :: mida : 22
  + ACCP / REJC :: protocol
    + S'ha rebut correctament el paquet ACCP/REJC
    + ACCP amb id : 0 hash : facace50
  + IDLE / CONN :: protocol
    + S'ha rebut correctament el paquet IDLE / CONN
    + CONN
  + ADDS :: protocol
  + CNOK :: recv :: protocol
  + CNOK :: send :: protocol
Nou paquet BUFR
+ Afegint localització
  + Inici del thread de connexio
  + openListener
  + Thread de connexio: escoltant peticions
    + Escoltant connexions
  + Noves Dades meteorologiques
  + Afegint temperatura
  + Enviant dades
+ Afegint localització
  + Noves Dades meteorologiques
  + Afegint temperatura
  + Enviant dades
+ Afegint localització
  + Noves Dades meteorologiques
  + Afegint intensitat de precipitaci≤
  + Enviant dades
+ Afegint localització
  + Noves Dades meteorologiques
  + Afegint humitat
  + Enviant dades
+ Afegint localització
  + Noves Dades meteorologiques
  + Afegint temperatura
  + Enviant dades
+ Afegint localització
  + Noves Dades meteorologiques
  + Afegint intensitat de precipitaci≤
  + Enviant dades
  + AUTH :: protocol
  + OK!! / REJC :: protocol
    + S'ha rebut correctament el paquet OK!! / REJC
    + OK!!
  + QUIT :: protocol
  + OK!! / RCLI :: protocol
    + S'ha rebut correctament el paquet OK!! / RCLI
    + RCLI
```

```
  + BYE! :: protocol
  + BYE :: recv :: protocol
  + OK!! :: protocol
  + BYE! :: protocol
    + (connThread) Rebut BYE!
```

9.3.3 El node client

El mateix moment el node client indicava:

```
java -cp .;..\javalib ClientUser 127.0.0.1 20300 127.0.0.1 20200

SMD Client Node v0.1

  + Inicialitzant el node client
  + Connectant a la xarxa
    + Connexio oberta
    + HELO :: Enviant
    + Esperant ACCP / REJC
    + ACCP amb id : 0 hash : facace50
    + Esperant IDLE / CONN
    + Rebut IDLE
    + S'ha creat el hook pel Ctrl+C
  + Obrint un listener
    + Creant un nou ServerSocket en el port 20300
  + Esperant a una connexio
    + Connectat! Configurant connexio a 127.0.0.1:1611
  + clientMain: se'ns han connectat
    + ADDS rebut
    + Enviant CNOK
  + Esperant a una connexio
  + Inici del thread de connexio amb el id: -1
    + DATA station : 0
    + DATA station : 0
    + DATA station : 0
    + DATA station : 0
    + DATA station : 0
    + DATA station : 0
    + BYE! al vei id : -1
  + Fi del thread de connexio amb el id: -1
  ! Sortint de la aplicacio
    + Enviant AUTH
    + Rebent autoritzaci≤
    + OK!!
    + QUIT : Enviant desconnexio de la xarxa
    + Esperant rebre : BYE!
    + BYE! rebut
    + Enviant BYE!
    + BYE! enviat
```

9.3.4 Resultat gràfic de la execució

Amb la traça vista aquí el resultat ha estat el que es pot veure a la figura 13A. En aquesta figura només s'ha activat una sola capa la de la pressió reduïda a nivell atmosfèric o MSLP (Mean Surface Level Pressure).

Es pot veure un menú de capes ocultable on hi ha les diverses capes en que s'han categoritzat les dades meteorològiques que es reben i la observació de la pressió a Barcelona, que en aquest cas és de 1018 hectoPascals (o també mil·libars).

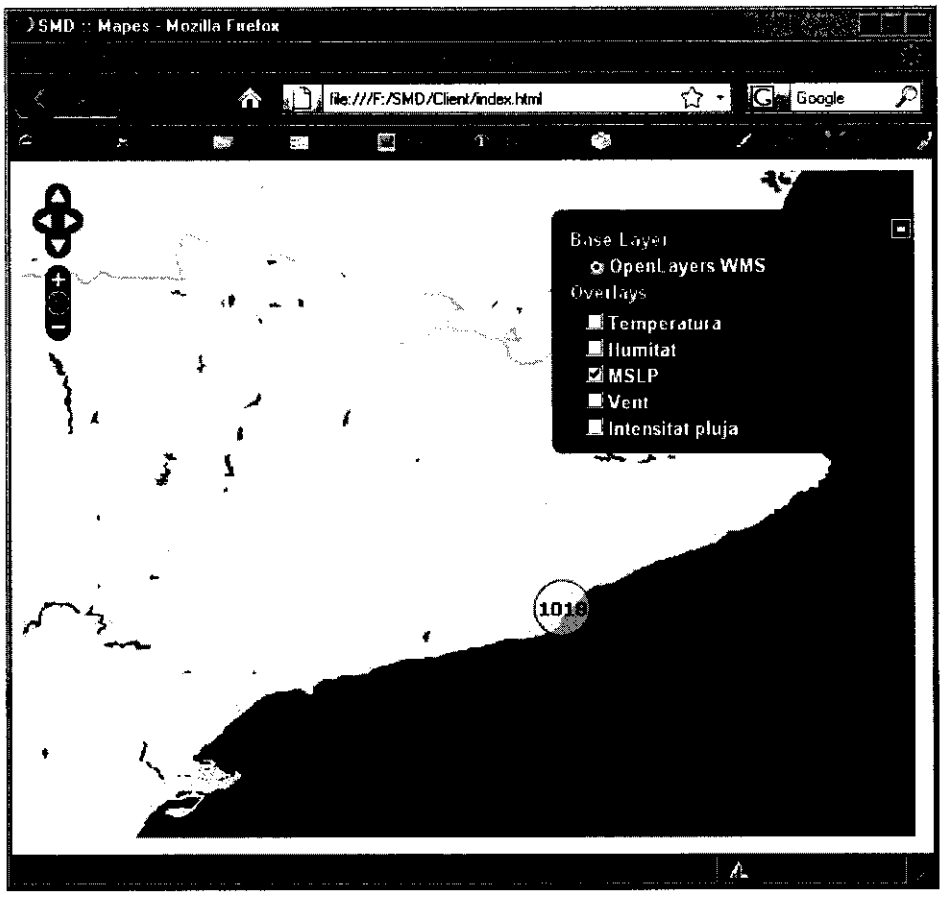


FIGURA 13A

9.5 Proves de rendiment

Per tal d'esbrinar com funcionaria en major escala aquest sistema s'ha provat a llençar més estacions gradualment, que generessin més dades i en conseqüència més tràfic i més activitat dels threads. També s'ha optat per llençar tants nodes client com siguin possibles per tal de mesurar la càrrega que originen aquests.

