H-LINK SIMULATOR

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Albert Sàez Montllor,
dirigit per
Juan Carlos Moure López
i co-dirigit per
David Rico Carreto
Bellaterra, 16 de juny de 2008.

ANNEX 3: CERTIFICACIÓ DE DIRECCIÓ

El sotasignat, Juan Carlos Moure López

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, 13 de JUNY de 2008.

ANNEX 4: CERTIFICACIÓ DE DIRECCIÓ EN EMPRESA

El sotasignat, David Rico Carreto

de l'empresa, HITACHI Air Conditioning Products, Europe S.A

CERTIFICA

Que el treball a què correspon aquesta memòria ha estat realitzat en l'empresa sota la seva supervisió mitjançant conveni per al desenvolupament del projecte final de carrera.
firmat amb la Universitat Autònoma de Barcelona.

Així mateix, l'empresa en té coneixement i dóna el vist-i-plau al contingut que es detalla en aquesta memòria.

Signat: .....

Vacarisses, 12 de Juny del 2008

Índex

Pàgina

Projecte H-Link Simulator

1. Introducció

El projecte H-Link Simulator ha estat proposat per la empresa Hitachi Air Conditioning Products Europe SA.

Aquesta empresa oferia a un informàtic la possibilitat de desenvolupar-lo durant una beca que alternés el desenvolupament del projecte amb les tasques de desenvolupament de software de control d'aires condicionats.

Els clients de la casa Hitachi es troben amb la necessitat de poder controlar les xarxes d'unitats d'aire condicionat. Per a realitzar-ho, no serveix anar amb el comandament a distancia màquina a màquina, i per a satisfer la demanda de poder-ho fer de manera centralitzada, Hitachi va crear un software de control anomenar CSNet Web. A la figura 1 podem veure com és la seva connexió amb la xarxa real d'unitats d'aire condicionat.

El CSNet Web és el principal producte de control d'unitats d'aire condicionat que te la marca al mercat. Funciona sobre un aparell Hardware anomenat HARC que serà explicat més endavant però que consta bàsicament d'un petit microprocessador. El CSNet Web esta dotat d'una interfície gràfica que s'executa a un PC connectat al HARC via Ethernet.

Com es pot veure a la figura 1, el CSNet Web es connecta a una xarxa d'unitats d'aire condicionat de la marca Hitachi utilitzant un bus H-Link. El protocol que les unitats i el HARC es propietari, per el que no en podrem mostrar la seva implementació.

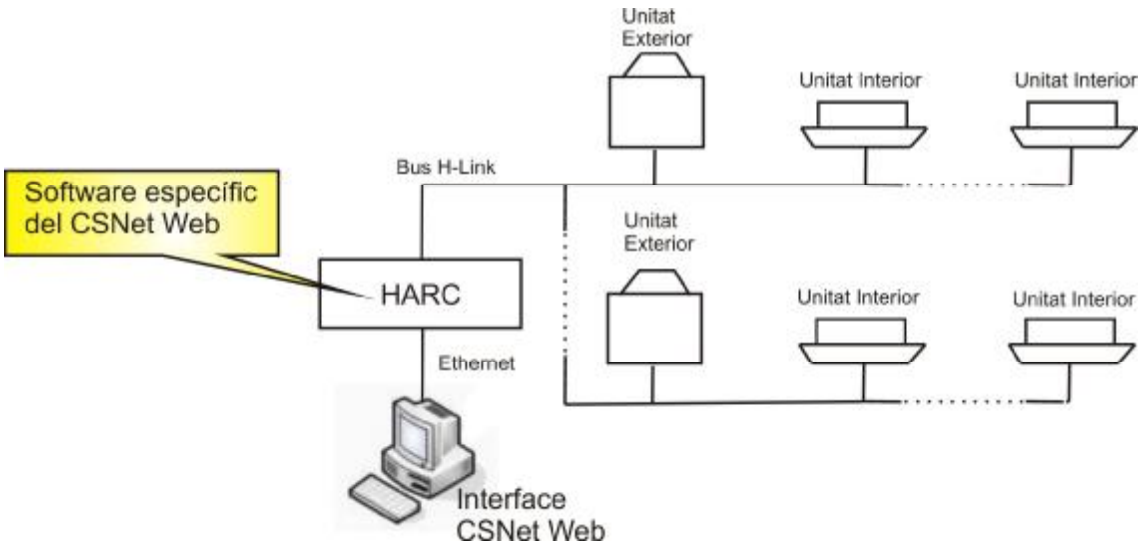

Figura 1: Esquema instal·lació d'unitats d'aire condicionat amb el CSNet Web.

Un cop Hitachi va treure el CSNet Web al mercat, va ser molt ben acollit, però sempre hi havia errors que s'escapaven a l'hora de testejar-lo. Aquest errors solen ser en grans instal·lacions, amb moltes màquines funcionant.

Hitachi no disposa a la seva fàbrica d'una instal·lació suficientment gran com per poder provar el CSNet Web amb les condicions d'aquestes grans xarxes. Tenir aquesta xarxa ocuparia molt d'espai, valdria molts diners i precisaria d'un manteniment elevat en temps.

Per poder estar segurs de que el producte es robust per tal de satisfer a la primera als clients amb xarxes grans, que òbviament corresponen a clients molt importants per a la marca, Hitachi proposa crear un simulador que pugui recrear l'existència d'un gran nombre de màquines.

El disseny i implementació d'aquest simulador és l'objectiu d'aquest projecte.

Aquest simulador ha de fer creure al CSNet Web l'existència d'unes unitats d'aire condicionat inexistents. D'aquesta manera es pot testejar, verificar i analitzar el software del CSNet Web en una instal·lació gran abans de llençar-lo al mercat.

Per tal de poder parlar el protocol H-Link, el simulador que es crearà funcionarà sobre un altre HARC que conte un petit processador en el que hi funciona un Linux limitat i sobre el que hi programarem en C/C++.

El simulador s'executarà en HARC diferent i que s'executa de forma independent del HARC del CSNet Web. Això serà així per a recrear amb més fidelitat la instal·lació real. El fet de separar-ho fa que els missatges siguin externs al HARC i per tant podem provar els possibles errors del CSNet Web i la comunicació amb les unitats.

En la figura 2 podem veure com quedarà la configuració del sistema utilitzant el simulador que es crearà per tal de poder testejar el CSNet Web.

El simulador por conviure amb unitats reals, ja que a ulls del CSNet Web serà com si ho fossin.

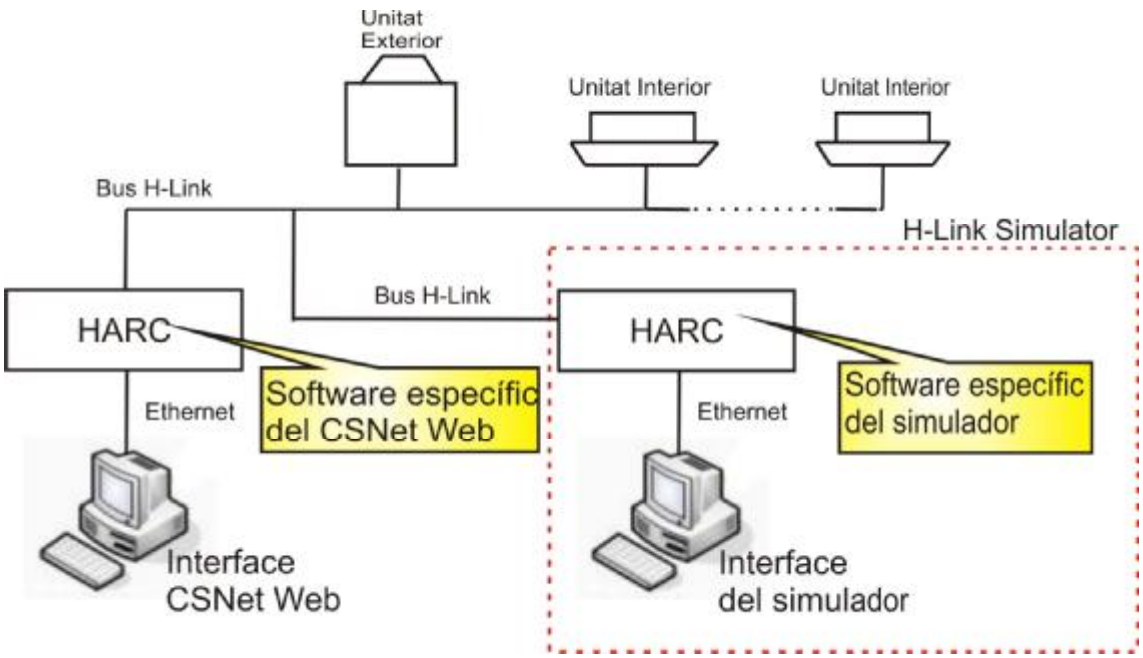

Figura 2. Esquema de d'instal·lació real amb el simulador.

Com que la connexió és en forma de bus sèrie, només es pot enviar un missatge a través a la vegada. És per això, que amb un sol HARC podem simular moltes unitats de manera fidedigne. A la instal·lació real en el pitjor dels casos per al CSNet Web rebria missatges de forma constant ja que no hi haurien col·lisions, i això ho pot simular perfectament el simulador enviant els missatges de forma seqüencial.

L'elaboració del software del simulador tindrà dues parts La primera és l'elaboració d'un software servidor que s'encarrega de simular les unitats d'aire condicionat. La segona part és l'elaboració d'un software client que s'encarrega de dotar al servidor d'una interfície gràfica.

Separar el simulador en 2 parts ens permet simplificar el seu us i dividir les tasques en dues parts que podran ser desenvolupades i testejades per separat.

L'objectiu del servidor és tenir la funcionalitat del simulador, però ens interessa que aquesta funcionalitat pugui ser governada de la manera més amigable possible, i aquesta és l'objectiu del client.

Existeix la previsió que aquest simulador sigui exportat a altres fabriques i que s'utilitzi a fires, i per tal de no anar acompanyat sempre d'un usuari expert, la implementació de la interfície client, resulta més que necessària.

2. Objectiu

Objectius principals:

- Crear el software que simuli l'existència d'una xarxa d'unitats d'aire condicionat de la marca Hitachi. Aquest software és el del servidor.
- Crear el software que permeti visualitzar i configurar l'estat del simulador des d'un PC. Aquest software és el del client.
- Elaborar la interacció entre el servidor i el CSNet Web a través del bus amb protocol H-Link.

Objectius secundaris:

- Capacitat per part del simulador de generar errors en la xarxa que estem simulant.

3. Conceptes previs

H-Link: Protocol de connexió propi de la marca Hitachi.

HARC: petit i limitat hardware, que disposa d'un processador sobre el que funciona Linux. Té entrades per a connectar amb el bus H-Link i amb un cable Ethernet.

Unitat interior: màquina d'aire condicionat que funciona dins d'un edifici. Té capacitat de connectar amb el bus H-Link.

Unitat exterior: màquina d'aire condicionat que funciona fora del edifici, que governa a les interiors. Té capacitat de connectar amb el bus H-Link.

CSNET WEB: software de visualització i control d'una xarxa real d'unitats d'aire condicionat. Funciona sobre un HARC.

4. Requisits

Després d'analitzar el problema es varen definir els següents requisits.

