Quina tecnologia utilitzar per implementar el Servidor de Coneixement?

Marc Vall-llosera i Jordi Torres
Càtedra Telefónica-UPC
http://www.upc.es/web/CatedraTelefonicaUPC/

Working Report WR-2004-02, setembre 2004

Resum. Aquest report pretén proposar una infrastructura tecnològica que permeti implementar les idees i conceptes explicats en el Working Report WR-2004-01 del *Knowledge Infrastructure Lab*. Comencem amb un estudi de les tecnologies d'aprenentatge en xarxa o P2P que ens poden ajudar a aconseguir els nostres objectius, fins a arribar a proposar la nostra pròpia infrastructura per l'aprenentatge en xarxa.

ÍNDEX

1. Aprenentatge en xarxa i P2P

1.1 Tecnologies estudiades

Tot seguit estudiarem diferents tecnologies o infrastructures ja existents que adaptant-se a algun dels models P2P anteriors ens ajuden ja sigui a crear un entorn de col·laboració o aprenentatge en xarxa o bé simplement a permetre una comunicació P2P entre usuaris de la xarxa. L'objectiu és analitzar aquestes opcions tecnològiques i trobar el camí definitiu per desenvolupar la nostra pròpia infrastructura d'aprenentatge en xarxa.

1.1.1 ANTS

El projecte **ANTS** [19] [20] [27] [28] és el resultat de la tesis doctoral de Pedro García i ha estat desenvolupat a la Universitat de Murcia i a la Universitat Robira i Virgili. El sistema ANTS vol proporcionar un entorn de treball col·laboratiu i multi-usuari.

1.1.1.1 Què és ANTS?

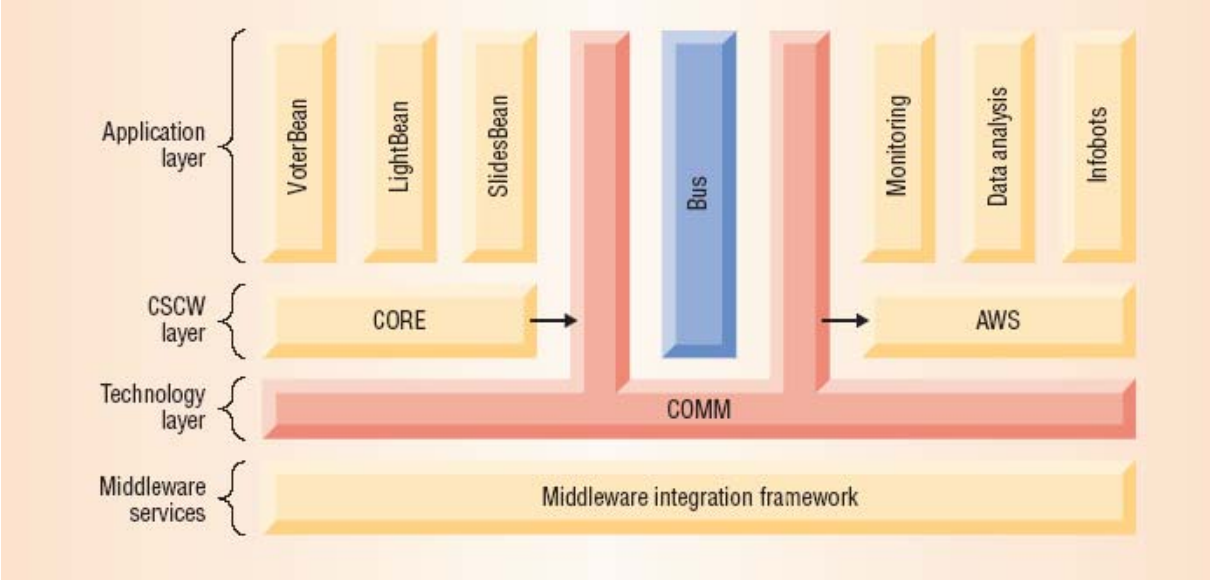
El projecte ANTS, parteix de les noves oportunitats ofertes per Internet gràcies als avenços en informàtica distribuida així com a les millores en les xarxes de comunicació. Aquestes noves oportunitats donen lloc a noves àrees d'investigació entre les que cal destacar l'àrea anomenada *Computer Supported Cooperative Work (CSCW)*. L'objectiu principal de CSCW és descobrir noves fórmules d'utilitzar les tecnologies informàtiques per millorar els processos d'interacció, comunicació i treball en grup; també anomenat *groupware*. Així doncs, el projecte ANTS el podem situar en aquest àmbit.

Un dels objectius inicials de la plataforma resultant és que aquesta pugui ser la base de sistemes d'aprenentatge col·laboratiu que formarien part de l'àrea de *Computer Supported Cooperative Learning (CSCL)*. Com veiem, l'àmbit d'aquest projecte està molt relacionat amb les nostres idees i amb la creació de comunitats d'aprenentatge virtual.

Per tant, podem definir ANTS com un marc de treball dissenyat per proveir una base sòlida pel desenvolupament d'aplicacions CSCL. Per una banda, el marc de treball està construit sobre les tecnologies punteres com J2EE, serveis de notificacions i XML. Per l'altra, ANTS proporciona uns servies de CSCW sòlids que fan el sistema robust i extensible.

1.1.1.2 Arquitectura

L'arquitectura global d'ANTS consisteix en tres capes principals *(Il·lustració 1.1)* que són: capa tecnològica, capa CSCW i capa d'aplicació.



Il·lustració 1.1 - Arquitectura ANTS

Capa tecnològica

Com a infrastructura sòlida que proporciona seguretat, escalabilitat, persistència, transaccions i rendiment s'utilitza J2EE. Això permet que la seva infrastructura sigui independent del venedor del sistema; podem escollir qualsevol servidor d'aplicacions que compleixi les especificacions J2EE.

Pel que fa a la gestió de la base de dades, el sistema també és genèric tot i que ha estat provat amb HypersonicSQL i Oracle8i.

Referent al *middleware* orientat a transmissió de missatges es necessita un servei de publicació/subscripció de notificacions pel marc de treball. La plataforma ha implementat una API que permet escollir entre qualsevol solució que s'adapti a JMS *(Java Message Service)* o el servidor de notificacions Elvin. Per últim, tota l'arquitectura utilitza XML per intercanvi d'informació i XSLT per la seva visualització.

Capa CSCW

Aquesta és la part més important de tot el sistema. Està dividida en dos mòduls principals: ANTS CORE i ANTS AWS interconectats pel bus de col·laboració.