9.5.1 El hardware

El hardware sobre el que s'ha provat el sistema compren un ordinador de fa quatre anys amb els següents components:

Processador Athlon XP 2000+ a 1,666 Ghz (velocitat de stock)

1 GB de memòria RAM DDR a 333 Mhz (166 x 2)

Disc dur de 120 GB Serial ATA

Targeta gràfica ATI Radeon 9600 Pro

Hi ha més components hardware instal·lats al sistema però no es creu que afectin el rendiment de la aplicació. Donat l'actual ritme d'actualització d'ordinadors aquest sistema es considera bàsic per als estàndards d'avui: molts portàtils de gama baixa i mitja incorporen processadors o memòria molt més ràpids i amb una arquitectura hardware superior.

9.5.2 Característiques de la simulació

En la simulació hi ha una sèrie de factors comuns independents de la càrrega de nodes pròpia de la simulació. El factor principal, el sistema operatiu ha estat un Windows XP SP3 amb la major part de serveis apagats.

Per a simular una estació meteorològica de la manera més real possible s'ha imitat el comportament d'una estació professional, en concret, el model de comportament a imitar s'ha basat en la estació Oregon Scientific WMR928N. Aquesta estació és connecta a l'ordinador mitjançant el port sèrie i envia observacions de entre 5 i 16 bytes en intervals aproximats de 5 segons.

L'emulador ha imitat aquest comportament retallant la mida de les observacions entre 2 i 4 bytes, per eliminar la redundància.

Tan aviat es rep una observació el node S la codifica en format BUFR i la envia al node C al que estigui connectat. La mida dels paquets BUFR que viatgen per la xarxa és d'aproximadament 70 bytes, en funció del tipus d'observació que conté.

El comportament dels clients és sempre igual: tan aviat es rep un paquet DATA el primer que es fa és enviar-los als veïns als que estigui connectat i posteriorment decodificar-lo en un objecte Observation i posteriorment escriure totes les observacions a disc.

9. Resultats

En tots els casos en que s'ha provat la associativitat sempre ha estat de 3. Cal tenir en compte que quan es fan càlculs sobre la xarxa sempre s'estima per sobre de la capacitat real: es suposa que tots els nodes C tenen 3 veïns (per l'efecte de l'associativitat) quan els que estan a l'extrem del graf possiblement tinguin només 1.

9.5.2 Càrrega de 10 Nodes

S'ha procedit a fer una simulació amb 10 nodes de cada tipus, és a dir, 10 nodes estació units a 10 nodes client. La simulació obté les següents xifres:

Nombre total de threads : 40

10 : un per cada client connectat a una estació

10 : un per cada estació connectat a client

20 : un per cada client connectat al seu veí (2 per connexió, amb 10 connexions)

Càrrega de CPU: 1% estable

Memòria: 114 MB

94 MB : 9400 KB per cada node C

20 MB : 1920 KB per cada node S

Xarxa: 4340 B/s (0,0347 Mbps) = Total Nodes S + Total Nodes C

Node S: 70 B cada 5 segons = 14 B/s

Total Nodes S: 14 B/s x 10 Nodes S = 140 B/s

Node C:

Entrada: 700 B (70B x 10 Nodes S) cada 5 s = 140 B/s

Sortida: 1400 B (700B x 2 Nodes C veïns) cada 5 s = 280 B/s

Total Nodes C: (140 + 280) B/s x 10 Nodes C = 4200 B/s

Les estacions s'han situat en diverses poblacions de Catalunya, a les següents figures es pot apreciar aquest fet.

La figura 14A presenta un aspecte general amb totes les capes de dades activades i el menú recollit. En la figura 14B es pot observar la mateixa vista però amb el menú de capes visibles. En el menú s'indica la cartografia externa que s'obté d'Internet i les capes a mostrar: en aquest cas estan hi ha 5 que corresponen al tipus d'observació que contenen els paquets de dades.

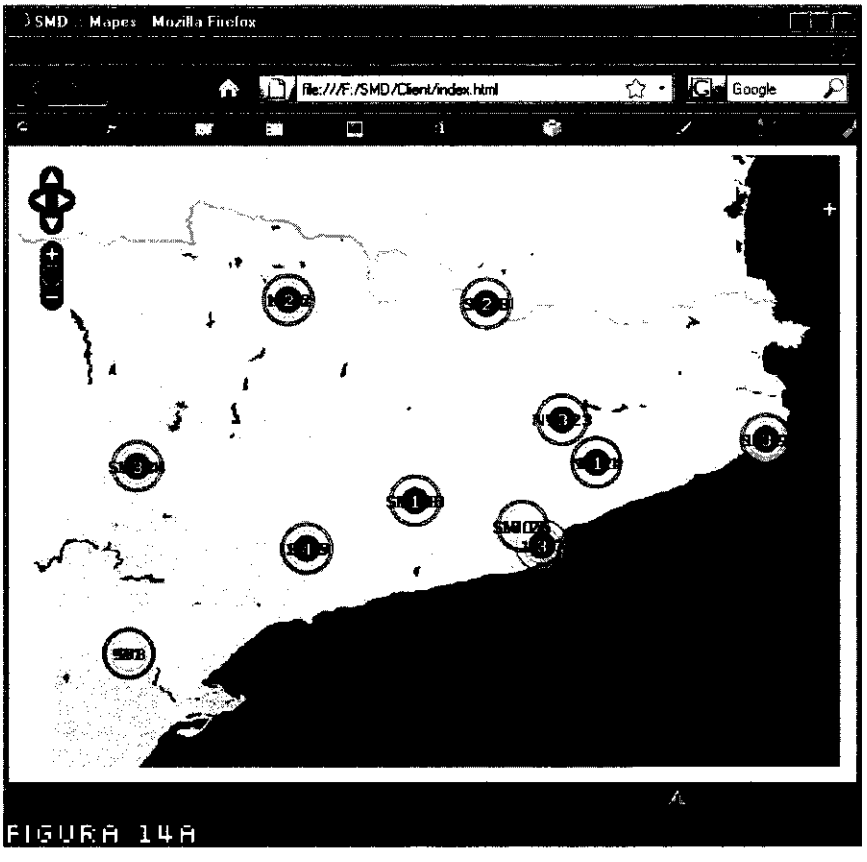


FIGURA 14A

FIGURA 14B

La següent figura 14C mostra únicament una capa, la del vent, on s'indica la direcció i la velocitat en km/h.