El servidor ha de:

- atendre les peticions de connexió del client o clients
- escoltar, processar i contestar els missatges rebuts pel client
- guardar la informació de les unitats que simula i modificar-la si el client ho requereix
- mantenir-se sempre encès, i ressuscitar tant en cas de morir com en cas d'entrar en estat de saturació
- generar els missatges H-Link necessaris per a la comunicació amb el CSNet Web.
- rebre els missatges H-Link necessaris per a la comunicació amb el CSNet Web.
- controlar errors de comunicació.

El client ha de:

- tenir una interfície gràfica amigable
- mostrar les dades del client sempre actualitzades
- permetre la edició de la topologia de la xarxa simulada
- permetre la modificació dels registres de les unitats
- controlar els errors de comunicació
- controlar els possibles errors provocats per un mal us de l'usuari

5. Antecedents

Hi ha un antecedent dins de la pròpia casa Hitachi a japó. Es tracta d'un simulador semblant al que és vol crear, però hi ha queixes de que no funciona correctament i que és molt limitat.

Un dels principals problemes que es van trobar a la hora de millorar-lo és que totes les especificacions estan en Japonès.

La versió antiga no s'ha pogut provar ja que esta en desús per a un baix rendiment, i se'n haurien de demanar copies a Hitachi Japó i això portaria molt temps.

Aquest temps encara seria més gran si tenim en compte que aquest no és un projecte crític per a la casa.

Els motius de que és desestimes el us del anterior versió, va ser perquè donava un molt mal rendiment, anava extremadament lent, i era molt limitat.

La seva codificació era molt poc estructurada i eficient, per això es va considerar que era més fàcil començar de zero, podent enfocar realment el projecte cap a les autentiques necessitats de la fàbrica europea i no les de Japó.

És desitja que aquest projecte assoleixi les funcionalitats i el funcionament que havia de tenir el primer, a part de la possibilitat de poder ampliar o modificar-lo si en un futur canviessin les necessitats ja que aquest codi si seria obert per a Hitachi Air Conditioning Europa S.A.

6. Estat del art

6.1 Simulador H-Link

Ens trobem que el projecte a desenvolupar, es un projecte per a us intern, sense un propòsit comercial directe, és a dir que el simulador únicament serà per a us de Hitachi simulant màquines Hitachi.

De la mateixa manera no es factible trobar coses similars a les altres empreses, ja que si tenen simuladors de les seves unitats no el tindran a la venda ni serà públic, per tant dons l'estat del art s'enfoca de manera més concreta en funció del problema.

6.2 Llenguatges de programació

Respecte als llenguatges de programació que es solen utilitzar en la creació d'aplicacions tant a Hitachi com a la competència són Java i C++.

Succeeix igual a la majoria d'empreses d'altres sectors també, ja que els dos son llenguatges orientats a objectes i que són d'una relativa senzillesa i potencia.

Entre ells dos el que surt més beneficiat és el Java per varis motius, entre ells, és més senzill, és molt potent i pot ser utilitzat sobre qualsevol sistema operatiu que tingui un intèrpret de Java.

Tot i això en aquest projecte s'utilitzaran els 2, ja que el principal problema de Java és el seu volum i necessitat de potencia per a ser utilitzat, en el cas del servidor, ens suposarà una pega, ja que com s'explicarà més endavant el HARC té una capacitat de memòria i computació reduïda.

Per aquest motiu, sobre el HARC s'utilitzarà C/C++. El client d'altre banda, com funciona sobre un PC normal i corrent, ens permetrà explotar la potencia de Java per a la creació d'entorns gràfics.

6.3 Instal·lacions d'aire condicionat

L'organització de les instal·lacions d'aire condicionat té una gran varietat en la seva topologia. El client és qui mostra una necessitat i en funció d'això i del preu tria una marca o l'altre.

En el cas d'Hitachi ens trobem que podem arribar a tenir 256 unitats dins la instal·lació. És una limitació donada pel bus de comunicacions però això no impedeix que el nombre d'unitats sigui major ja que es pot duplicar material per tal de tenir-ne més.

En el cas de la competència succeeix el mateix, en cas de no ser factible una instal·lació sempre es pot duplicar el material per a aconseguir-ho.

A la hora de la veritat el nombre d'instal·lacions sol ser en la majoria dels casos força menor, però els casos en que es superen son clients molt importants ja que disposen d'un gran nombre de màquines.

6.4 Motors de simulació

Per a aquest projecte s'ha estudiat la possibilitat d'usar motors de simulació, per a descriure els diferents estats en el que es pot trobar el servidor. En concret s'ha valorat la possibilitat d'utilitzar systemC, que és un llenguatge de descripció de hardware del estil Verilog o VHDL que utilitza la sintaxis de C++.

En la realització d'aquest projecte però, no s'utilitzarà perquè el sistema simulat és poc complexa com per justificar la utilització d'un llenguatge específic de simulació. A part d'això, el HARC és una màquina limitada, i si utilitzem C/C++ tindrem més control sobre el volum del programa i el tipus d'optimitzacions que vulguem realitzar-hi.

7. Estudi de viabilitat

La viabilitat de dur a terme el projecte estaria dividida en dues parts, l'obtenció del hardware, i el desenvolupament del software.

Pel que fa a la obtenció del Hardware, no hi ha cap problema, ja que al ser un projecte realitzat a una empresa, serà l'empresa qui l'aconsegueixi i qui el financí.

A Hitachi Air Conditioning Europe S.A. és disposa de tot el material necessari, HARC's, unitats d'aire condicionat, CSNet Web i busos H-Link.

L'ús d'un PC per a programar i utilitzar el projecte desenvolupat no suposarà cap dificultat, ja que és pot emprar un de l'empresa o un de personal.

Respecte al desenvolupament del software, estarà també dividit en dues parts, l'obtenció dels entorns de programació dels mateixos, i el desenvolupament del software.

La viabilitat d'obtenció del software necessari per a la programació, no serà cap problema, ja que necessitarem entorns per a Java i C++.

S'utilitzarà Eclipse i/o Netbeans, tots ells tenen dues grans qualitats, la primera és que son bons i potents, la segona és que són lliures i així no tenir la complicació de l'obtenció de la llicencia.

Per tant com podem veure a la taula de costos que hi ha a continuació, els costos totals del projecte no seran un problema per a la elaboració ja que, el material serà pagat per Hitachi, o en el seu defecte serà software lliure.

Element	**Cost**
1 HARC	0 €
Busos H-Link	0 €
Cable Ethernet	0 €
CSNet Web	0 €
Entorns de programació	0 € (Gratuïts)
PC	0 €
Total	**0 €**

Taula de Costos.

La viabilitat de dur a terme el projecte doncs, resideix únicament a la complicació de la programació i l'elaboració del mateix. Aquesta complexitat es veu reduïda al afrontar al dividir el simulador en client i servidor.

S'ha de tenir en compte com a complicació que al tractar amb un protocol propi de comunicació s'haurà de entendre com funciona el mateix.

El temps que es disposa per a la realització del mateix s'estima suficient ja que al desenvolupar-se dins de part del horari laboral garanteix un nombre d'hores setmanals que sumades a les de pròpia aportació, han de ser més que suficients.

En el següent capítol podrem veure la planificació de les tasques a realitzar de manera que podrem fer-nos una idea de com queda repartit el temps i a que es dedica, i així apreciarem que realment és suficient.

8. Planificació temporal de les tasques a realitzar

Per a la elaboració del projecte, inicialment es precisa de coneixements de programació en Java i C++, tal com coneixements bàsics de xarxes de computadors. Dins de Java és precisa de coneixements en Swing, ja que permet la creació d'entorns gràfics utilitzats per a la creació del client.

Al llarg del desenvolupament del projecte, es necessitarà conèixer com és el pas de missatges del protocol H-Link.

A la figura 3, es pot veure les diferents tasques a realitzar al llarg del projecte.

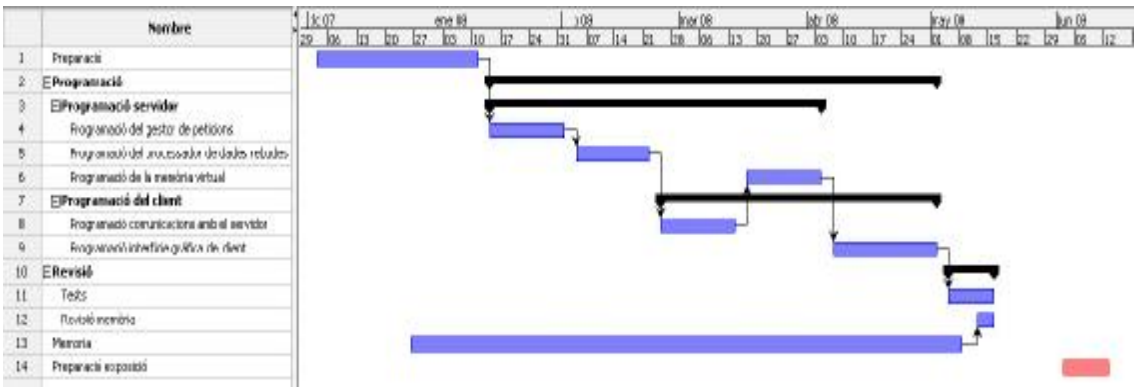

Figura 3: Diagrama de Gantt de les tasques a realitzar.

L'organització del projecte consta de 3 parts diferenciades:

- Preparació
- Programació
- Revisió

La part de preparació consisteix a definir què i cóm s'ha de realitzar, i a estudiar la viabilitat del projecte així com el seu abast inicial.

La segona part és la més important, la més llarga i la més difícil. No només emprarà temps a casa sinó que també temps de treball a la pròpia empresa. Consistirà en la programació i tindrà 2 grans flancs:

- La programació del servidor
- La programació del client

Començarem amb la programació del servidor, que tindrà 3 fases:

- Programació del gestor de peticions
- Programació del gestor de dades rebudes
- Programació de la memòria virtual

El gestor de peticions, és el que s'encarregarà de tractar totes les peticions de connexió que realitzaran els clients.

La següent fase, és la programació del gestor de dades rebudes, i com el seu nom indica, el que darà és processar els missatges rebuts per a realitzar les ordres que per a ells s'indiquin. També serà l'encarregat de preparar els missatges de resposta i d'enviar-los.

La tercera fase de la programació del servidor, és la que s'encarrega de la construcció de la memòria virtual, que consisteix a una estructura per guardar les dades de manera que utilitzant un sol valor, indicant el número de registre, accedim a la informació de la unitat desitjada.

Aquesta ultima fase però, s'implementarà després d'haver preparat al client per a comunicar-se amb el servidor, per comprovar d'aquesta manera que el funcionament es correcte.

La programació del client, per la seva banda, tindrà 2 fases:

- Programació de les comunicacions amb el servidor
- Programació de la interfície gràfica

La programació de les comunicacions amb el servidor, procurarà que client i servidor, es comuniquin de manera consistent, aconseguint així interacció entre els 2.

La programació de la interfície gràfica del client, consistirà en dotar d'un entorn amigable per a utilitzar les funcions del client, així com per mantenir de manera actualitzada la informació que guarda el servidor, que es mostrarà de la manera més ordenada possible.

Durant l'elaboració del projecte s'anirà desenvolupant una tasca paral·lela que és la de l'elaboració de la memòria, per tal d'evitar tenir una saturació a l'últim moment en la realització de la mateixa.