1. **ANTS CORE**

   Aquest mòdul inclou suport explícit per sessions, artefactes compartits, control de coordinació, seguretat i integració per aplicacions síncrones i asíncrones. El seu disseny està fortament influenciat per les arquitectures extensibles i genèriques MOO *(Multiuser object-oriented)*.

2. **ANTS AWS**

   La infrastructura de conciencia en el costat del servidor permet disparar un conjunt d'actuadors que realitzen unes determinades tasques com a resposta a certs esdeveniments captats per sensors. Els sensors representen qualsevol component software o hardware que produeix i transmet events cap al sistema de notificacions.

   AWS proporciona com veiem, les bases per un sistema sofisticat de monitorització d'aplicacions, agents intel·ligents i actuadors activats per events rebuts del sistema de notificacions.

Capa d'aplicació

El sistema és suficientment genèric pel desenvolupament de components col·laboratius i aplicacions o agents de monitorització sobre ell, aconseguint una suau transició entre aplicacions locals i distribuides. El sistema ofereix facilitats per la creació de nous components així com la creació de capturadors d'events i actuadors que responen a aquests.

Alguns exemples d'aplicacions desenvolupades en aquesta capa poden ser:

- *MapTool*: una pissarra digital compartida
- *Pong*: joc multiusuari
- *Client MOO:* interfície de text MOO
- *JLE (Java Learning Environment)*: un entorn col·laboratiu d'aprenentatge que inclou serveis com gestió de cursos, avaluacions, seguiment del progrés, etc.
- *MOVE:* un entorn col·laboratiu virtual que permet a humanoids virtuals intractuar els uns amb els altres. També els permet interactuar amb diferents components

com poden ser presentacions o simulacions 3D. Els humanoids i les escenes estan representades amb VRML.

- *BSCL (Basic Support for Cooperative Learning):* adaptació i integració d'una part de la plataforma ANTS al BSCW per millorar aquest sistema de treball en grup asíncron permetent una monitorització adequada i donant suport a la col·laboració síncrona i a les sessions compartides. Es va integarar la pissarra digital al BSCW. Aquest treball, s'emmarca dins el projecte europeu ITCOLE.

1.1.2 Simple Discovery Server

En aquest cas, ens fixem en com podem desenvolupar un dels models P2P anteriorment vistos per trobar els *peers* dins la xarxa. No estudiem una plataforma d'ajuda a la creació d'entorns d'aprenentatge; només una manera de trobar els *peers* dins la xarxa. Així, ens centrem clarament en una infrastructura que es correspon amb el segon model: **P2P amb un simple servidor de localització**. Adaptat al nostre cas podria consistir en el següent:

a. Un servidor de localització que ens permetés allotjar una base de dades Accés i ens permetés executar ASP`s. En aquest estat inicial d'estudi proposem aquesta opció tot i que podríem escollir-ne d'altres com un servidor amb una base de dades MySQL i amb intèrpret de PHP.
b. Aquest servidor seria aliè a la comunitat i només tindria la funció de localitzar els participants connectats. Per tant, hauria de tenir disponibilitat 7x24

Com veiem, aquesta opció té l'inconvenient de necessitar un servidor que ens ofereixi el que necessitem i amb una alta disponibilitat. Per resoldre-ho podríem destinar un PC de la comunitat amb una IP fixa a aquestes tasques o bé podríem buscar algun servidor per Internet.

- En aquest moment –i per fer les proves- http://freeasp.us ens ofereix el que necessitem:
  - Free ASP Hosting que inclou suport MS Access, accés FTP, suport Flash i ASP 3.0
  - Tot això a l'adreça http://freeasp.us/yourname
  - 2 GB d'ampla de banda i 50 MB d'espai
  - Sense popus ni banners

Així doncs, de cares a les primeres proves utilitzarem aquest servidor. Pel que fa al funcionament i la interacció entre els participants *(peers)* i el servidor la podríem separar en tres etapes.

Etapa 1: Engegar el Servidor de Coneixement

Quan un participant entra a la comunitat d'aprenentatge virtual (CAV), és a dir, engega el seu **Servidor de Coneixement** el primer que fa és registrar-se al **Servidor de localització**. Aquest primer pas es fa mitjançant una petició ASP en la qual hi haurà la següent informació (com a mínim):

a. **USERID**: identificador del participant dins la CAV
b. **IP**: IP del Servidor de Coneixement

Per tant una crida *http* a un servidor d'ASP's vàlida tindria el següent format:

```
http://serverhost/login.asp?USERID=username&IP=ip
```

Taula 1.1 – Login al discovery server

Totes les crides ASP al servidor es farien d'una forma transparent a l'usuari i des del programa que desenvoluparem. Per fer-ho, utilitzarem la classe ***java.net.url*** J2SE. Podria ser de la forma següent:

```
// URL de la petició ASP amb el nom i la IP
urlName = "http://SERVERNAME/login.asp?USERID=" + name + "&IP=" + ip_address;

// creem l'objecte per la URL i enviem la petició
URL url = new URL(urlName);

// obrim la connexió de la URL
URLConnection connection = url.openConnection();

// Connexió amb la URL
connection.connect();

// A partir d'aquí podriem obrir un canal per llegir la resposta de la URL i
// processar aquesta
```

Taula 1.2 - Codi JAVA per enviar una petició al discovery server

Tots els missatges de resposta del servidor amb el resultat de l'operació o qualsevol altre informació estaran en format **XML**.

Etapa 2: Actualitzacions

Com hem dit, tots els Servidors de Coneixement que vagin entrant a la comunitat es van registrant al Servidor de Localització. Per aquest motiu, un Servidor de Coneixement cada cert temps haurà (de forma automàtica o manual) d'anar actualitzant la seva llista de Servidors de Coneixement ja que pot ser que se n'hagin incorporat de nous o que d'altres hagin sortit de la comunitat.