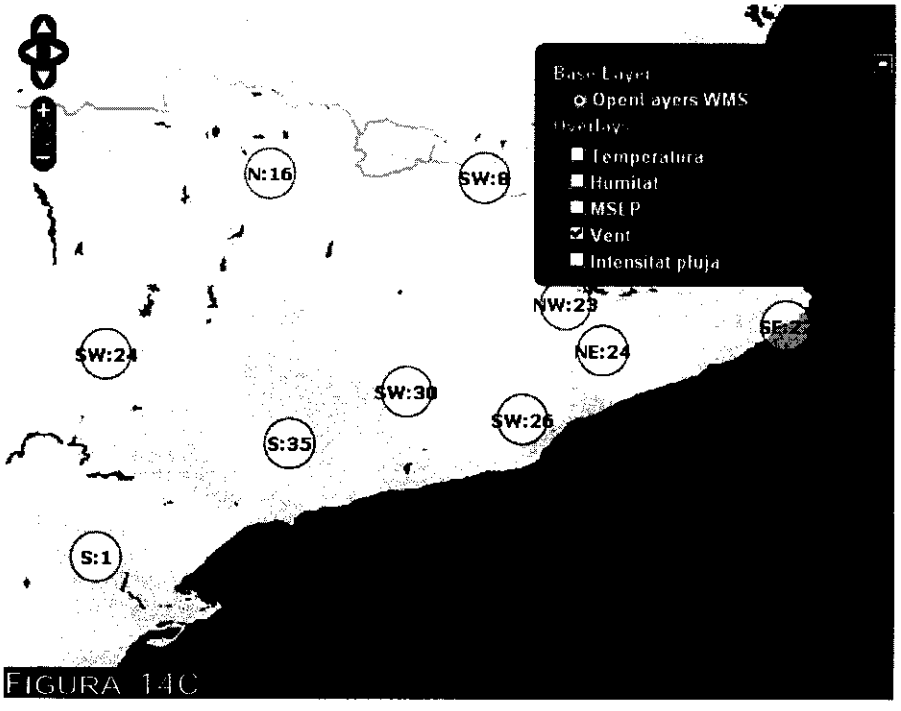


FIGURA 14C

A les següents figures es mostra la capa de pressió atmosfèrica reduïda a nivell de mar (MSLP). Una part negativa és que la dispersió de les estacions és molt irregular i per tant es pot donar lloc a que en l'àrea de Barcelona les indicacions es superposin tal com passa a la figura 14D.

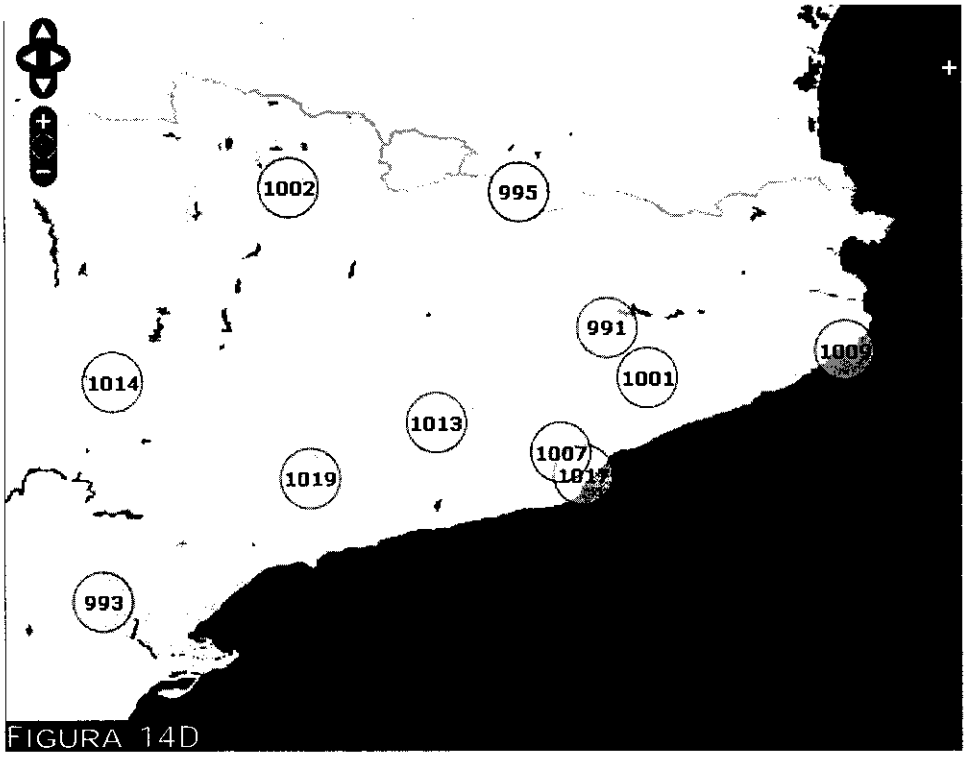


FIGURA 14D

Per tal de poder observar amb més precisió només caldrà fer un zoom sobre l'àrea corresponent tal i com mostra la figura 14E.