La revisió consistirà en dues parts, la primera tracta d'assegurar que tot funciona correctament. Això es farà mitjançant diversos tests, que no només han de mirar que el client i el servidor no fallin, sinó que hauran de fixar-se també en veure que no haguem deixat res per fer.

La segona part de la revisió, es realitzarà un cop acabada la memòria, i consistirà en revisar que el contingut de la memòria és el especificat en les pautes per a la realització d'un projecte final de carrera.

9. Protocol H-Link

És un protocol de comunicació creat per Hitachi i utilitzat únicament per la mateixa marca. El seu funcionament no està obert al públic. El que fa és permetre la comunicació entre els diferents productes de les xarxes d'aire condicionat de la marca Hitachi.

Per evitar les col·lisions segueix el protocol CSMA/CD.

El protocol H-Link té copyright, i prohibició explicita per contracte de no mostrar el seu funcionament ni estructura en públic.

Aquest fet doncs, suposa un problema per a la implementació del projecte, ja que és perd la gràcia de mostrar el funcionament en si del protocol.

La solució optada, és desenvolupar externament la part de les comunicacions via H-Link que seran cridades des del servidor com a una llibreria estàtica externa.

D'aquesta manera el codi queda amagat, però es pot provar la seva funcionalitat.

La decisió de fer-ho en una llibreria estàtica va ser que ens complicava menys la implementació però el principal problema d'haver-ho fet així és que a la hora de compilar ens haurem d'assegurar que el fitxer binari de la llibreria és la ultima versió.

No és descarta fer el pas a una llibreria dinàmica de manera que únicament s'hagi de canviar el codi a la llibreria i la pròpia compilació ja refaria el binari. No s'ha dut a terme durant la implementació amb llibreria dinàmica dins d'aquest projecte per que s'ha preferit invertir el temps en parts de la que es pogués apreciar el resultat de forma més directa.

10. Organització del hardware/software

10.1. Organització del Hardware

Elements Hardware del projecte:

- Bus H-Link
- Unitats d'aire condicionat
- HARC
- HARC Web
- PC
- Cable Ethernet

10.1.1. Bus H-Link

És un Bus sèrie de 2 fils non polar del tipus Twisted pair shielded. És 9600 bauds.

La seva funcionalitat és permetre enviar i rebre dades de les màquines d'aire condicionat Hitachi. Cal notar que donades les seves característiques te una limitació a 256 unitats connectades.

10.1.2. Unitats d'aire condicionat

Les unitats d'aire condicionat no són un element en si del projecte, ja que seria impossible, portar-les per exemple a la presentació. Són mencionades en aquesta part perquè la funcionalitat final del projecte és simular-les i interactuar amb les reals, i per tant s'ha considerat necessari fer una petita explicació de que són.

Podem trobar dos tipus generals de màquines d'aire condicionat:

- Outdoor Unit: per abreviar les anomenarem sovint OU (Outdoor Units). Són aquelles que es troben al exterior de les instal·lacions. Cada una governa un conjunt d'unitats interiors, que usarà per a graduar el condicionament del aire en aquella zona.
- Indoor Unit: per abreviar les anomenarem IU (Indoor Units). Es troben al interior de les instal·lacions i són les encarregades de efectuar les accions necessàries que se'ls hi indiqui per tal de donar un bon condicionament del aire.

10.1.3. HARC

És un element hardware programable que permet comunicar-se a traves de cable i protocol H-Link, es a dir, és un controlador real que ens permet interactuar amb una instal·lació d'aire condicionat Hitachi. També té connexió Ethernet, que ens permetrà la connexió amb el client del simulador.

Les característiques del HARC són:

- 1 Microprocessador SH3 a 200MHz
- 64MB de memòria RAM
- 1 memòria compact flash de 256MB
- Sistema operatiu Linux basat en la versió 2.4
- 1 connexió Ethernet
- 1 connexió H-Link
- Disposa també d'altres tipus de connexions sèrie que no utilitzarem

Figura 4

Sobre el HARC programarem software del servidor, que el que farà es simular unitats d'aire condicionat.

Tot i que hem mencionat abans que la limitació del bus H-Link és de 256 unitats, Hitachi limita el seu us a 224. De les quals, 160 són unitats interiors, i 64 unitats exteriors, també és limita el màxim nombre de unitats interiors que pot tenir una unitat exterior a 64.

En aquest simulador, s'utilitzarà inicialment un únic HARC on es simularà les possibles 224 màquines, però la programació del mateix ha d'estar enfocada a poder ser remodelat en un futur no molt llunyà per tal de poder separar les unitats exteriors i les interiors en 2 HARC's.

Per tant, un HARC tindrà 3 modes de funcionament:

- Simulador d'unitats interiors
- Simulador d'unitats exteriors
- Simulador de ambdós tipus d'unitats

La aportació de separar-ho en 2 HARC's és que els missatges entre unitats interiors i exteriors, també serien H-Link, assemblant-se més a la xarxa real, i permetent muntar una xarxa més especifica per si les necessitats de les proves ho requerissin.

És a dir, si volguéssim provar una unitat exterior, podríem utilitzar el simulador en mode unitat interior, i interactuar-hi, o a la inversa. Però com ja hem dit, el present projecte, tot i valorar i tenir present els 3 modes de funcionament a implementar, només implementarà el que simula els 2 tipus d'unitats, ja que la totalitat de productes que és desenvolupen en el departament de controls de Hitachi Air Conditioning S.A. estan

enfocats a el control general de les xarxes i com les unitats exteriors ja estan dissenyades per a controlar les interiors, no serà necessari la implementació dels altres 2 modes.

Per a programar en el HARC, ja que és una màquina amb Linux, i és limitada, no disposem de tot el gcc, únicament tenim la possibilitat de programar en assembler, C o C++. D'aquests 3, ens interessaria agafar un llenguatge orientat a objectes, però Hitachi diu que C++ dona problemes de rendiment amb STL, es per això que triarem d'utilitzar C++ ja que és orientat a objectes, però únicament l'utilitzarem per a les classes, per a la resta utilitzarem C Basic ja que dona un millor rendiment.

A part del HARC on programarem el simulador, en necessitarem un altre únicament per a comprovar el correcte funcionament, i en aquest HARC el que hi farem és fer-hi funcionar el CSNet Web.

10.1.4. HARC Web

És tracta d'un HARC igual que en el que hi programarem el simulador, l'únic que aquest estarà preparat per a fer córrer el software del CSNet Web. Per tant, aquets HARC no forma part del projecte, però si serà necessari per a poder comprovar el correcte funcionament del mateix ja que és el HARC contra el que haurem de comunicar-nos.

Sobre aquest HARC no hi programarem, únicament el farem anar per veure que el CSNet Web detecta les unitats que estem simulant i que els hi pot enviar ordres de control.

Aquest HARC es connectarà amb el simulador per H-Link, i per Ethernet amb el ordinador que estarà executant el CSNet Web.

10.1.5. PC

És un ordenador normal i corrent que executarà el software del client creat.

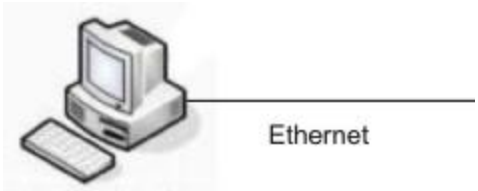

Figura 5: Ordinador amb connexió Ethernet.

La seva funcionalitat es comunicar-se amb el HARC, per poder visualitzar el funcionament de les unitats simulades en ells, i per a interactuar amb ells. Aquesta comunicació serà via Ethernet.

Per a programar en el PC utilitzarem Java ja que podem aprofitar les llibreries swing per a dotar d'una interfície gràfica atractiva sense tenir que perdre temps en el disseny. A més Java es un llenguatge orientat a objectes que ens pot ser molt útil per a la elaboració del treball.

Sobre el mateix PC o utilitzant-ne un altre, farem córrer el software del CSNet Web connectat al seu corresponent HARC per tal de poder comprovar el correcte funcionament del nostre simulador.

10.1.6. Esquema físic de l'organització del hardware

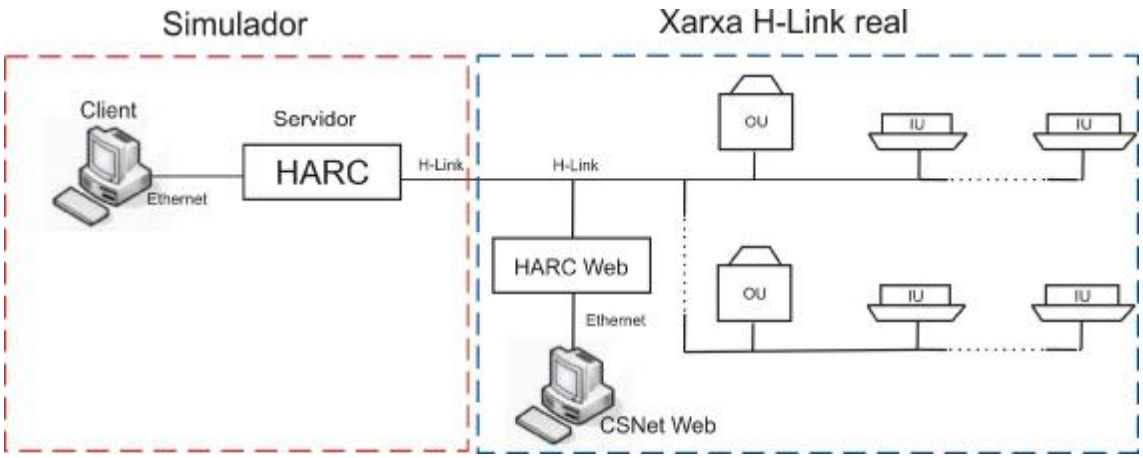

Figura 6: Esquema físic de la xarxa real amb el simulador H-Link.

Com podem veure a la figura 6, tindrem el PC, on executarem el software del client, connectat via Ethernet al HARC que simularà les unitats interiors i exteriors, aquest PC mostrarà per pantalla els valors dels registres de les unitats que es trobaran al HARC.

Des del HARC ens comunicarem amb el HARC Web via H-Link. Per a controlar el HARC Web utilitzarem el CSNet Web que funcionarà sobre un ordinador connectat via Ethernet amb el HARC Web.

Les unitats interiors i exteriors d'aire condicionat com és comprensible el dia de la presentació no estaran presents ja que únicament serviran per a la elaboració del projecte per a veure que el CSNet Web treballa bé tant amb les unitats simulades com amb les reals.

Per fer el simulador necessitarem únicament un PC i un HARC.

Com l'objectiu del projecte és simular les unitats de cara al CSNet Web, necessitarem per a comprovar el seu funcionament un altre PC que pot ser el mateix on executarem el client del simulador i un HARC Web, de manera que puguem executar el CSNet Web i veure que l'objectiu del simulador es compleix.

10.2. Organització del software

Elements software del projecte:

- Servidor
- Client

10.2.1. Servidor

Ha de simular l'existència de les unitats d'aire condicionat, tant exteriors com interiors. És troba dins del HARC, i ha de respondre tant a les comunicacions via H-Link, com a les peticions del client via Ethernet.