Per demanar una llista actualitzada, la crida *http* té el següent format:

`http://serverhost/userlist.asp`

Taula 1.3 - Petició de llista d'usuaris al discovery server

Etapa 3: Desconnectar del Servidor de Coneixement

L'últim pas correspon a l'acabament de la sessió per part d'un participant a la CAV. Per informar al Servidor de Localització que un *peer* abandona la comunitat, només cal fer la següent crida passant com a paràmetre l'identificador del *peer* corresponent:

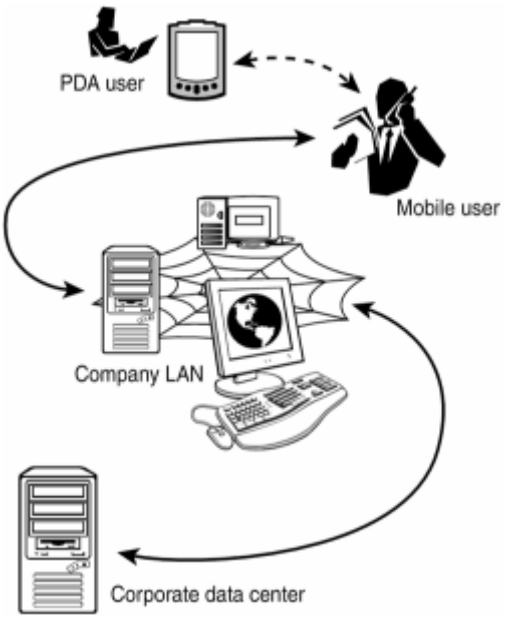
`http://serverhost/logout.asp?USERID=username`

Taula 1.4 - Logout del discovery server

Quan un *peer* de la comunitat té la llista d'IPs actualitzada de la resta de *peers* la interacció directa entre *peers* (no caldria ja el servidor) s'hauria d'implementar a partir de les primitives bàsiques que ofereix J2SE que són els *sockets*. Mitjançant *sockets* els diferents *peers* es podrien enviar/rebre missatges seguint uns protocols que hauriem de definir. Amb altres tecnologies com les que veurem a continuació ens podem estalviar el fet de començar de zero ja que proposen una plataforma estàndard pel desenvolupament d'aplicacions P2P, la qual cosa és molt interessant de cares a aquest projecte.

1.1.3 Projecte JXTA

JXTA [24] [25] [26] és un conjunt de protocols oberts i generals per peer-to-peer (P2P) que permeten que qualsevol dispositiu connectat a la xarxa –des d'un telèfon mòbil a una PDA, des d'un PC a un servidor- comunicar-se i col·laborar com a iguals *(Il·lustració 1.2)* directament a través de la xarxa. Els protocols JXTA són independents del llenguatge de programació utilitzat i podem trobar múltiples implementacions (anomenades *bindings* dins el projecte JXTA) per diferents entorns. Nosaltres, degut a que volem utilitzar per la realització del projecte eines *open-source* i gratuïtes, només ens fixarem amb el *binding* del **Projecte JXTA v2.0** per la plataforma Java 2 Standard Edition (J2SE).



Il·lustració 1.2 - Entorn juxtaposat al clàssic model client-servidor

1.1.3.1 Per què apareix JXTA?

A mesura que la Web continua creixent tant en contingut com en nombre de dispositius connectats, P2P està adquirint cada vegada més popularitat. El software basat en tecnologies P2P i conegut per tothom (Napster, Kazza, Emule, etc.) permet compartir fitxers, computació distribuïda i serveis de missatgeria instantani entre d'altres funcionalitats. Mentre cada una d'aquestes aplicacions realitza tasques diferents, totes elles comparteixen una sèrie de propietats comuns com ara trobar *peers*, buscar recursos i fitxers o transferir informació. Actualment, el desenvolupament d'aplicacions és ineficient ja que els programadors resolen els mateixos problemes i dupliquen implementacions d'infrastructures similars. A aquest fet, hi podem afegir que la majoria d'aplicacions són específiques d'una plataforma i no permeten comunicar-se i compartir informació amb altres aplicacions.

Un primer objectiu del Projecte JXTA és proporcionar una plataforma amb les funcionalitats bàsiques necessàries per una xarxa P2P. A part, també intenta solucionar alguns dels potencials defectes que podem trobar en molts dels sistemes P2P existents:

- **Interoperabilitat** – La tecnologia JXTA està dissenyada per permetre a *peers* que ofereixen serveis P2P el poder-se trobar i comunicar entre ells
- **Independència de la plataforma** – La tecnologia JXTA està dissenyada per ser independent dels llenguatges de programació, protocols de transport i plataformes de treball
- **Ubiqüitat** – La tecnologia JXTA està dissenyada per ser accessible per qualsevol dispositiu digital, no només PCs o una plataforma de treball específica

Una característica comuna dels *peers* en una xarxa P2P és que aquests solen existir al voltant d'una xarxa regular. El fet d'estar subjectes a una connectivitat no predictible i possiblement amb adreces de xarxa variables provoca que aquests *peers* quedin fora de l'àmbit estàndard del DNS. JXTA acomoda els *peers* al voltant de la xarxa proporcionant-los un sistema per adreçar-se de forma única i independent del tradicional servei de noms. Mitjançant l'ús de JXTA IDs, un *peer* es pot "passejar" a través de diferents xarxes canviant transports i adreces de xarxa, fins i tot estant temporalment desconnectat i continuar encara sent adreçable per altres *peers*.

1.1.3.2 Què és JXTA?

El projecte JXTA és, doncs, una plataforma oberta de computació a través de la xarxa dissenyada per treballar amb el paradigma *peer-to-peer*. El seu objectiu és desenvolupar els blocs i serveis bàsics per permetre la creació d'aplicacions innovadores per grups de *peers*.

El terme "JXTA" és una abreviació de *juxtapose*. És un reconeixement al fet que P2P és un paradigma juxtaposat al paradigma client-servidor o computació basada en Web que és actualment el model tradicional de computació distribuïda.

JXTA proporciona un conjunt de protocols oberts i una implementació de referència *open-source* pel desenvolupament d'aplicacions P2P. Els protocols JXTA estandarditzen la manera com els *peers:*

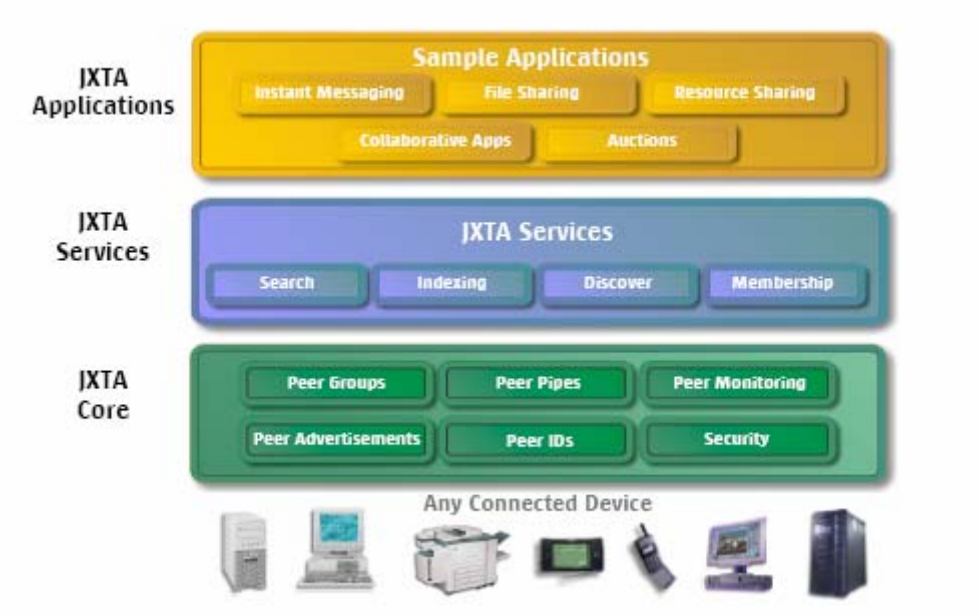
- es descobreixen entre ells
- s'auto-organitzen en grups
- anuncien i descobreixen serveis
- es comuniquen amb els altres
- monitoritzen els altres

Aquests protocols estan dissenyats per ser independents de llenguatges de programació i protocols de transport. Els protocols poden ser implementats en Java, C/C++, Perl i altres

llenguatges. També poden ser implementats sobre TCP/IP, HTTP, Bluetooth, HomePNA o altres protocols de transport.