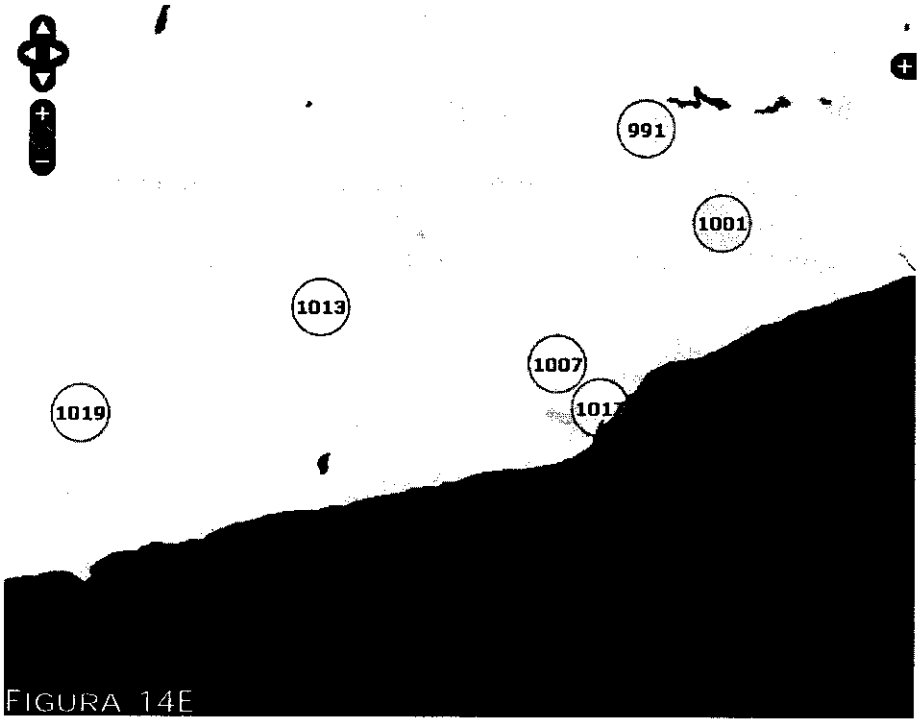


FIGURA 14E

A la següent figura 14F es pot observar quin és el resultat d'activar dos capes: la de la humitat i la de la intensitat de precipitació. La última queda per sobre en un color més fort.

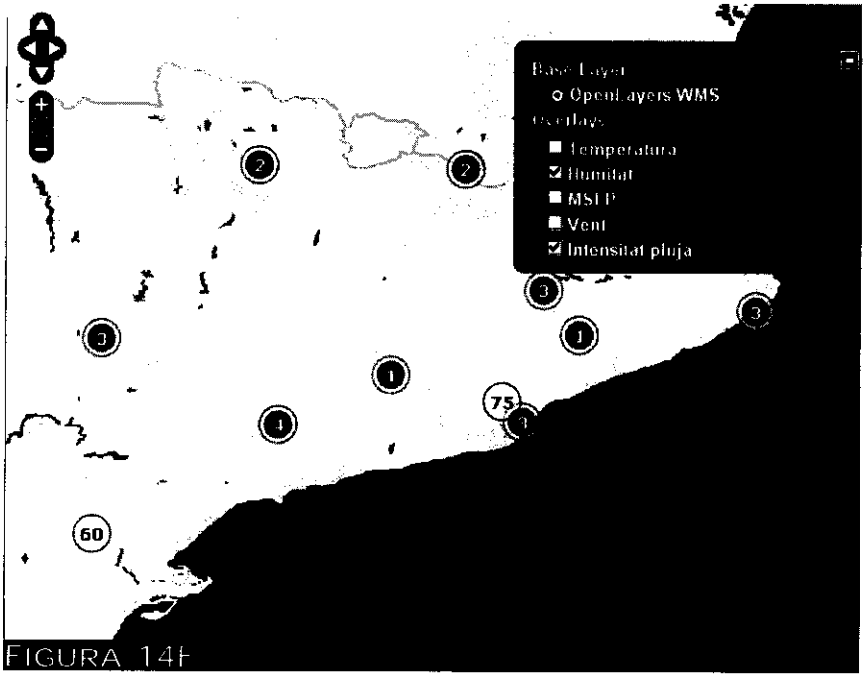


FIGURA 14F

Els resultats del test amb 10 nodes és molt satisfactori. La càrrega de CPU és manté molt baixa i la memòria per node també. Cal dir que no s'esperava que el cada node client, que està implementat en Java, ocupés 9400 KB de memòria RAM. No obstant ni el tràfic de xarxa ni el consum de CPU fan que el sistema vagi lent.

El gràfic mostrat al navegador també realitza les operacions relativament ràpid, es pot moure per les zones del mapa i fer zoom amb rapidesa.

9.5.3 Càrrega amb 50 Nodes

La simulació amb 50 nodes ha arribat en alguns aspectes al límit del sistema. Aquesta simulació s'ha llençat amb 50 nodes estació i 50 nodes client amb associativitat 3. S'obtenen les següents xifres

Nombre total de threads : 200

50 : un per cada client connectat a una estació

50 : un per cada estació connectat a client

100 : un per cada client connectat al seu veí (2 per connexió, amb 50 connexions)

Càrrega de CPU: 25% inestable, oscil·lacions entre el 2 i el 40-50%

Memòria: 615 MB

515000 MB : 10300 KB per cada node C

100 MB : 1920 KB per cada node S

9. Resultats

Xarxa: 525700 B/s (4,2 Mbps) = Total Nodes S + Total Nodes C

Node S: 70 B cada 5 segons = 14 B/s

Total Nodes S: 14 B/s x 50 Nodes S = 700 B/s

Node C:

Entrada: 3500 B (70B x 50 Nodes S) cada 5 s = 700 B/s

Sortida: 7000 B (3500B x 2 Nodes C veïns) cada 5 s = 1400 B/s

Total Nodes C: (3500 + 7000) B/s x 50 Nodes C = 525000 B/s

Aquest és el resultat gràfic de la simulació amb 50 nodes. La figura 15A és una vista global de tot el mapa:

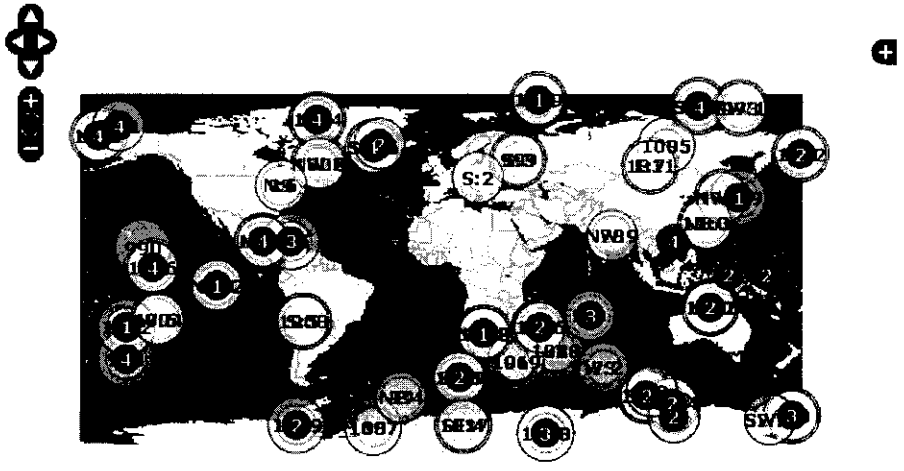


FIGURA 15A