10.2.2. Client

Realitzat en Java i funcionant sobre el PC, haurà de llegir la informació que rebrà via Ethernet del HARC que simula les unitats.

10.3. Presa de decisió del control

Primer de tot hem de prendre la decisió de qui tindrà el control sobre les unitats d'aire condicionat i tenim dues opcions, realitzar-ho des de el PC o des de el HARC.

Degut a les limitacions físiques del HARC, el control el passarem al PC, que a més ens proporciona major facilitat de comprensió del que estem fent ja que disposa directament de la interfície del simulador. Per tant el HARC simularà únicament el comportament de les unitats enfront les ordres de control rebudes.

Aquets control serà bàsicament el fet de poder afegir o eliminar màquines, modificar o llegir registres de les mateixes. A part d'això el Client tindrà control sobre el servidor, li podrà enviar ordres de reset, així com canviar diverses opcions de funcionament.

11. Comunicació

11.1. Comunicació entre el PC i el HARC del simulador

La comunicació entre el HARC del simulador i el PC, és realitzarà mitjançant un cable Ethernet. Seguit el protocol Modbus, que funciona sobre TCP/IP.

Volíem un protocol que funciones sobre TCP/IP, ja que és molt fiable, i teníem tres opcions principals que eren HTTP, la creació d'un protocol propi o la cerca d'algun altre protocol ja existent.

La opció del protocol propi, es va destacar ràpidament ja que era innecessari re inventar la roda sense que fos aquest l'objecte del projecte.

La opció del HTTP, va descartar-se perquè hauríem de treballar sobre fitxers i no sobre dades, i treballar sobre fitxers segons feia més feixuc pel tema de tenir que processar-los, així doncs preferíem algun altre protocol, i és aquí on va aparèixer el Modbus.

El protocol Modbus és molt utilitzat en automatització, i a l'empresa s'utilitza en varis dels seus altres productes com per exemple en el CSNet Web, fet que l'elecció del Modbus fos atractiva, tant per a possibles futures aplicacions i interaccions amb altres programes de l'empresa, com pel fet de tractar un protocol nou, no tractat al llarg dels estudis d'informàtica.

Però la decisió d'agafar aquest protocol, no va acabar en aquesta senzilla però contundent raó, és va estudiar el protocol, arribant a dues grans conclusions per a triar-lo. La primera és que només té un overhead de 4 bytes, i que permet llegir gran quantitat de registres d'un sol cop.

La segona raó era un dels seus principals problemes, que és la seguretat, però en aquest cas no és una cosa que ens importi ja que la connexió és directa utilitzant el cable, pel que no hem de patir per possibles hackers.

Per tant, Modbus, és un protocol que encaixa a al perfecció amb les necessitats de realitzar les comunicacions que necessitem entre el client i el servidor.

Java disposa de llibreries de comunicació Modbus, pel que simplificaria la part de la comunicació en el client, tot i que s'haurà d'implementar des de 0 en el servidor.

Com es pot veure en els annexes, aquest protocol implementa, moltes funcions, però nosaltres ens quedarem amb les bàsiques, que són el *readMultipleRegister*, i el *writeRegister*.

La tria d'aquestes dos és obvia ja que necessitàvem llegir i escriure ens els registres del servidor, però el fet de escollir la lectura múltiple, és perquè ens permetrà llegir un volum més gran de dades de cop, reduint, el overhead final i el temps de lectura de molts registres, com serà el cas de quan el client s'inicialitzi, que necessitarà llegir tota la topologia del servidor, per crear el entorn gràfic.

En el cas de l'escriptura, no necessitarem en cap moment escriure més d'un registre, ja que l'usuari només els podrà caviar d'un en un en el client, i en el cas per exemple de eliminar una unitat, se li enviarà l'ordre al servidor, i no caldrà doncs anar esborrant un a un els registres, que és un dels possibles motius pel qual ens hagués estat interessant implementar aquesta funció.

Tot i això la funció *writeMultipleRegister* ha estat implementada i comprovada per si en un futur trobéssim la necessitat d'utilitzar-la.

D'altre banda s'ha de comentar que el protocol Modbus esta basat en una estructura master-slave, es a dir un dels dos comunicants es el que demana la informació i l'altre simplement la servirà, per tant no podrem fer que el servidor, si una data se li modifica, notifiqui a tots els clients connectats que aquest canvi ha passat.

Per a fer-ho hauríem de establir una altre comunicació master-slave en sentit contrari. Aquest fet ens duplicaria el nombre de connexions, i això faria que l'atenció als clients fos més lenta ja que se'ls atén seguint una cua circular com s'explicarà més endavant.

Així doncs tenint en compte aquesta restricció i aprofitant que el fet de que haguem decidit donar-li el control de la comunicació al client, el que farem per a que actualitzat, és que vagi consultant les dades al servidor.

D'aquesta manera, tindrem missatges innecessaris ja que consultarem dades que potser no han canviat, però ens ajustarem a l'estructura master-slave, i alliberarem al HARC de més feina innecessària.

11.2. Comunicació entre el simulador i el CSNet Web

En la xarxa real d'unitats d'aire condicionat, les unitats es comuniquen mitjançant el protocol H-Link, per tant per a comunicar-nos amb el CSNet Web, les nostres unitats fictícies, hauran de seguir les especificacions del H-Link.

A continuació veurem com es la comunicació entre el HARC del CSNet Web, les unitats exteriors (OU) i les unitats interiors (IU):

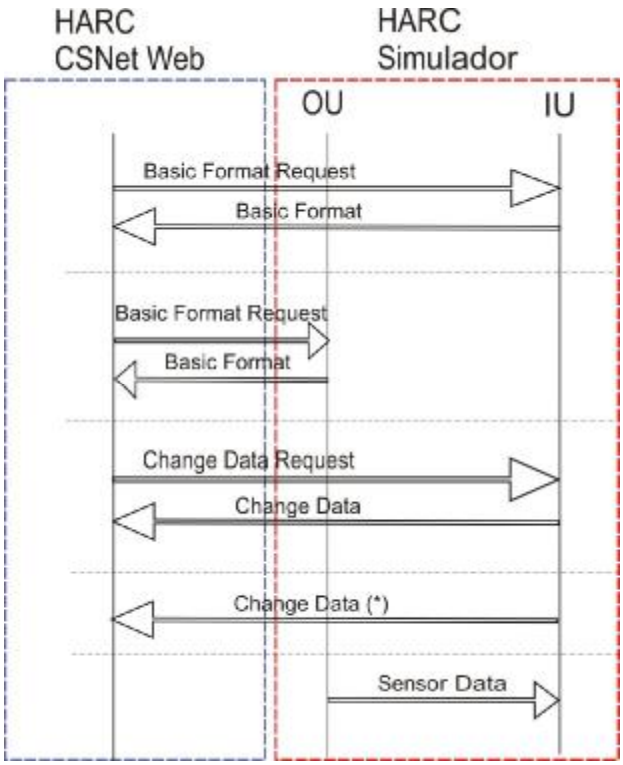

Figura 7: Esquema de pas de missatges entre el CSNet Web i el simulador.

Com el projecte ha de simular les unitats exteriors i les interiors, en la part en vermell de la figura 7, hem marcat la tot el que hem de simular.

L'obtenció de dades de les unitats interiors i exteriors per del CSNet Web es realitza a partir de la petició H-Link de Basic format a les unitats tant interiors com exteriors per tal de que aquestes li responguin, el paquet Basic format que contindrà la informació necessària per a mantenir al dia la informació que mostra el CSNet Web.

Realitzar un canvi en les dades de les unitats interiors és fa realitzant un funcionament semblant al de la consulta. El CSNet Web envia un Change Data Request amb les dates a canviar i un cop canviades es notificarà al CSNet Web amb un Change Data que aquestes han estat canviades.

Com s'aprecia la figura 7, el CSNet Web i les unitats exteriors no intercanvien paquets de Change Data. Això és ja que a les OU no se'ls hi poden canviar.

El Change Data (*), que veiem a la figura 7, és un Change Data normal, però cal notar que es donarà quan a una unitat interior se li realitza un canvi no demanat per el CSNet

Web, aquesta informarà amb l'enviament d'un paquet Change Data indicant les dades canviades, és així per tal que el CSNet Web sàpiga quan abans que aquesta ha estat canviada sense esperar a que li toqui preguntar-li.

El missatge Sensor Data, va de OU a IU, i formaria part d'uns missatges que inicialment obviarem ja que simulem les unitats exteriors i interiors en un mateix HARC. Per tant, aquests missatges serien interns al driver. La seva funcionalitat es de realitzar un control de les dades de les IU cada 3 segons, i ens serà interessant per poder conèixer aquestes dades.

En el cas de fallida en el pas del missatge, el CSNet Web intentarà reenviar-lo 32 cops, i si no ho aconsegueix esperarà 2 minuts fins per a reenviar-lo, i així repetidament fins que ho aconsegueixi.

De la mateixa manera, si la resposta no li arriba en 2 minuts el CSNet Web tornarà a enviar la petició fins que la resposta li arribi.

12. El servidor

12.1. Introducció

Com ja s'ha explicat abans, el servidor ha d'atendre les peticions del client i ha de mantenir-se sempre viu. També ha de tenir guardada tota la informació de les unitats d'aire condicionat que estem simulant preocupar-se de seguir la comunicació H-Link amb el CSNet Web.

Aquest software funcionarà sobre un HARC que disposa de Linux, així que per a la programació del codi s'ha triat el llenguatge de programació C/C++, en el que utilitzarem les llibreries de TCP/IP per a la comunicació amb el client.

El servidor esta format per varis threads on cadascú desenvoluparà part de les funcionalitats a implementar pel servidor.

12.2. Parts que el formen

12.2.1. Control principal

Es tracta d'un thread que és l'encarregat de posar en marxa el servidor, s'encarrega també de comprovar que la resta de threads segueixen vius, o que segueixin funcionals, es a dir, que no estiguin saturats.

Per a fer-ho utilitzant variables on cada un dels threads emmagatzema el temps actual de manera que si algun no respon, els mata i els torna a llençar.

Aquest thread guarda les variables de control que es poden modificar des de el client, de manera que si detectem que per exemple la variable reset ha estat activada, aquest thread que és el que du el control sobre els altres, els cancel·larà i tornarà a executar-los.

Al arrencar, aquest thread, també crearà la memòria virtual on és guardarà la informació de les unitats d'aire condicionat simulades.

12.2.2. Escoltador de peticions

La seva funcionalitat és processar els peticions de connexió que rep el servidor, de manera que si un o més clients volen connectar-hi, els hi assignarà un socket pel qual comunicar-se, i seguirà esperant més possibles connexions.

Aquest socket serà passat a un altre thread que serà l'encarregat de escoltar els missatges.

El motiu de no crear un thread per cada connexió, és per evitar col·lapsar el HARC, ja que és una màquina poc potent i si tinguéssim molts threads podria entrar en mal funcionament.

12.2.3. Receptor de missatges Modbus

La funció d'aquest thread es atendre totes les connexions creades. Per a fer-ho recorre l'array on guarda les connexions com si fos una cua circular, és a dir, va del primer al últim, i llavors torna al primer.

La comprovació que fa per cada socket és mirar si ha rebut alguna cosa, si ha rebut un missatge el passa a la cua del processador de missatges, indicant-li també per a quin socket ha de respondre, en el cas que s'hagi perdut la connexió tancarà el socket.