1.1.3.3 Arquitectura

L'arquitectura del projecte JXTA està dividida en tres capes com podem veure en la figura següent:



Il·lustració 1.3 - Arquitectura projecte JXTA

- **Capa de Plataforma *(JXTA Core)***

  La capa de plataforma també coneguda com a *JXTA Core*, encapsula les mínimes i bàsiques primitives que són comuns en una xarxa P2P. Inclou blocs que permeten mecanismes clau per aplicacions P2P com ara descobriment, transport, creació de *peers* i grups de *peers* i primitives de seguretat.

- **Capa de Serveis *(Services Layer)***

  Aquesta capa inclou serveis de xarxa que no tenen que ser estrictament necessaris pel funcionament d'una xarxa P2P, però que són comuns o recomanables en entorns P2P. Exemples d'aquests serveis poden ser búsqueda i indexació, directori, sistemes d'emmagatzemament, compartició de fitxers, serveis de fitxers distribuïts, agregació de recursos, traducció de protocols, autentificació i PKI *(Public Key Infrastructure)*.

- **Capa d'aplicació *(Applications Layer)***

  La capa d'aplicació inclou la implementació d'aplicacions integrades com pot ser una aplicació P2P de missatges instantanis, compartició de documents i recursos, gestió i distribució de contingut d'entreteniment, sistemes P2P d'email i molts d'altres.

El límit entre servei i aplicació no és rígid. Una aplicació per un client pot ser vista com un servei per un altre. El sistema complert està dissenyat per ser modular, permetent als desenvolupadors escollir i utilitzar la col·lecció de serveis i aplicacions que satisfacin les seves necessitats.

1.1.3.4 Components

La xarxa JXTA consisteix en una sèrie de nodes interconectats coneguts com a ***peers***. Aquests s'auto-organitzen en ***peer-groups***, que proporcionen un conjunt comú de **serveis**. Exemples de serveis que pot proporcionar un *peer-group* inclou compartició de documents o aplicacions de *chat*.

Els *peers* anuncien els seus serveis amb uns documents XML anomenats ***advertisements***. Aquests permeten als altres *peers* aprendre com connectar-se i interactuar amb el servei ofert per un *peer*.

Els *peers* utilitzen ***pipes*** per enviar ***missatges*** entre ells. Les *pipes* son un mecanisme de transport de missatges asíncron i unidireccional utilitzat pel servei de comunicació. Aquests missatges son documents XML, el sobre del qual, conté informació sobre *routing, digest* i credencials. Les *pipes* estan lligades a un ***endpoint*** específic com pot ser un port TCP i associades a una adreça IP. Aquests conceptes els descriurem en el següent punt.

- **Aspectes clau**
  - L'ús de documents XML *(advertisements)* per descriure recursos en la xarxa
  - L'abstracció entre *pipes – peers* i *peers – endpoints* sense dependre de cap servidor central de noms/adreces com un DNS
  - Un esquema uniforme d'adreçament *(peer IDs)*

1.1.3.5 Conceptes JXTA

A continuació descriurem els components més importants de la plataforma JXTA.

- **Peers**

  Un *peer* és qualsevol dispositiu connectat a la xarxa que implementa un o més protocols JXTA. Un *peer* pot ser un sensor, un telèfon, una PDA, així com PCs,

servidors o supercomputadors. Cada *peer* funciona de manera independent i asíncrona respecte a tots els altres *peers* i està identificat de forma única per un ***Peer ID***.

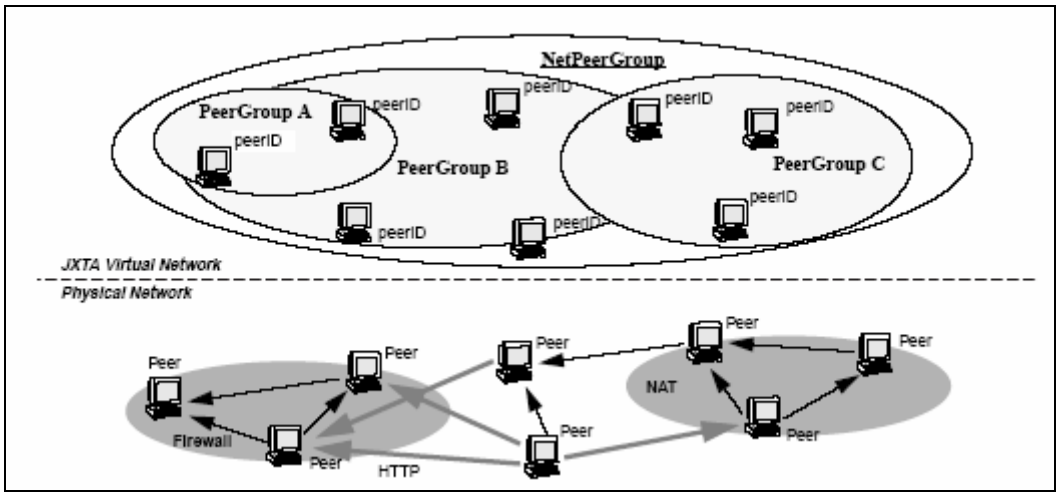
- **Peer Groups**

Un grup de *peers* és una col·lecció de *peers* que s'han posat d'acord alhora d'implementar uns serveis comuns. Els *peers* s'auto-organitzen en grups, cadascun identificat per un únic ***Peer Group ID***. Cada grup pot establir la seva pròpia política d'admissió de nous membres. Aquesta pot ser des oberta (tothom es pot afegir al grup) fins a estrictament protegida i segura (es demanen una sèrie de credencials per poder accedir al grup).

Els *peers* poden pertànyer a més d'un grup alhora. Per defecte, el primer grup instanciat és el *NetPeerGroup* per la qual cosa tots els *peers* pertanyen a aquest grup inicialment. Un cop en aquest grup, se'n poden crear de nous o afegir-se a un de ja existent.