La figura 15B mostra un zoom sobre Amèrica amb la capa de temperatures activada:

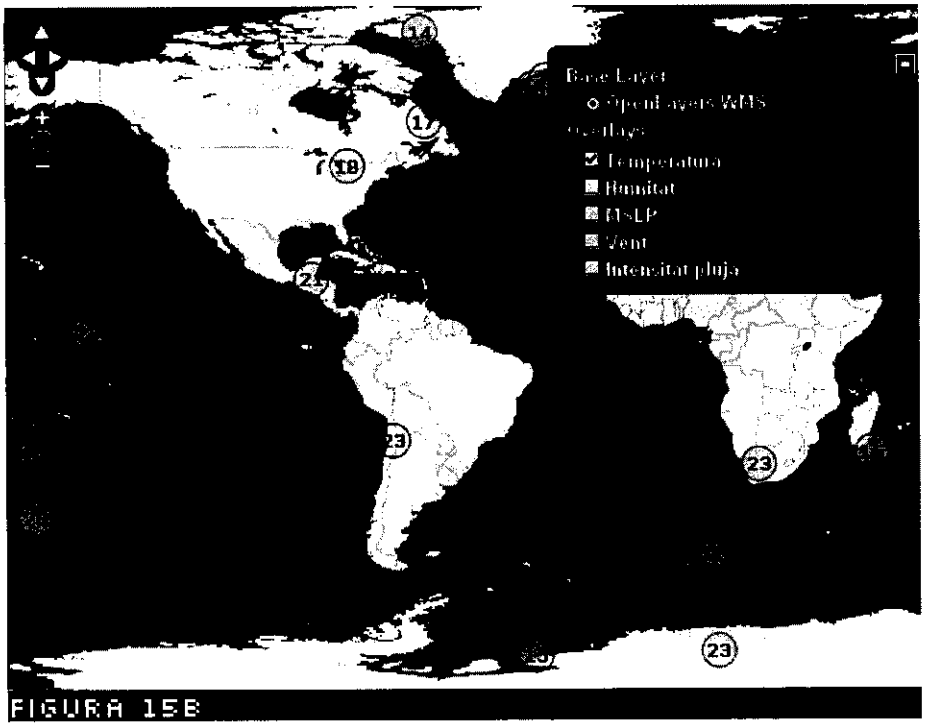


FIGURA 15B

Tot seguit un zoom sobre el sud-est d'Àsia amb la capa de Vent activada.

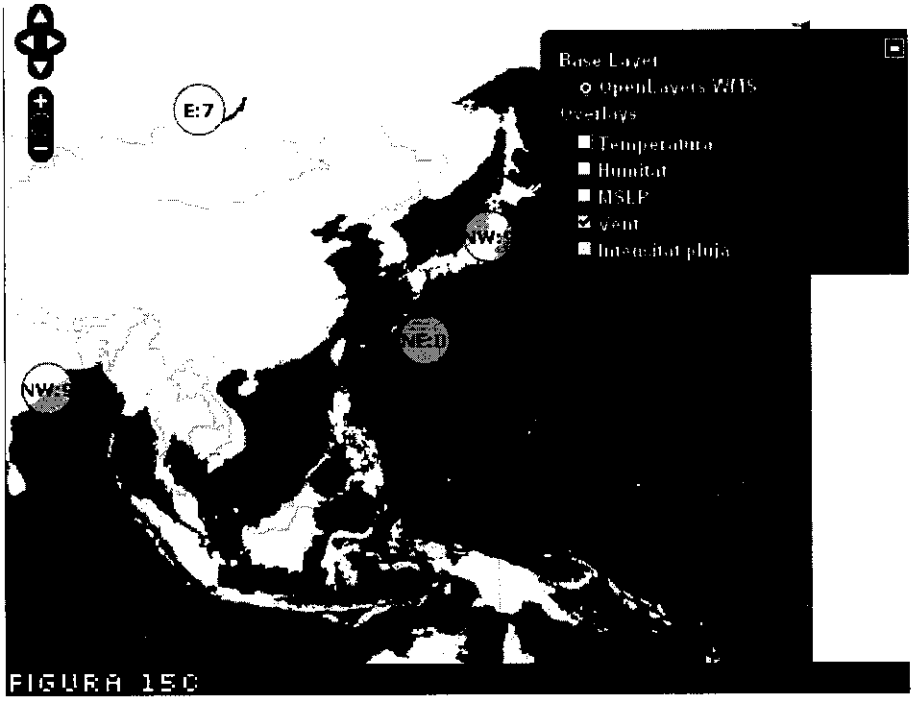


FIGURA 15C

El principal problema trobat en aquesta simulació no correspon al mecanisme d'enviament de dades entre nodes, la càrrega de la CPU o la memòria RAM usada sinó en la velocitat del mapa en el que es visualitzen les dades. Mapes com el de la figura 15A són feixucs degut a la quantitat d'informació a pintar sobre un mateix mapa, i el medi en que s'estan executant els gràfics: Javascript, que no és un llenguatge especialment òptim (tot i que les implementacions actuals l'han millorat molt en velocitat).

9.5.4 Càrrega amb 100 Nodes

No s'ha pogut fer una càrrega més enllà d'aquests 50 node C i 50 nodes S de l'apartat anterior. La prova amb 100 nodes no ha funcionat degut a que el sistema ha indicat un error en el moment de crear els threads.

Com que el sistema més lent vist fins ara en les proves ha estat la visualització mitjançant el navegador s'ha decidit llençar 100 nodes S connectats a un sol node C, de tal manera que hi haurà 100 dades sobre el mapa.

Els resultats numèrics obtinguts són aquests:

Nombre total de threads : 200

100 : un per cada client connectat a una estació

100 : un per cada estació connectat a client

Càrrega de CPU: 4% estable

Memòria: 201 MB

9240 KB : 9240 KB pel node C

192 MB : 1920 KB per cada node S

Xarxa: 2800 B/s (0,022 Mbps) = Total Nodes S + Total Nodes C

Node S: 70 B cada 5 segons = 14 B/s

Total Nodes S: 14 B/s x 100 Nodes S = 1400 B/s

Node C:

Entrada: 7000 B (70B x 100 Nodes S) cada 5 s = 1400 B/s

Total Nodes C: 1400 B/s

A continuació es poden veure les imatges resultants de la execució d'aquesta prova. La figura 16A és una imatge de tota la Terra amb totes les capes activades.