El motiu de no processar els missatges en aquest thread es obvi, quan més el carreguem, més lenta serà la sensació d'atenció de cada connexió ja que tardarà més en tornar a rebre-la per això es desprèn del missatge tan bon punt detecta que s'ha rebut.

12.2.4. Processador de missatges Modbus

Te una cua on trobarà els missatges rebuts, i el que farà aquest thread és anar llegint missatge a missatge, processant la informació que conté i atenent-lo, per finalment acabar responent-lo.

Llegeix el missatge part per part, i és l'encarregat de dur a terme l'acció requerida en el missatge, ja sigui una lectura o una escriptura de registre.

Un cop realitzada l'acció ha de respondre, ja sigui amb un error o amb la resposta requerida per el missatge. Per tal de no destorbar al receptor de missatges per a que enviï la resposta, el que hem fet, és guardar el socket per on hem de respondre a la mateixa cua on guardem el missatge, així un cop l'hem processat, serà el mateix processador de missatges qui el respongui.

Té implementat tots els error de les especificacions del protocol Modbus, que és el que utilitzem per a comunicar-nos amb el client.

Els missatges d'escriptura de registre cridaran també a la funció de la llibreria externa H-Link per tal de notificar aquest canvi en un missatge de canvi de dades.

12.2.5. Memòria virtual

En aquest cas, no és tracta d'un thread independent, sinó d'una estructura continguda dins del servidor on guardarem la informació de les unitats simulades.

Permet la lectura i l'edició d'aquestes dades així com la creació o eliminació de les mateixes.

Al arrencar, carrega la informació d'un arxiu XML amb la següent forma:

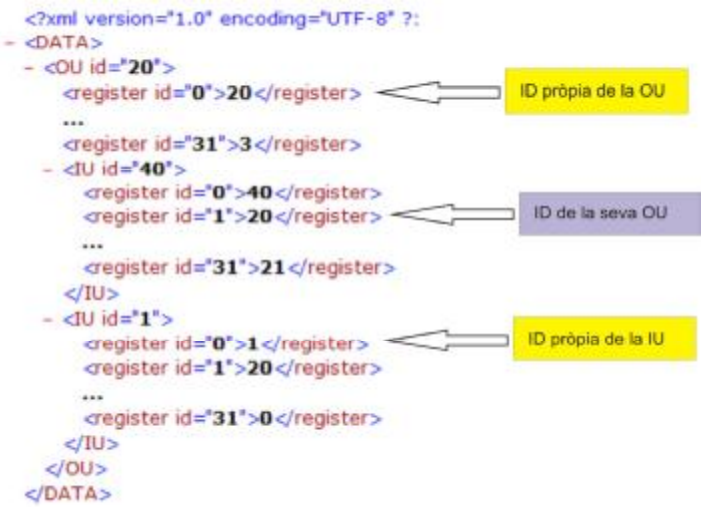

Figura 8: exemple de arxiu XML on guardem les dades de la instal·lació.

La lectura d'aquest arxiu, en funció del mode en el que estigui activat el HARC, llegirà únicament les unitats interiors, les unitats exteriors o ambdues.

La màquina d'estats del lector del XML és estricte. Els registres han d'estar escrits tal qual es mostra en la figura 8, ja que sinó donarà un error. Això ha estat implementat així ja que agilitzava la seva lectura de la qual no era una part del objecte del projecte en la que perdre-hi pas temps, i com aquest XML serà únicament utilitzat internament pel servidor, no ha de tenir problemes.

També s'haurà d'especificar la identitat de les unitats, però això es per un tema de que és més senzill llegir aquest arxiu per una persona en el cas de que haguéssim de mirar el perquè no funciones la lectura d'aquest arxiu.

No es precisa introduir tots els registres de la unitat cada cop, si no ha estat especificat, el lector del XML l'inicialitzarà ell mateix.

Com podem apreciar en la figura 8, entre els tags de les unitats exteriors és declaren les unitats interiors, però tindrem uns registres reservats. Aquest registres seran, en el cas de les unitats exteriors, només el 0, on guardarem la identitat de la unitat, i en el cas de la unitat interior, el 0 pel mateix motiu, i el 1 és per fer referència de quina unitat exterior forma part aquesta unitat interior.

Això esta fet així per facilitar l'edició de la topologia de les unitats, mantenint la coherència del projecte e si, ja que si per exemple volem moure una unitat interior a una altre unitat exterior, únicament modificaríem el valor del registre des de el client, i ja haurem simulat aquest moviment.

D'altre banda això també ens permet l'opció de poder tenir la unitat a una posició però que la seva identitat sigui diferent. Això seria semblant a canviar-li el nom l'únic que només acceptem números.

Un cop finalitza la seva execució, guardarà la informació en el fitxer restablint els canvis realitzats.

El motiu d'haver triat d'utilitzar un arxiu XML per a guardar la informació és bàsicament que és un dels més utilitzats. Podríem haver creat una estructura de fitxer pròpia, però el fet que a Hitachi Air Conditioning Europe S.A. s'utilitzi per a altres aplicacions, ens va convèncer a tirar endavant amb el XML per donar la possibilitat més endavant a poder aprofitar els arxius creats.

Durant la creació del projecte, és va crear un petit script que genera automàticament un XML d'exemple, a mode de poder crear així l'arxiu des de 0 i poder realitzar les diverses probes que podien destruir-lo al llarg de la seva codificació inicial.

Per accedir a la informació de cada unitat i dels registres de control, ho fa utilitzant un únic índex, que és l'índex del registre que rebrem via Modbus.

A la figura 9 que és mostra a continuació es veu com esta repartit els rangs d'aquest índex per a assignar cada part de memòria a un tipus d'unitats:

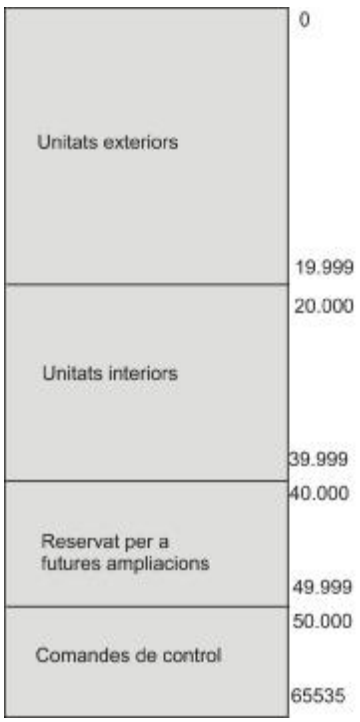

Figura 9: estructura de la memòria virtual.

El primer rang, associa 20.000 registres a les unitats exteriors. Tenint en compte que com a màxim en podem tenir 64, i que cadascuna té 64 registres, estarem utilitzant únicament 4096 registres, però ens guardem marge per si més endavant la capacitat de les comunicacions H-Link augmentessin i per tant poguéssim afegir més unitats.

El segon rang és l'assignat a les unitats interiors, els hi reservem fins a 20.000 registres, en aquest cas com el màxim nombre de unitats interiors que podem tenir és de 160, i també tenen 64 registres, estarem utilitzant com a màxim 10240 registres.

El tercer rang són 10.000 registres que és reserven per a futures ampliacions.

El quart i últim rang, esta assignat a les comandes de control, és un rang molt ampli per a unes poques comandes de control, però com no hi ha hagut problema d'espai, s'ha sobre dimensionat per a facilitar futures ampliacions.

12.2.6 Comunicador H-Link

Aquest thread és l'encarregat de fer de pont entre el servidor i la llibreria externa per a la comunicació H-Link que hem creat.

Recordem que aquesta llibreria ha estat creada compilada prèviament per a evitar que es pugui veure el codi ja que el protocol H-Link és un protocol propietari.

En executar aquest thread, crida a la funció d'establiment de connexió H-Link.

A partir d'aquest moment el thread gestiona dues cues que té la llibreria, una d'enviament i un altre de recepció de missatges.

En aquestes cues el que fem es guardar el numero del registre a canviar i el seu valor. D'aquesta manera el thread només s'ha de preocupar de dues coses. La primera és que quan hi hagi un canvi a la memòria virtual, aquest sigui introduït a les cues de la llibreria H-Link. La segona és anar consultant la cua de missatges H-Link rebuts per tal de realitzar a la memòria virtual els canvis que s'hi indiquin.

Com podem veure, en el codi públic només veurem els moviments dels registres, la part de la construcció dels missatges H-Link i del seu enviament queda oculta dins la llibreria H-Link.

De la mateixa manera, si rebem un missatge H-Link, la funció de recepció de dades de la llibreria ens indicarà que l'hem rebut i ens col·locarà a la cua

12.3. Esquema de l'estructura del software del servidor

En la figura 10, podem veure l'esquema de l'estructura del software del client per a la part de les comunicacions via Modbus. Els diferents conceptes han estat explicats amb anterioritat.

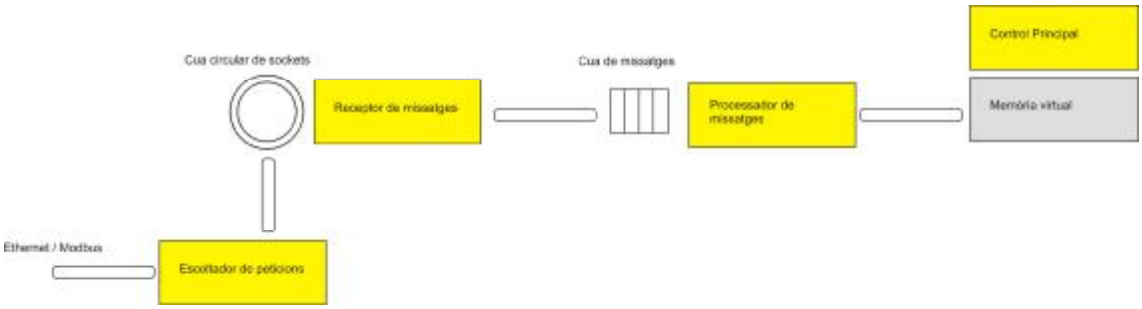

Figura 10: estructura realitzada per a les comunicacions Modbus.

El thread escoltador de peticions s'encarrega d'esperar client, un cop demanem la connexió, l'escoltador de peticions passa al socket al receptor de missatges.

El receptor de missatges segueix l'ordre de la cua circular i va mirant a cada socket si hi ha algun missatge, en cas de ser així aquest missatge serà passat al processador de missatges.

El processador de missatges és l'encarregat de realitzar les operacions que s'indiquen al missatge, consultant o canviant els valors de la memòria virtual.

El control principal és el propietari de la memòria virtual i és l'encarregat de controlar que tots els threads segueixin vius.

En la figura 10, podem veure marcats amb groc podem veure els threads i marcat en gris tenim la memòria virtual, que és troba dins del thread principal. Aquesta llegenda de colors és manté per a la figura 11 on podem veure l'estructura per a la comunicació H-Link.