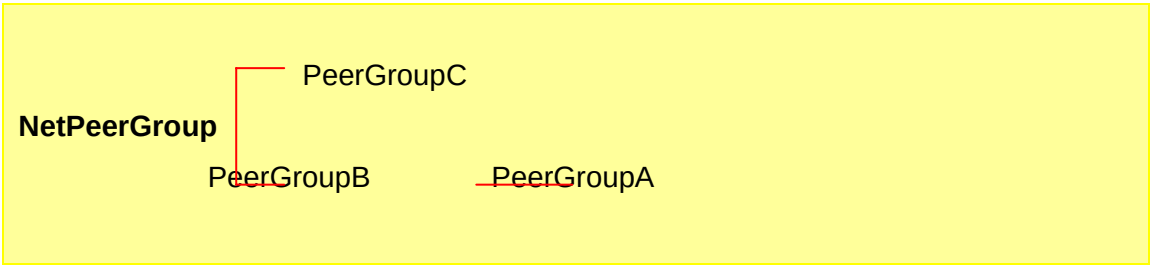
Els protocols JXTA descriuen com els *peers* poden publicar, descobrir, afegir-se i monitoritzar grups de *peers*; no indica quan o perquè s'han de crear grups.

Els grups formen una estructura jeràrquica amb relacions pare-fill, en les quals cada grup té un únic pare. Les cerques es propaguen dins del grup mentre que l'*advertisement* del grup es publica en el grup del pare a més del propi grup. En la següent il·lustració podem veure diferents grups creats sobre el grup per defecte *NetPeerGroup*.



Il·lustració 1.4 - Xarxa virtual JXTA amb diferents grups

L'estructura jeràrquica d'aquests grups seria la següent:



Il·lustració 1.5 - Estructura jerarquica dels peer groups

Pel que fa a la composició dels grups:

- PeerGroupB mostra un grup que agrupa varis dominis físics
- PeerGroupC mostra un grup que mapeja exactament els límits d'un domini NAT

Un *peer group* proporciona un conjunt de serveis anomenats *peer group services*. JXTA defineix un conjunt bàsic de serveis per un grup. Es poden desenvolupar altres serveis addicionals específics per les necessitats del grup. Perquè dos *peers* puguin interactuar a través d'un servei, aquests han de formar part del mateix *peer group*.

El nucli de serveis en un *peer group* inclou els següents:

- *Discovery Service* – Utilitzat pels *peers* membres per buscar recursos dins el grup com poden ser altres *peers*, *peer groups*, *pipes* i serveis.

- *Membership Service* – Utilitzat pels *peers* membres per rebutjar o acceptar una sol·licitud d'entrada al grup. Els *peers* que vulguin entrar en un grup el primer que han de fer és trobar-ne un membre i llavors fer la sol·licitud. Acceptar o rebutjar la sol·licitud depèn de tots els membres del grup. El servei forçarà una votació dels *peers* o l'elecció d'un representant del grup que decideixi.

- *Access Service* – Utilitzat per validar peticions enviades d'un *peer* a un altre. El *peer* que rep la petició proporciona al servei les credencials del *peer* sol·licitant i informació sobre la requesta feta per determinar si l'accés està permès.

- *Pipe Service* – Utilitzat per crear i gestionar connexions via *pipe* entre membres del grup.

- *Resolver Service* – Utilitzat per enviar preguntes genèriques a d'altres *peers*. Els *peers* poden definir i intercanviar preguntes per trobar qualsevol informació que creguin necessària.

- *Monitoring Service* – Utilitzat per permetre a un *peer* monitoritzar els altres *peers* del mateix grup.

Serveis de xarxa

Els *peers* cooperen i es comunicar per publicar, descobrir i invocar serveis de xarxa. Els *peers* poden publicar múltiples serveis. Tots els serveis són descoberts mitjançant el *Peer Discovery Protocol (PDP).*

JXTA reconeix dos nivells de serveis de xarxa:

1. *Peer Services*

   El servei ofert per un *peer* és accessible només en el *peer* que publica el servei. Si el *peer* cau, el servei també cau. Vàries instàncies del servei poden córrer en diferents *peers* però cada una publica el seu propi *advertisement*.

2. *Peer Group Services*

   Està composat per una col·lecció d'instàncies (que poden cooperar entre elles) d'un servei corrent en varis membres del *peer group*. Si algun dels *peers* cau, no afecta al servei col·lectiu del grup (suposant que el servei encara està disponible en algun dels *peers* del grup). Aquests serveis estan publicats com una part del *peer group advertisement.*

Moduls

Els mòduls en JXTA són una abstracció per representar qualsevol peça de "codi" utilitzada per implementar un comportament en el món JXTA. Els serveis de xarxa són l'exemple més corrent d'un comportament que pot ser instanciat en un *peer*. L'abstracció del mòdul no especifica que és aquest "codi"; pot ser un classe Java, un fitxer *.jar*, una llibraria dinàmica *.dll*, un conjunt de missatges XML o un script.

Els mòduls proveeixen una abstracció genèrica que permet a un *peer* instanciar un nou comportament. Quan els *peers* exploren o entren a un nou *peer group*, solen

buscar nous comportaments que llavors instanciaran. Per exemple, quan entres a un *peer group*, un *peer* ha d'aprendre un nou servei de búsqueda que només s'utilitza en aquell grup. Per tal d'afegir-se al grup, el *peer* ha d'instanciar aquest nou servei de búsqueda. El marc de treball del mòdul permet la representació i *advertisement* de comportaments independents de la plataforma i permet als *peers* descriure i instanciar qualsevol tipus d'implementació del comportament. Per exemple, el *peer* pot instanciar el comportament en una implementació Java o C.

Aquesta abstracció de mòdul inclou:

- *Module Class* – Utilitzat principalment per anunciar l'existència d'un comportament. Cada *module class* és identificat per un únic ID, el ModuleClassID.

- *Module Specification* – Utilitzat per accedir al mòdul. Conté tota la informació necessària per accedir o invocar al mòdul. Per exemple, en el cas d'un servei, aquesta especificació hauria de contenir un *pipe advertisement* que permetés la comunicació amb el servei.

  Hi poden haver vàries especificacions per un *module clas.* Cada especificació és identificada per un únic ID, el ModuleSpecID (aquest conté el ModuleClassID).

- *Module Implementation* – És la implementació d'un *module specification*. Poden haver-hi vàries implementacions per una especificació donada. Cada implementació conté el ModuleSpecID de l'especificació associada que implementa.