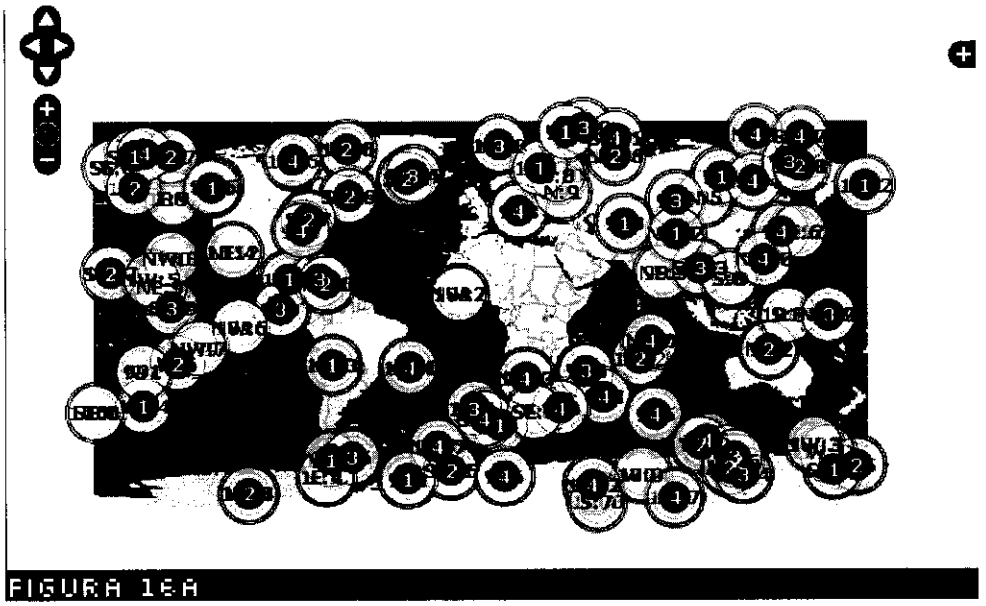


FIGURA 16A

9. Resultats

Des de la mateixa perspectiva de zoom es mostra la capa de temperatures en la figura 16B.

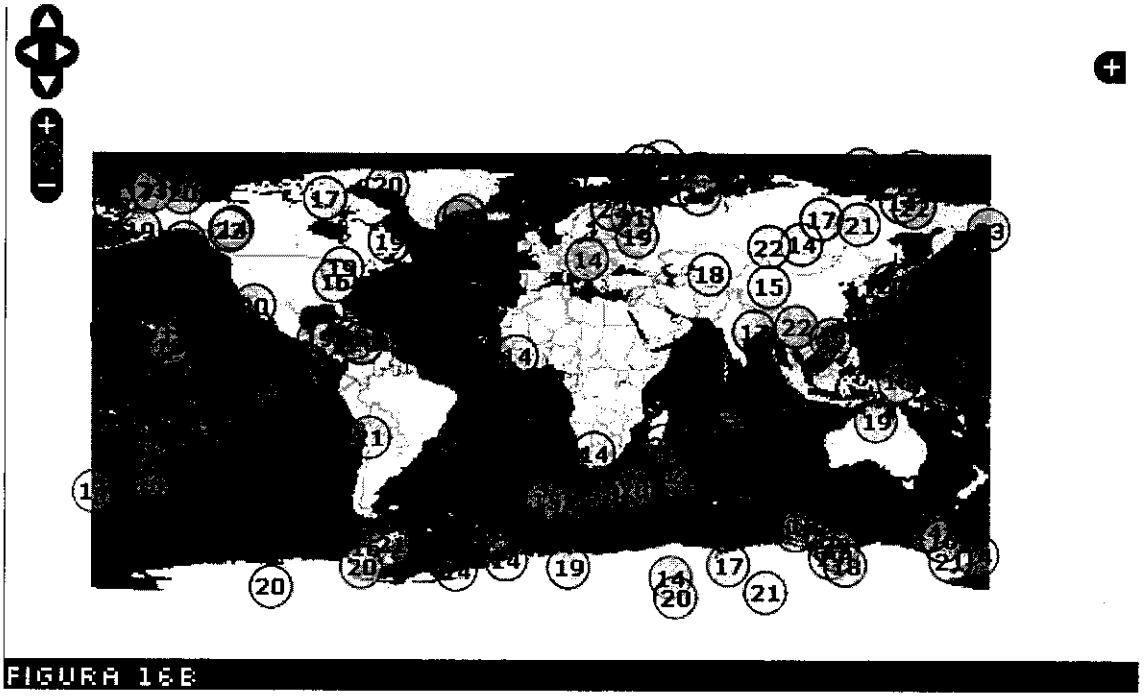


FIGURA 16B

La figura 16C és un zoom centrat sobre el continent americà on es mostren els vents.