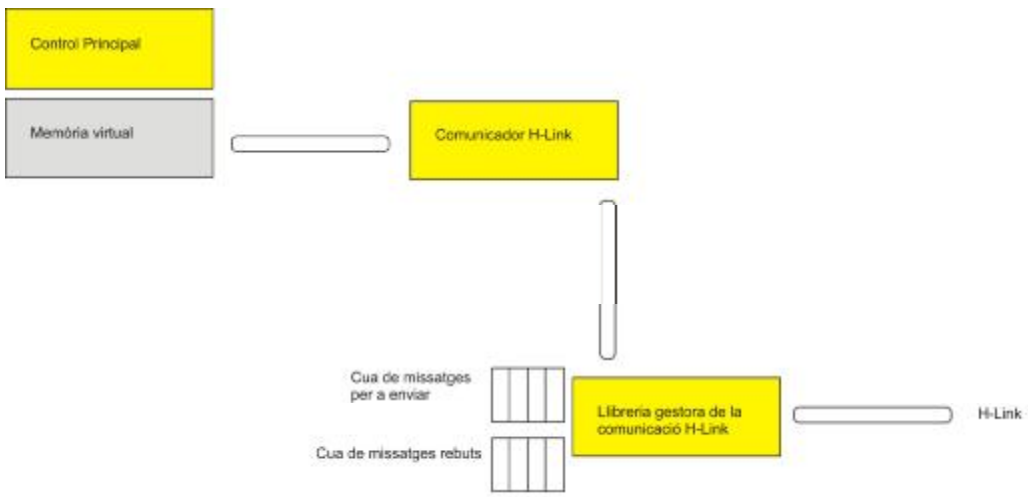

Figura 11: estructura realitzada per a les comunicacions H-Link.

Com ja hem mencionat abans, tant les cues com la llibreria gestora de la comunicació H-Link és la part del codi que ja esta compilada per tal de que no se'n pugui veure el funcionament del protocol H-Link.

El comunicador H-Link és el que es farà de pont tant en la recepció d'ordres processades per la llibreria H-Link com en la introducció d'ordres a la mateixa.

12.4. Implementació d'errors

El que pretenem és permetre a l'usuari introduir errors en les unitats simulades.

Es poden pensar diverses tipologies d'errors, però per a aquest projecte se'n van plantejar únicament dos.

La primera es inhabilitar la comunicació de la unitat, fer que no enviï missatges de resposta cap al CSNet Web. Aquesta inhabilitació pot ser total o parcial, es a dir, el simulador pot no respondre únicament a els missatges que el CSNet Web faci a els registres que seleccionem, i evidentment, podem seleccionar-los tots.

El segon error plantejat consisteix en fer que el CSNet Web no pugui governar la màquina simulada. Es tracta de que les comunicacions siguin les correctes, però la màquina no faci cas de les ordres rebudes.

En els dos casos, el que es pretén es veure com reacciona el CSNet Web contra aquestes adversitats.

13. El client

13.1. Introducció

El client es l'encarregat de dotar-nos d'una interfície agradable per a l'usuari. El client ha de fer simple i automatitzar, les accions que farà, alliberant a l'usuari de la part pesada de la feina.

Per tal de que al usuari li sigui més familiar, se li ha donat semblança pel que fa a icones i estructura amb el software client del CSNet Web.

Com ja hem explicat abans, el client és qui tindrà el control de la comunicació Modbus amb el servidor. Per a aquesta comunicació s'ha utilitzat una llibreria ja implementada de Java, anomenada *ModbusTCPMaster* en la qual, hi trobem implementades totes les comunicacions bàsiques del protocol Modbus.

El client ha estat programat utilitzant Java swing que simplifica molt les feines d'elaboració d'interfícies gràfiques. Aprofitant aquesta versatilitat donada per swing el client ha estat enfocat dos modes de treball, el controlador, i el visualitzador de dades.

El controlador fa la funció que realitzaria un comandament a distancia respecte una unitat d'aire condicionat real. Té diferents camps que poden ser modificats i enviats al servidor per a que ho modifiqui en la memòria virtual que hi hem implementat.

Figura 12: pantalla inicial.

El client s'ha de mantenir actualitzat en tot moment per si algun altre client connectat al servidor modifiques alguna de les dades, per això s'utilitza un thread de refresc que serà explicat més endavant.

Aquest thread també serveix per a controlar si la connexió segueix vigent per a notificar-ho al usuari si hi hagués algun problema.

13.2. Disseny de d'interfície

El primer punt en la construcció de la interfície del client va ser el fet de decidir com mostrar la informació. Teníem dues necessitats diferents, la de poder controlar cada unitat i la de poder veure l'estat de les unitats. Per a això es van crear dos modes de visualització de la informació que són el visualitzador i el controlador que seran explicats més endavant.

L'altre gran qüestió era com mostrar la topologia per a que l'usuari trií la unitat, teníem dues opcions que era en mode de llista o en mode d'arbre.

El mode llista en instal·lacions grans faria més complexa la cerca d'una unitat, mentre que l'arbre escurça la llargada d'instal·lacions grans i ens dona una imatge més semblant a la que té la instal·lació real ja que de cada unitat externa en penjaran les unitats internes de que disposi.

Es per aquestes dues raons que varem triar l'opció del arbre per a que l'usuari selecciones quines unitats vol veure.

A continuació, a partir de la figura 13, s'explica com ha quedat estructurada la interfície:

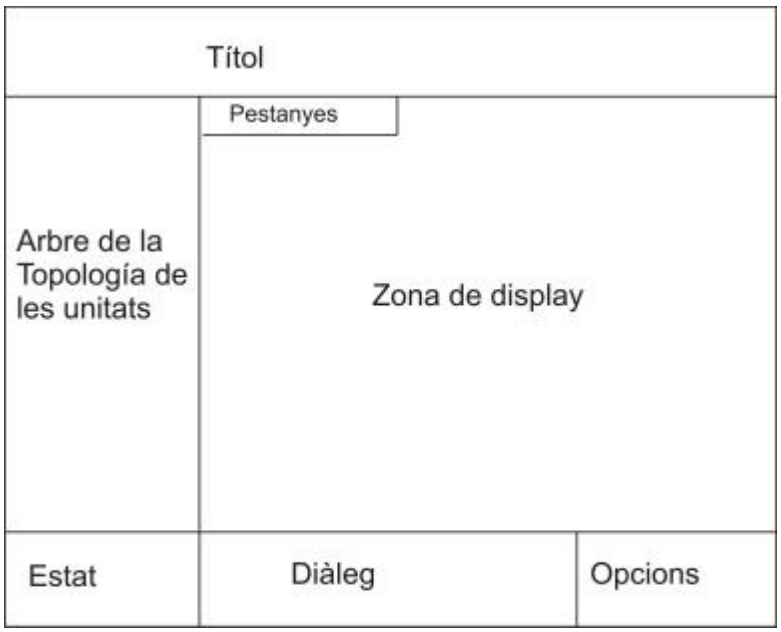

Figura 13: Estructura de la interfície.

Com podeu veure a la figura 13, la pantalla del client estarà dividida en 7 parts diferents.

La primera i menys important part és la part del títol, que no és més que una simple capçalera que li dona un toc més agradable a la vista.

A la part de sota de la pantalla, i trobem 3 parts en la que definirem la zona de comunicacions.

En aquesta zona, trobem l'àrea de la informació del estat de la connexió que ens permetrà connectar i desconnectar, així com saber l'estat de la connexió mitjançant els següents icones:

X : Error en connexió

 : Connectant

 : Connectat

En el cas de no aparèixer icona vol dir que no hi ha connexió i que tampoc esta intentant establir-la.

L'àrea de diàleg, ens permetrà visualitzar tot tipus de notificacions que el client ens volguí fer, des de notificacions d'errors, fins a simples missatges de enviament de missatge correcte.

La tercera part de la zona de comunicacions és la que conte les opcions generals de control del servidor i respecte al programa del client mateix.

En aquesta zona i trobem 4 botons per a les principals accions:

- Save: envia la senyal al servidor per a que guardi la topologia actual en el fitxers a mode que a partir d'aquest moment sigui la topologia per defecte.
- Configuració IP: permet introduir la IP del HARC al que volem accedir. Aquesta IP es guarda en un arxiu XML en el client per tal de poder-la llegir al arrencar el programa la propera vegada.
- Reset: envia una senyal al servidor per a que es reinicií.
- Exit: finalitza l'execució del programa client.

La franja més important de la interfície és la que ens trobem al mig de la pantalla que és la que ens permetrà triar i visualitzar la informació.

Per a triar-la utilitzarem l'arbre de la topologia de les unitats, que ens permetrà veure d'una forma clara i ordenada quina estructura té la nostra xarxa. Un cop seleccionada la informació, serà mostrada a la zona de display.

La informació que és mostra a la zona de display es diferent en funció del mode de funcionament que hem triat en el client. Aquest mode es selecciona triant una pestanya o una altre.

Si estem en el mode de control, veurem la informació d'una sola unitat a mode de comandament a distancia, i si estem al mode de visualització veurem la unitat exterior i totes les interiors de la unitat que haguem seleccionat.

A la zona del arbre de la topologia de les unitats trobem 3 botons que ens permetran afegir o eliminar unitats.

13.3. Parts que el formen

13.3.1 ModbusTCPMaster

Com ja hem comentat abans, es una llibreria feta de Java que ens permet les comunicacions Modbus, el que hem fet amb ella és crear una classe que n'aprofiti el que volem.

Inicialment s'estableix la connexió, i en cada pas de missatges, si no pogués comunicar-se guardem una variable que ens indica que hem perdut la connexió.

Només s'han implementat les funcions *readMultipleRegister* i *WriteRegister*, ja que mai necessitarem escriure més d'un registre a la vegada, i a la hora de llegir-ne pot ser que en necessitem un o més, però ho podem fer tot amb la funció *readMultipleregister*, ja que d'aquest manera la lectura d'un registre serà igual de rapida, però en el cas de llegir més d'un registre, reduirem l'overhead de les capçaleres.

Per fer més agradable l'ús d'aquestes funcions s'han desenvolupat funcions concretes que agilitzi i optimitzin des de la lectura dels registres d'una unitat, com a la lectura de varies unitats consecutives.

13.3.2. Estat del sistema

És la pantalla inicial que ens trobem quan ens connectem o bé quan seleccionem la arrel de la nostre topologia.

El que ens permet, és visualitzar l'estat dels registres de lectura del servidor que ens informen de quin és l'estat i la situació del sistema actual.

Figura 14: Pantalla de visió del estat del sistema.

13.3.3. Visualitzador

El visualitzador ens permet veure l'estat de tots els registres d'una unitat exterior i de tots els de les seves unitats interior. Per a fer-ho utilitza taules que mostren de manera més clara la informació.

Aquesta informació és editable, tret dels 2 primers registres que són considerats restringits i fixes per a cada unitat.

La informació es pot seleccionar, de manera que es pot copiar la informació a algun arxiu si ens fos necessari.

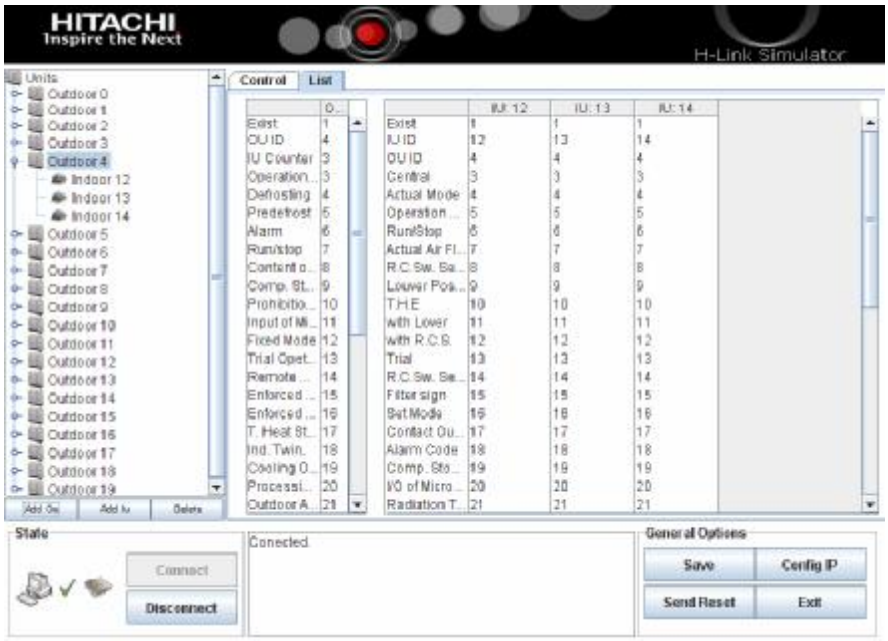

Figura 15: Pantalla del mode visualitzador.

La taula de les unitats interiors creix i decreix en funció del nombre d'unitats interiors que tingui la unitat exterior en cada moment.

Ambdues taules tenen barra lateral independent per tal de poder visualitzar els registres que vulguem ja que aquest registres identifiquen atributs diferents en funció de si es una unitat exterior o interior.

13.3.4. Controlador

El controlador ens permet actuar de cara a la unitat en qüestió com si estiguéssim utilitzant un controlador remot.

Figura 16: Pantalla del mode controlador.

A la figura 16 veiem com en la implementació del controlador se li ha donat semblança als típics controladors remots. En aquest cas ens hem basat en un de la casa Hitachi.

Per dotar de robustesa al programa s'ha afegit rutines que comproven els valors dels registres de manera que en cas de introduir un caràcter il·legal, es mostraria un missatge d'error i no s'enviaria aquest valor al servidor.

13.3.5. Thread refrescador

El Thread refrescador s'encarrega d'anar consultant al servidor les dades dels registres que es mostren. Si el valor d'aquets registres ha canviat, els canvia a la interfície gràfica, sinó han canviat dorm durant 1 segon i torna a començar.

El valor d'un segon com a espera entre cada consulta va ser triat assumint que era suficientment gran com per no passar més temps esperant la resposta del servidor que atenent al usuari, i suficientment petit com per tenir les dades actualitzades en tot moment.

13.4. Funcionalitats implementades

13.4.1. Gestió de la informació

El client ha de tenir al dia la informació que el client guarda. Per a fer-ho, la primera cosa que hem de decidir es com guardar aquesta informació.

Tenim bàsicament dues opcions, la primera seria guardar la mateixa forma en que hem implementat la memòria virtual del client, i guardar un array amb les unitats exteriors i un altre amb les interiors. Però donat que el Java és un llenguatge molt potent em aprofitat per a que cada unitat exterior tingui el seu propi array de unitats interiors.

S'ha triat la opció de tenir un array de unitats interiors per a cada unitat exterior ja que la cerca d'una unitat interna serà en general més ràpida per què no haurà de recórrer tot l'array.

Per rebre les dades del servidor, s'han implementat diverses funcions que utilitzant la llibreria *modbusTCPmaster*, li demani al servidor la informació desitjada.

Utilitzant aquestes llibreries agafem la informació de les unitats i generem la interfície gràfica que se'ns mostra. Per a mantenir aquesta informació al dia es van valorar dues opcions:

- Crear una comunicació Modbus també en la direcció oposada a la existent per a que el servidor notifiques els canvis.
- Que els clients es mantinguessin al dia per enquesta, es a dir, que vagin preguntant l'estat dels registres.

La primera opció trencava la idea de la comunicació Modbus de màster i esclau, a part de que carregaria al servidor molt més de feina, i com funciona sobre una maquina no potent, ens varem decantar per la segona opció.

Per a dur a terme la segona opció es va crear un thread que refresques la informació que estàvem mostrant cada segon. Aquest thread el que fa es preguntar els valors dels registres que estem ensenyant a la pantalla al client.

Per a evitar que cada segon se'ns reconstruís la pantalla impossibilitant així que el client fes cap acció, el que es va optar per fer va ser crear unes rutines en que si els valors dels registres mostrats no haguessin canviat, no es reconstruiria la pantalla mostrada, que només ho faria en cas que si hagués canviat.

Aquesta opció podria tenir una pega en el cas ens connectéssim molts clients a un servidor, que com te una cua d'atenció circular, podria donar al usuari la sensació d'estar desatesos. Però com la idea no es que hi hagi un gran nombre de clients connectats a un servidor, aquest problema no es rellevant.

L'error podria ser crític si el temps que triga el servidor en fer cas 2 cops a un mateix client fos superior al timeout del protocol Modbus, en aquest cas el client creuria que ha perdut la connexió i ho mostraria pel diàleg de la interfície. D'aquesta manera ens

cobrim les esquenes de cara a un error que no esta previst que doni mai, però que en el cas de donar-se no provocaria un error no controlat al programa del client.

13.4.2. Optimització en la lectura de grans blocs de dades del servidor

En l'execució del client em trobat una gran dificultat en fer que el client estigues sempre actualitzat i que sempre donés sensació d'atenció al usuari.

Al principi de la codificació d'aquest projecte, quan el servidor corria sobre una màquina virtual en Linux, varem notar que el temps de les comunicacions era suficientment gran com per ser una mica molest i per això varem implementar aquesta optimització.

Un cop el servidor va córrer sobre el HARC, els temps de comunicació es van fer pràcticament imperceptibles, però aquesta millora ens assegurava que en cas de problemes amb la velocitat de connexió, el client donés una resposta més que acceptable.

Aquesta optimització el que intenta és agilitzar la consulta de la informació de molts registres. S'ha optimitzat el pas de missatge perquè entre enviament del missatge i espera de resposta perdíem massa temps.

El que es va fer es reduir el nombre de cops que necessitàvem enviar un missatge, augmentant doncs la quantitat de informació que s'envia cada cop.

El protocol Modbus permet que el màxim nombre de registres llegit d'un sol cop sigui de 7D en hexadecimal és a dir de 125 registres. Com per a cada unitat guardem 64 registres varem decidir crear una funció que els llegís anés llegint de 120 en 120 registres i assignant-los a la unitat corresponent. Es va agafar el nombre 20 en comptes del 125 degut a que les llibreries de Modbus del Java estableixen la limitació en 121 i varem preferir fer números rodons.

Aquesta optimització va dividir el temps de lectura de totes les unitats pràcticament entre 2. Només ens serveix per això, per a la lectura de registres consecutius, es a dir unitats exteriors consecutives i unitats interiors consecutives.

En tot moment el servidor mante 2 registres que indicant el nombre de unitats internes i externes que té, que el client pot consultar abans de llegir grans blocs de dades per a saber fins a on ha de llegir i d'aquesta manera evitar passar-nos del rang de valors existents en la lectura i rebre d'aquest mode un error.

13.4.3. Control del simulador

Per a poder tenir control sobre l'execució del software que tenim al HARC, es van deixar uns registres restringits, mitjançant el quals, es poden donar ordres de control des de el client.

Utilitzant això, dotem al usuari del client amb la possibilitat d'afegir i treure unitats a la topologia, fer recomençar al HARC, guardar l'arxiu de la topologia o conèixer valor de comptadors generals per tal d'agilitzar les comunicacions amb el client.

Aquestes ordres serien conegudes per la resta de clients connectats al HARC ja que utilitzem comptadors per a tenir al dia l'estat del sistema que te el client i el que te el HARC, i en cas de que hi hagi diferencia, el client s'actualitzarà.

14. Proves i Resultats

14.1. Proves de robustesa

14.1.1. Prova 1: Pèrdues de connexió

La primera prova realitzada en aquest apartat ha consistit en desconnectar el servidor quan el client hi estava treballant, ja fos refrescant o construint l'arbre de la topologia. En ambdós casos, ha detectat correctament que s'ha perdut la connexió, i ho ha mostrat pel diàleg.

Realitzar aquesta primera prova a la inversa no té sentit ja que com la comunicació segueix el patró master-slave, el servidor respon, però no es preocupa de si la informació li arriba al client.

La segona part d'aquesta prova consistia en provar que passava si el client no es podia connectar, ja que el servidor estava apagat d'inici. En aquest cas, i seguint el guió previst, el client va provant de tornar a connectar-s'hi cada segon, i va mostrant pel diàleg missatges de que la connexió no ha estat possible.

14.1.2. Prova 2: Prova de saturació del servidor

La intenció d'aquesta prova era detectar possibles errors, o augments de temps que els clients han d'esperar per a ser atesos.

La prova ha consistit en connectar 15 clients a la vegada i veure que passava.

El resultat no ha decebut, el temps d'espera segueix igual, o si augmenta és imperceptible, el servidor no s'ha col·lapsat, i tots els clients s'han mantingut refrescant la seva informació durant 10 minuts sense problemes.

14.1.3. Prova 3: Caràcters il·legals i fora de marge

S'ha provat de introduir caràcters il·legals o valors fora de marge en els camps editables de les taules i dels diàlegs del client, i el resultat ha estat excel·lent ja que el client esta creat en mode de llista blanca, només accepta uns caràcters concrets, per a qualsevol anomalia mostra un missatge d'error.

14.2. Proves de coherència

14.2.1. Prova del controlador

Aquesta prova el que ha consistit és en obrir 2 clients, un d'ells amb la pestanya del controlador oberta. El que s'ha fet és modificar registres des de l'altre client utilitzant el controlador i el visualitzador, i veure que en el controlador inicial rebíem els canvis fets a l'altre client.

El resultat de la prova ha estat satisfactori ja que en tots els casos el controlador del primer client ha rebut les modificacions.

14.2.2. Prova del visualitzador

L'objectiu d'aquesta prova es semblant al anterior però te la diferencia de que en aquest cas, veurem els canvis en el visualitzador, és a dir, des de un altre client modificarem registres tant des de el controlador com des de el visualitzador, i veurem a les taules del primer client com els valors canvien.

El resultat d'aquesta prova també ha estat el desitjat ja que en tots els casos hem pogut veure els canvis realitzats a l'altre client.

14.3. Prova de funcionament final del simulador

Per a comprovar que el simulador ha complert el seu objectiu principal, ens trobàvem amb la prova de foc. Era la hora de connectar-lo al CSNet Web.

Les primeres tomes de contacte no van ser bones, hi havien errors en la programació de la llibreria H-Link que ens permet comunicar-nos amb CSNet Web, un cop depurats els errors ens varem disposar a tornar-ho a provar i el resultat va ser l'espera't.

En aquest test el que varem simular va ser 50 màquines externes i 120 de internes. El CSNet Web les va detectar i ens va permetre treballar amb elles.

A la figura 17 podem veure una captura de pantalla del simulador quan estava connectat amb el CSNet Web.