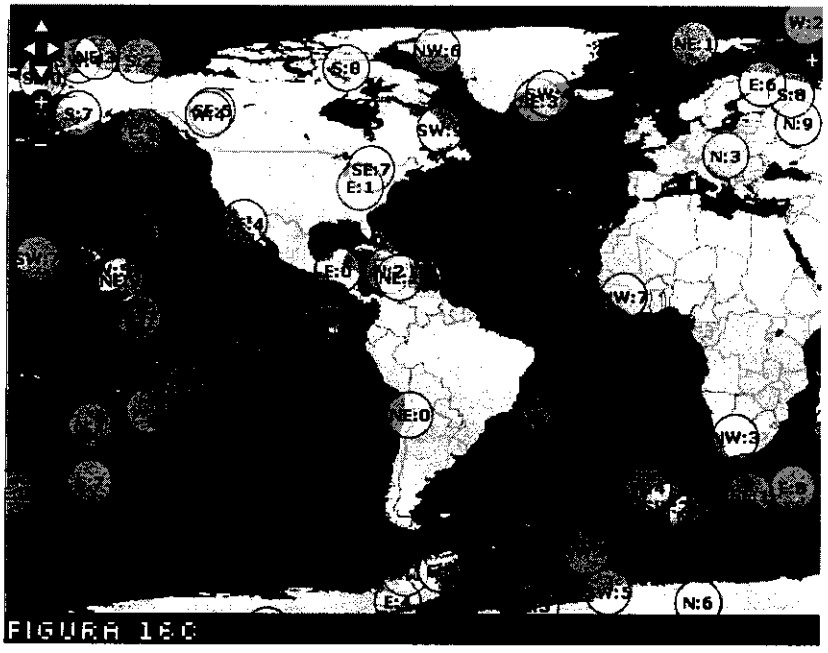


FIGURA 16C

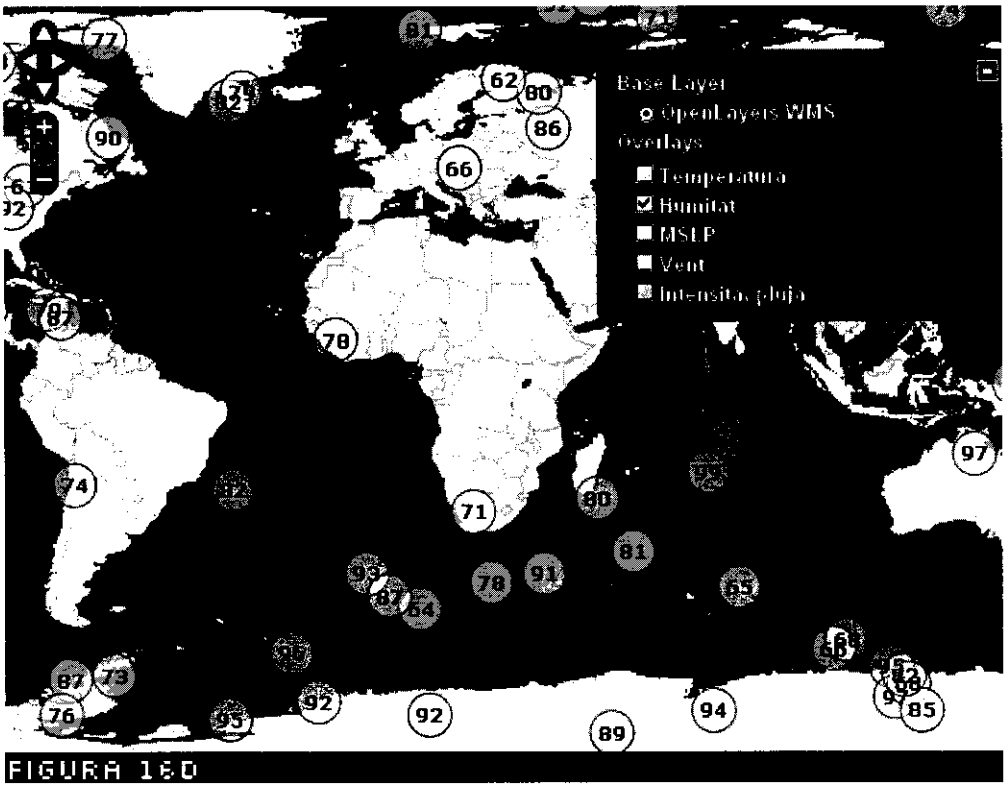


FIGURA 16D

A la figura 16D és mostra la capa d'humitat relativa en un zoom centrat prop del meridià de Greenwich.

La interacció amb el mapa és fa força complicada degut al gran nombre de nodes que s'ha de pintar. Es fa recomanable algun esquema que permeti reduir o limitar el nombre de nodes a pintar de tal manera que la navegació pel mapa sigui factible. També cal destacar que el fet de pintar gràfics sobre un mapa el realitza el navegador per la qual cosa un processador actual, de nova arquitectura amb multi-core i més freqüència de rellotge pot ser suficient per a moure una major quantitat de símbols sobre el mapa.

9.5.5 Conclusions sobre els resultats

Contrastant els resultats anteriors en els aspectes de memòria i de CPU es poden observar diversos efectes.

En l'aspecte de memòria es pot apreciar quin llenguatge ocupa la major part: Java. la relació entre un node S (implementat en C++) i un node C (implementat en Java) en l'aspecte de memòria és de l'ordre de 5 vegades major, és a dir, Java usa 5 vegades més memòria que C++.

Sobre els threads també hi ha aspectes a destacar. Mentre que un node S (C++) usa un sol thread (a més del procés principal) els nodes C (Java) usen diversos threads (un per a cada connexió amb altres nodes C o S). S'ha observat que quan es creen threads en Java, el nombre de threads totals del sistema no incrementa de la mateixa manera sinó que incrementen el doble. Per exemple, en la prova de 50 nodes (apartat 9.5.3) tenim 200 threads dels quals 150 són dels nodes C: si el sistema tenia 400 threads actius en el moment anterior a fer la prova, instants després, un cop la prova estava en el ple desenvolupament el sistema indicava un total de 1200 threads, cosa que fa sospitar que Java llença, de manera interna, altres threads.

En la prova en que s'han executat 100 nodes S amb un sol node C (apartat 9.5.4) també hi ha 200 threads (100 dels nodes S i 100 dels nodes C), però la seva activitat està molt reduïda donat que cada thread s'executa a intervals de 5 segons, que és el temps en que

l'emulador d'observacions en genera una nova observació. Mentre que la prova amb 50 nodes (apartat 9.5.3) té 150 threads amb molta més activitat donat que cada thread està enviant observacions que arriben al node C de manera continua. És aquest últim fet el que suposa un cost més elevat de CPU.

9.6 Resultat de la planificació

La planificació ha resultat una mica més llarga del que s'esperava a continuació s'exposen totes dues planificacions: la inicial i la final.