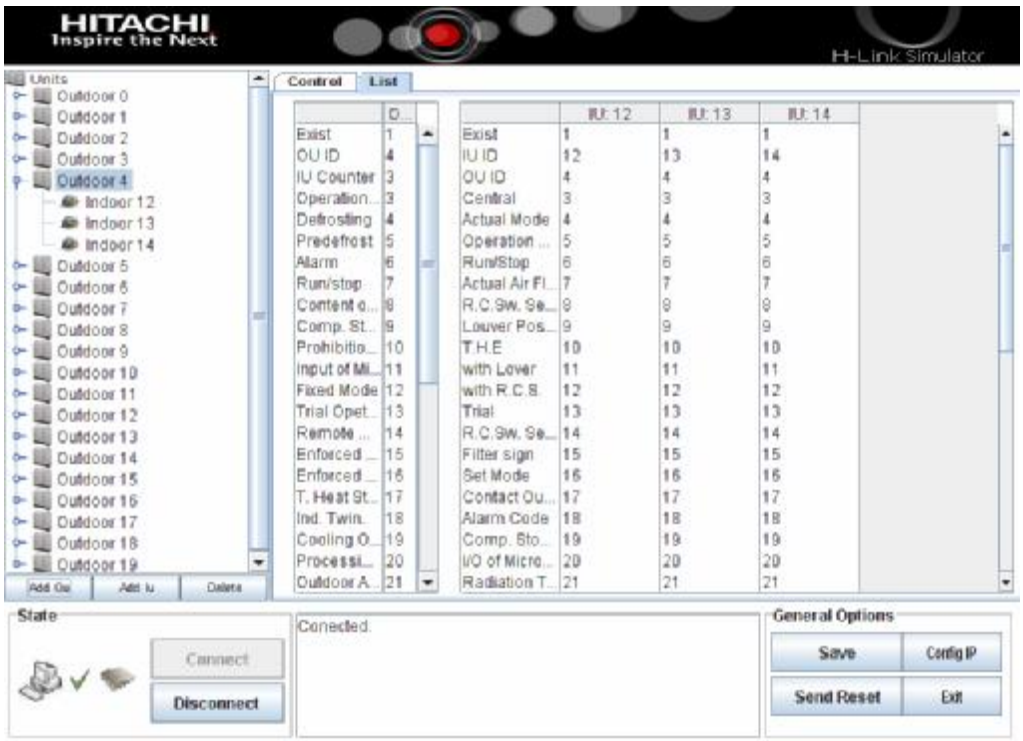

Figura 17: Captura de pantalla del simulador durant la prova.

A continuació podem veure una captura de pantalla del resultat que ens va mostrar el CSNet Web amb la simulació que li varem introduir.

Figura 18: Captura de pantalla del CSNet Web utilitzant les màquines simulades

Ja s'ha comentat abans al capítol del client però, aprofitant la figura 18, recordem que hem pres l'estètica de la interfície client basant-nos amb el CSNet Web.

Per a acabar de comprovar que el resultat era el desitjat faltava provar de manipular registres des de el client del simulador i que aquest fossin llegits per al CSNet Web. Les proves van ser un èxit, en canviar un valor al servidor, aquest envia sense problemes el missatge de dades canviades que el CSNet Web va llegir i va mostrar els canvis en la seva interfície.

15. Conclusions

La conclusió principal és que el projecte ha complert amb tots els objectius principals establerts:

- S'ha creat el software que simula l'existència d'una xarxa d'unitats d'aire condicionat de la marca Hitachi.
- S'ha creat el software que permet visualitzar i configurar l'estat del simulador des d'un PC.
- S'ha elaborat la interacció entre el simulador i el CSNet Web.

Tots els objectius aconseguits han estat degudament provats com s'ha pogut veure en el resultat de proves i resultats.

S'han satisfet les necessitats que tenia el simulador que ara està a la espera de la seva instal·lació permanent en la sala de Training de Hitachi.

La decisió de separar el projecte en dues parts ha estat molt encertada ja que ens ha permès anar evolucionant cada una per separat, sense que els problemes d'una perjudiquessin a l'altre.

Guardar les dades dels registres en un fitxer XML ens ha permès depurar el codi ja que podíem comprovar de manera entenedora quin valor hauria de tenir el registre.

La planificació temporal es va seguir sense variacions importants tret de en l'elaboració de la part del H-Link que va provocar que no hi hagués temps per a poder assolir l'objectiu secundari de introduir errors en les màquines simulades.

L'elecció del Modbus com a protocol de comunicacions entre client i servidor ha estat un èxit ja que ens ha permès tenir unes comunicacions senzilles i eficients sense necessitat de tenir que crear un protocol propi.

Les cues circulars d'atenció al client que té el servidor han servit per no inundar el HARC amb molts threads sense perjudicar la sensació d'atenció que te el client.

L'ús de Java Swing ha facilitat molt la creació d'una interfície per al usuari simple i entenedora.

La optimització per aprofitar el màxim els missatges Modbus va solucionar els problemes sorgits amb els colls d'ampolla a l'hora de refrescar les pantalles del client que devien esperar a que les comunicacions acabessin.

El problema del copyright del protocol H-Link va estar degudament salvat utilitzant un fitxer binari que tingues la funcionalitat desitjada però que amagués el codi.

15.1. Possibles ampliacions

15.1.1. Ampliació 1: mode de funcionament del HARC

El servidor ha estat programat inicialment per a simular tant les unitats interiors com les exteriors en un sol HARC, però com ja es va comentar en la introducció del HARC, la programació ja estava orientada a la possibilitat d'utilitzar 2 HARC's i d'aquesta manera poder simular via H-Link, els missatges que hi ha entre unitats interiors i exteriors.

El HARC ha de tenir 3 modes de funcionament:

- Simulador d'unitats interiors
- Simulador d'unitats exteriors
- Simulador de ambdós tipus d'unitats

El tipus de funcionament de ambdós tipus d'unitats és el implementat per defecte, i el que proposa aquesta millora és implementar els altres 2 modes. Per tant en funció del mode, el HARC carregarà la informació d'un tipus d'unitat o de les 2, i respondrà únicament per a aquestes unitats.

15.1.2. Ampliació 2: Simulació d'errors

Aquesta ampliació consisteix a complir l'objectiu secundari que no s'ha complert durant el desenvolupament d'aquest projecte.

El que es vol es poder generar error a les unitats simulades al servidor.

Els possibles errors podrien ser:

- Pèrdua de connexió
- Pèrdua de control de la unitat

15.1.3. Ampliació 3: fitxer de log

L'objectiu d'aquesta ampliació és poder tenir un accés a la informació generada en cada moment per al servidor. El interès d'això és poder depurar possibles errors ja existents o que es produeixin en futurs intents d'ampliació del software.

Ara mateix, per a poder depurar el servidor ara només existeixen dues maneres que son:

- executar-lo en un PC i veure les dades per la consola.
- connectar-s'hi utilitzant ssh i executar-lo per a poder veure les dades per la consola ssh.

Aquesta ampliació proposa fer que el client tingui un accés més fàcil a aquesta informació. S'han pensat dues possibles solucions.

La primera consisteix a que el servidor guardi la informació de debug en un arxiu que el client pugui consultar després per telnet. Per a dur a terme aquesta versió s'ha de tenir en compte que la memòria flash del HARC no permet gravar sobre ella molt ràpidament ja que correríem el perill de fer-la mal be, així que hauríem de buscar alguna altre opció com guardar-ho cada un cert temps, i no deixar que aquesta informació creixi molt en volum.

La segona opció podria ser que el servidor envies per un socket la informació de debug als clients, ja fos utilitzant un grup multicast o enviant client a client aquesta informació que seria mostrada per el diàleg del client que podria tenir un botó de canvi de diàleg per veure el del client o el del servidor.

15.1.4. Ampliació 4: Test del protocol H-Link

L'objectiu d'aquesta ampliació és poder realitzar un test exhaustiu del funcionament del protocol H-Link. Per a fer-ho el que volem es poder recrear una instal·lació real i complexa.

Fins al final d'aquest projecte hem simulat aquesta instal·lació però només de cara al CSNet Web, amb aquesta ampliació el que volem és posar un HARC per a cada unitat simulada i tots ells connectats com una instal·lació real usant el bus H-Link.

El fet d'utilitzar només HARC's resulta molt més econòmic que no instal·lar màquines reals i com el pas de missatges seria idèntic, podríem realitzar un test al protocol i al bus H-Link.

16. Bibliografia

System.c:

Pàgina sobre el SystemC:
http://en.wikipedia.org/wiki/SystemC
[Autor: usuaris wikipedia] [Última visita: 2 de desembre del 2007]

Tutorials del SystemC:
http://www.doulos.com/knowhow/systemc/
[Autor: Doulos Ltd.] [Última visita: 5 de desembre del 2007]

Java:

Api de Java:
http://java.sun.com/j2se/1.5.0/docs/api/
[Autor: Sun Microsystems, Inc.] [Última visita: 13 de juny del 2008]

Pàgina de descarrega de Java:
http://www.java.com/en/download/index.jsp
[Autor: Sun Microsystems, Inc.] [Última visita: 22 de maig del 2008]

C/C++:

Tutorials i exemples de codi en C/C++:
http://www.cplusplus.com/
[Autor: cpluspluc.com] [Última visita: 26 de gener del 2008]

Protocol Modbus:

Pàgina d'informació general de Modbus:
www.modbus.org
[Autor: Modbus-IDA] [Última visita: 9 de maig del 2008]

Especificacions del protocol Modbus:
Modbus_Application_Protocol_V1_1b.pdf
[Autor: Modbus-IDA] [Data de descarrega: 9 de maig del 2008] [Annex nº]

Eclipse:

http://www.eclipse.org/
[Autor: Modbus-IDA] [Data de descarrega: 10 de febrer del 2008]

Protocols d'Internet:

Protocolos de internet. Diseño e implementación en sistemas UNIX
[Autors: Ángel López González y Alejandro Novo López] [Editorial: Ra-Ma.]
[Any: 1999]

Signatura del Autor:

Resum

Aquest projecte intenta donar solució al problema amb el que s'ha trobat la empresa Hitachi Air Conditioning Products SA a la hora de fer probes durant el desenvolupament del seu principal producte, el CSNet Web.

El simulador que s'ha realitzat en aquest projecte pretén reproduir la topologia d'una xarxa d'unitats d'aire condicionat per a que el CSNet Web les interpreti com si aquestes màquines fossin reals. Per a aconseguir-ho, el simulador reproduirà les comunicacions entre aquestes màquines i el CSNet Web.

Resumen

Este proyecto intenta dar solución al problema que se ha encontrado la empresa Hitachi Air Conditioning Produtcs Europe SA a la hora de hacer pruebas en el desarrollo de su principal producto de control de máquinas de aire acondicionado, el CSNet Web.

El simulador que se ha realizado en este proyecto, pretende reproducir la topología de una red de unidades de aire acondicionado para que el CSNet Web lo interprete como si esas maquinas fueran reales. Para ello, el simulador reproducirá las comunicaciones entre estas máquinas con el CSNet Web.

Summary

This project tries to solve to the problem that Hitachi Air Conditioning Produtcs Europe SA company found when they makes test during the development of its principal product of air conditionating control machines, called CSNet Web.

The simulator created in this project, tries to reproduce the topology of a network of air conditioning machines. CSNet Web interprets this topology as real machines. In order to achieve this goal, this simulator will reproduce the communications between these machines and CSNet Web.