A la figura 17A està la planificació original:

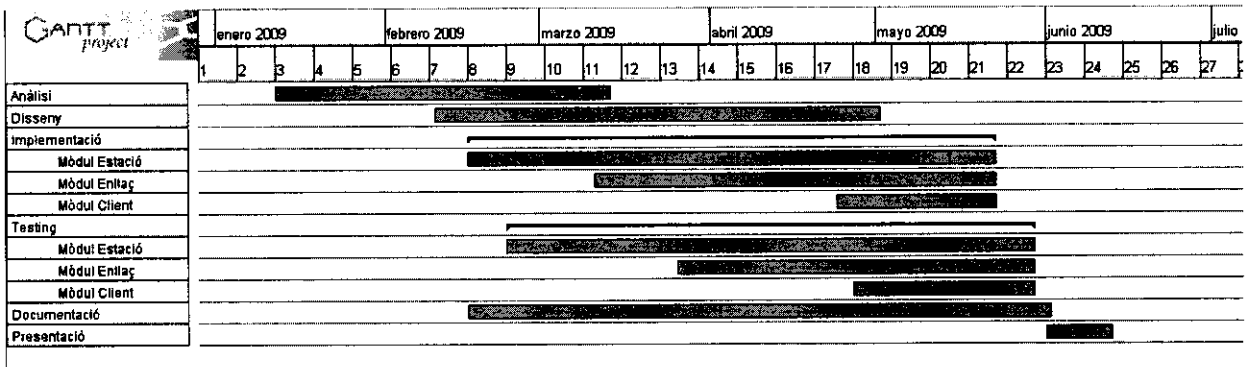


FIGURA 18A

Mentre que finalment el temps de desenvolupament de cadascun dels nodes s'ha allargat i en conseqüència també la entrega: en concret 3 dies més. com es pot veure en la figura 18B:

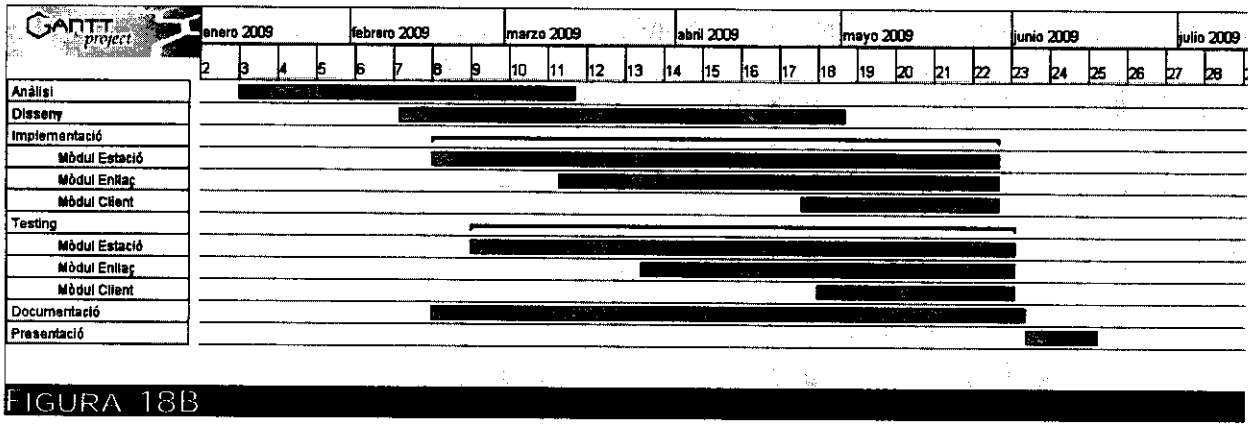


FIGURA 18B

9.7 Cost econòmic

El cost econòmic del projecte s'ha calculat en base a dos aspectes. El cost material i el cost del personal.

El cost material només és el de la línia telefònica: qualsevol tarifa plana oferida avui dia és suficient per fer que un tipus de node funcioni sense problemes. Es considera com a bàsic un ADSL de 1Mbps / 320 Kbps que avui dia s'oferta a preus de 20 a 30€

En l'aspecte personal el resultat econòmic caldrà avaluar-lo des dels rols que ocuparien la part de desenvolupament feta: és a dir, el disseny del sistema i el desenvolupament d'un prototip. És un cost aproximat donat que el projecte no està complet i d'altra banda és de codi lliure i ampliat per voluntaris de la comunitat.

Per tant el cost es detalla així:

Consultor: anàlisi i disseny del sistema

40 € / hora :: 12 setmanes = 19200 €

Analista: anàlisi i disseny del prototip i documentació

30 € / hora :: 16 setmanes = 18000 €

Programador : implementació del prototip i proves

25 € / hora :: 15 setmanes = 15000 €

Total: = 52200 €

9.8 Bibliografia i referències web

1: Sistema de coordenades WGS84 [Wikipedia]

http://en.wikipedia.org/wiki/WGS84

2: Software de visualització geogràfic global [Google Earth]

http://earth.google.com/

10 Conclusions

10.1 Introducció

En conseqüència del disseny inicial d'aquest sistema meteorològic distribuït i la seva posterior avaluació per a posar a prova els conceptes descrits s'han obtingut unes conclusions que s'han enfocat des de diferents punts de vista.

10.2 Objectius assolits

En el present projecte s'ha assolit l'objectiu bàsic, que era el disseny d'un Sistema Meteorològic Distribuït. No obstant el fet de dissenyar tot un sistema no ha resultat suficient i s'ha conclòs realitzar un prototip per tal de comprovar si la base del propi sistema era factible.

El prototip realitzat ha servit de gran ajuda per a posar a prova alguns dels conceptes de disseny que d'altra manera no haguessin donat un valor afegit a aquest projecte pel fet que es desconeixerien els resultats de la eficiència del mateix.

Cal destacar que no s'ha aconseguit implementar el 100% de tot el prototip, però sí s'ha implementat el suficient com per a posar a prova el rendiment del sistema en el seu estat més bàsic.

10.3 Coneixements obtinguts

El present projecte ha permès realitzar una volta completa a tota la carrera. El fet de dissenyar un sistema de manera completa dóna un enfoc més ampli donat que l'efecte ha estat el de fer un munt d'assignatures en una sola. Des de xarxes a estructures de dades, passant per enginyeria del software.

10.4 Valoració personal

Per a una persona afeccionada a la informàtica i a la meteorologia, el fet de poder realitzar un projecte sobre els seus interessos és un plaer.

La valoració personal és molt positiva per tot el que s'ha après i per les noves perspectives que dóna el fet de dissenyar un sistema de base. Considerant que el treball d'un enginyer és aplicar el que ha après al llarg de la seva carrera en la seva vida laboral, si a aquest fet li afegim que també es pot aplicar la enginyeria per a interessos personals la sensació que provoca és molt més plena.

10.5 Futur del projecte

Aquest projecte s'ha plantejat de bon principi com a projecte de software lliure, per tant és de lliure accés, de lliure modificació i de lliure distribució.

10. Conclusions

La direcció que prendrà el projecte a partir d'ara és crear el sistema sencer basat en la experiència adquirida pel prototip. Això involucrarà entre d'altres els següents aspectes.

1. Implementació de diversos drivers per tal que el node estació es pugui connectar a altres estacions meteorològiques.

2. Implementació d'un sistema de seguretat d'autenticació basat en clau pública

3. Implementació de mecanismes de control de coherència de la xarxa

4. Proves de rendiment sobre nodes en xarxa

5. Ampliació del sistema de visualització amb nous gràfics

6. I més important, fer que sigui usat entre la comunitat meteorològica.