Els mòduls s'utilitzen pels *peer group services* i també poden se utilitzats per serveis independents. Els serveis JXTA poden utilitzar l'abstracció de mòduls que acabem de veure per identificar l'existència d'un servei *(Module Class)*, l'especificació del servei *(Module Specification)* o una implementació d'aquest *(Module Implementation)*. Cada un d'aquests components té un *advertisement* associat, el qual pot ser publicat i trobat per d'altres JXTA *peers.*

Pipes

Els *peers* JXTA utilitzen *pipes* per enviar-se missatges entre ells. Les *pipes* són un mecanisme de transferència de missatges asíncron i unidireccional utilitzades pel

servei de comunicació. Suporten la transferència de qualsevol objecte, incloent codi binari, cadenes de caràcters i objectes Java.

Els *pipe endpoints* es refereixen a l' *input pipe* (el punt final de recepció) i a l' *output pipe* (el punt final d'enviament). Aquests són dinàmicament lligats a *peer endpoints* durant l'execució. Els *peer endpoints* es refereixen a interfícies de xarxa disponibles en un *peer* (p. ex. un port TCP i la seva adreça IP associada) que es poden utilitzar per enviar i rebre missatges.

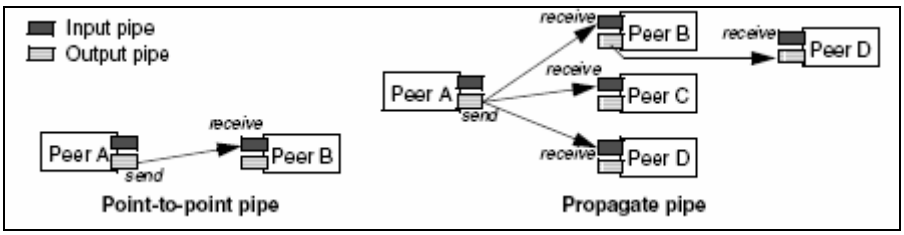
Podem trobar tres tipus de *pipes*:

1. *Point-to-point Pipes*
   Conecta exactament dos *pipe endpoints*: una *input pipe* en un *peer* rep missatges enviats des d'una *output pipe* d'un altre *peer*

2. *Propagate Pipes*
   Conecta una *output pipe* amb múltiples *input pipes*. Tota la propagació té lloc dins de l'àmbit d'un mateix *peer group*.

3. *Secure Unicast Pipes*
   És un tipus de *Point-to-point Pipe* que proporciona un canal segur de comunicació.

Les *pipes* són publicades i descobertes utilitzant *pipe advertisements* i són identificades de forma única mitjançant un *Pipe ID*.



Il·lustració 1.6 - Point-to-Point i Propagate Pipes

Missatges

Un missatge és un objecte enviat entre *peers* JXTA; és la unitat bàsica d'intercanvi d'informació. Els missatges són enviats i rebuts pel *Pipe Service* i pel *Endpoint*

Service. Normalment, les aplicacions utilitzen el *Pipe Service* per crear, enviar i rebre missatges.

Un missatge és una seqüència ordenada de parelles nom-valor anomenades *elements*, on podem trobar l'últim element afegit al final del missatge.

Els protocols JXTA són especificats com un conjunt de missatges intercanvits entre *peers*. Cada plataforma de software del *binding* descriu com el missatge es converteix a l'estructura de dades nativa de Java o C per exemple i viceversa.

S'accepten dos representacions pel *payload* del missatge: XML o binari.

- **Advertisements**

Tots els recursos que podem trobar en una xarxa JXTA – com *peers, peer groups, pipes* i serveis – son representats per *advertisements*. Els *advertisements* són unes estructures de metainformació en llenguatge neutral representades mitjançant documents XML. Els protocols JXTA utilitzen els *advertisements* per descriurei publicar l'existència dels recursos d'un *peer*. Els *peers* descobreixen recursos buscant els seus respectius *advertisements*.

Cada *advertisement* és publicat amb un temps de vida que especifica la disponibilitat associada al recurs. Això permet l'eliminació de recursos obsolets sense necessitat d'un control centralitzat. Un *advertisement* es pot tornar a publicar (abans que l'original expiri) amb la qual cosa s'allarga el temps de vida del recurs.

Els protocols JXTA defineixen els següents *advertisements*:

- *Peer Advertisement*
- *Peer Group Advertisement*
- *Pipe Advertisement*
- *Module Class Advertisement*
- *Module Spec Advertisement*
- *Module Impl Advertisement*
- *Rendezvous Advertisement*
- *Peer Info Advertisement*

Cada un d'aquests *advertisements* representa un recurs, en defineix unes propietats i té un identificador únic (ID).

Un exemple d'*advertisement* pot ser la figura que tenim a continuació:

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
<Id>
urn:jxta:uuid-
59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504
</Id>
<Type>
JxtaUnicast
</Type>
<Name>
TestPipe.end1
</Name>
</jxta:PipeAdvertisement>
```

Il·lustració 1.7 - Exemple d'un JXTA pipe advertisement

Els serveis o implementacions del *peer* poden modificar qualsevol dels *advertisements* anteriors per crear els seus propis.

- **Seguretat**

Les xarxes P2P dinàmiques com la que proposa JXTA necessiten donar suport a diferents nivells d'accés als recursos. Els *peers* JXTA operen en un model basat en rols de confiança, en el qual cada *peer* actua sota l'autoritat que li ha estat otorgada per un altre *peer* de confiança per realitzar una determinada tasca.

Cinc requeriments bàsics de seguretat han de ser-hi presents:

- Confidencialitat
- Autentificació
- Autorització
- Integritat de les dades
- No repudiació

Els missatges XML ofereixen la possibilitat d'afegir metainformació com credencials, certificats, *digests* i claus públiques als missatges JXTA, permetent així l'assoliment d aquestes mesures bàsiques de seguretat. El resum *(digest)* de missatges garanteix la integritat del missatge. Alhora poden ser encriptats (utilitzant claus públiques) i signats (utilitzant certificats) per confidencialitat i no repudiació. Les credencials poden ser utilitzades per assegurar l'autorització i autentificació.

Una **credencial** és un *token* utilitzat per identificar el remitent que s'ha de presentar cada vegada que s'envia un missatge.

IDs

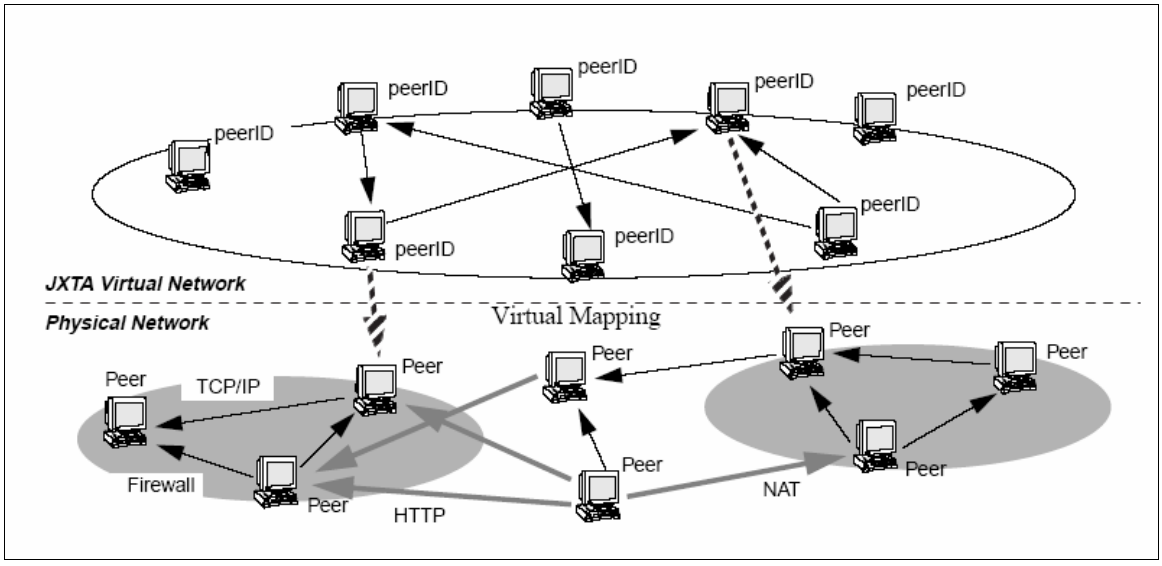
Tots els recursos que poden trobar en una xarxa JXTA necessiten ser identificats de forma única. Un JXTA ID identifica de forma única una entitat i serveix com a mitjà canònic per identificar-la. Actualment hi ha sis tipus d'entitats JXTA que tenen JXTA ID definits: *peers, peer groups, pipes, contents, module classes* i *module specifications*.

Els JXTA IDs es representen mitjançant URNs. Els URNs són una forma d'URIs que "... volen servir com un identificador de recursos persistent i independent de la localització". Com les altres formes d'URIs, els JXTA ID són representats com a text. Un exemple de JXTA *peer* ID és:

```
urn:jxta:uuid-
5961626164616261 4A7874615032503 3F3BC76FF13C2414CBC0AB663666DA53903
```

Els IDs únics són generats de forma aleatòria pel *binding* de JXTA de la plataforma J2SE. Hi ha dos IDs especials i reservats: el NULL ID i el *Net Peer Group ID*.

Així doncs, mitjançant els JXTA IDs que com hem dit permeten que un *peer* sigui adreçat de forma independent a la seva adreça física, tenim una xarxa virtual JXTA que està per sobre de la infrastructura física de xarxa existent *(Il·lustració 1.8)*. Aquesta xarxa virtual permet als *peers* intercanviar missatges entre ells de forma independent a la seva localització dins la xarxa. Per exemple, un ordinador portàtil inicialitzat amb DHCP, la qual cosa pot provocar que tingui diferents IPs, sempre tindrà el mateix *peer ID.*



Il·lustració 1.8 - Xarxa virtual del Projecte JXTA

1.1.3.6 Arquitectura de Xarxa

Per fer-nos una idea de com JXTA estructura la xarxa farem una breu explicació de tres punts bàsics pel seu funcionament:

1. Organització de la xarxa
2. SRDI *(Shared Resource Distributed Index)*
3. Firewall i NAT

Organització de la xarxa

La xarxa JXTA és alhora una xarxa de múltiples salts *(multi-hop)* i adaptable formada per *peers* connectats. Les connexions no son persistents i l'enviament de missatges entre *peers* no és deterministic. També cal tenir en compte que els *peer* poden afegir-se i sortir de la xarxa en qualsevol moment i que les rutes canvien de forma freqüent.

L'organització de la xarxa no queda fixada pel marc de treball de JXTA si bé podem trobar-hi quatre tipus de *peers*:

- ***Minimal Edge Peer***

  Aquest tipus de *peer* només pot enviar i rebre missatges però no guarda *advertisements* o enruta missatges destinats a d'altres *peers*. *Peers* que siguin dispositius amb recursos limitats (p. ex. PDA o telèfon mòbil) segurament seran d'aquest tipus.

- ***Full-featured edge peer***

  A diferència de l'anterior, aquest a més d'enviar i rebre missatges, també guarda *advertisements*. Un simple *peer* respon a peticions de descobriment de recursos amb informació que pugui tenir en els *advertisments* que té guardats, però no reenviarà cap petició. La majoria dels *peers* seran d'aquest tipus.

- ***Rendezvous Peer***

  Aquest *peer* a part de guardar *advertisements* també s'encarrega de reenviar peticions de descobriment per ajudar a altres *peers* a trobar recursos. Quan un *peer* s'uneix a un grup el primer que fa és buscar un *rendezvous peer*. Si no en troba cap ell mateix esdevé un *rendezvous peer* per aquell grup. Cada *rendezvous*

peer manté una llista a d'altres *rendezvous peers* coneguts així com una altra amb tots els *peers* que l'estan utilitzant.

Els *Edge Peers* envien peticions de búsqueda i descobriment als *rendezvous peers*, els quals reenvien aquestes si no les poden respondre. Aquest procés continua fins que un *peer* té la resposta o la petició expira. Els missatges tenen un temps de vida (TTL) establert per defecte a set salts *(hops)*. Els *loopbacks* s'eviten mantenint una llista de *peers* visitats en el missatge.

- ***Relay Peer***

  Un *relay peer* manté informació sobre rutes a altres *peer* i enruta missatges cap a d'altres *peers*. També s'encarreguen de reenviar missatges de part de *peers* que no poden adreçar directament altres *peers* (p. ex. perquè estan utilitzant NAT).

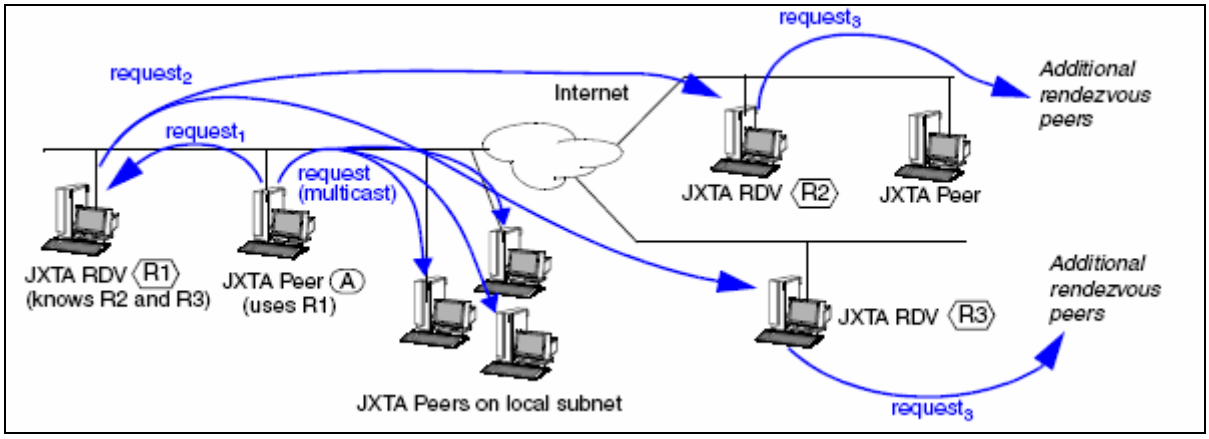
SRDI

La plataforma JXTA 2.0 J2SE suporta un servei d'índex de recursos compartits distribuit (SRDI) per proveir un mecanisme de propagació de peticions dins la xarxa JXTA més eficient. Els *rendezvous peers* que hem vista abans, mantenen un índex dels *advertisements* que han publicat els *edge peers*. Quan un d'aquests publica nous *advertisements* utilitza el servei SRDI per posar els índexs al seu *rendezvous*. Amb aquesta jerarquia *edge-rendezvous peer*, les peticions només es propaguen entre *rendezvous peers* la qual cosa redueix el nombre de *peers* involucrats en la búsqueda d'un *advertisement*.

Alhora, cada *rendezvous* manté la seva pròpia llista de *rendezvous* coneguts dins un grup. Aquests *rendezvous*, s'intercanvien periodicament informació entre ells sobre quins *rendezvous* conneixen per actualitzar la llista. També eliminen de la llista els *rendezvous* que ja no responen.

Quan un *peer* publica un *advertisement* aquest és indexat pel servei SRDI utilitzant claus com el nom o l'ID de l'*advertisement*. Només els índex de l'*advertisement* es copien al *rendezvous* minimitzant així la quantitat d'informació que s'hi ha d'emmagatzemar. Alhora, el *rendezvous* que rep el nou índex el reenvia a d'altres *rendezvous* seleccionats segons una funció de *hash* respecte a l'índex.

Un exemple de com es propaguen les **peticions** i del que acabem d'explicar el podem trobar en la següent figura:

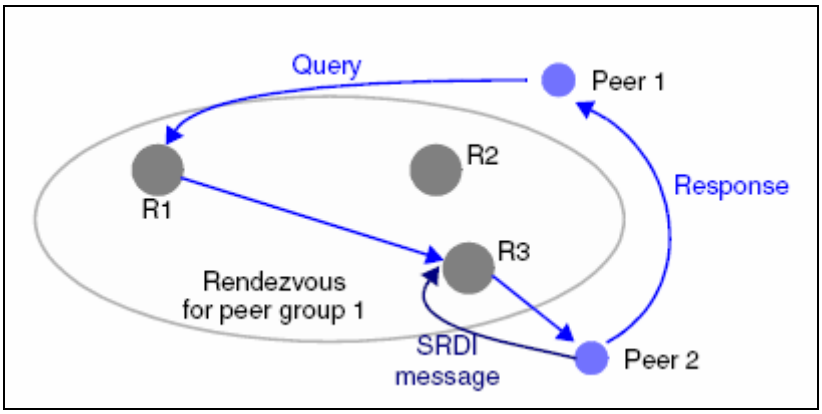


Il·lustració 1.9 - Propagació d'una petició via rendezvous peers

El *peerA (edge peer)* està configurat per utilitzar R1 com el seu *rendezvous*. Quan *peerA* comença una petició de descobriment o de búsqueda aquesta va inicialment al seu *rendezvous peer* – R1 en aquest exemple – i via multicast als altres *peers* de la seva subxarxa.

Els *peers* dins la mateixa subxarxa responen directament si tenen la informació demanada a la seva *caché* local.
Les peticions més enllà de la xarxa local són enviades als *rendezvous peers* connectats. Aquests intenten respondre buscant la petició a la seva *caché* local. Si tenen la informació demanada l'envien al *peer* i no propaguen més la petició. D'altra banda, si tenen l'índex del recurs demanat al seu SRDI el que fan és notificar-ho al *peer* que havia publicat l'*advertisement* i aquest respondrà directament al *peer* sol·licitant.
Si el *rendezvous peer* no té la resposta, s'utilitza un algoritme de límit de salts o *hops* per evitar que la búsqueda es faci infinita.

La Il·lustració 1.10 mostra una vista lògica de com funciona el servei SRDI. El *Peer2* publica un nou servei i envia un missatge SRDI cap al seu *rendezvous* (R3). Els índexs es guarden a R3 i es reenvien als altres *rendezvous* del grup. Llavors, el *Peer1* envia una petició per aquest recurs al seu *rendezvous* (R1). R1 mira a la seva *caché* local d'entrades SRDI i propaga la petició si no troba l'índex. Finalment, quan es troba el recurs al *Peer2*, aquest respon directament al *Peer1* amb l'*advertisement* demanat.



Il·lustració 1.10 - Funcionament servei SRDI

1.1.3.7 Protocols JXTA

JXTA defineix una sèrie de protocols o missatges XML per la comunicació entre els *peers*. Aquests utilitzen els protocols per descobrir-se, anunciar i trobar recursos, comunicar-se i enrutar missatges.

Tots els protocols són asíncrons i estan basats en el model pregunta/resposta. Un *peer* JXTA utilitza un protocol per enviar una petició a un o més *peers* en el seu grup i pot rebre zero, una o més respostes. Els *peers* no estan obligats a implementar els sis protocol; només els que utilitzin.

El resultat de la implementació dels protocols JXTA són els serveis. Els serveis també inclouen alters components *software* que suporten altres activitats diferents als protocols bàsics de JXTA i que poden ser utilitzats per d'altres *peers*. Aquests es troben en un nivell per sobre dels serveis implementats en la base de JXTA.
Els sis protocols són els següents:

- **Peer Discovery Protocol (PDP)**

  Aquest protocol és utilitzat per descobrir qualsevol recurs publicat per un *peer*. Com ja hem dit abans, els recursos estan representats com *advertisements* i poden ser un *peer*, un *peer group*, una *pipe*, un servei o qualsevol altre recurs que tingui un *advertisement*.

  PDP permet trobar *advertisements* en d'altres *peers* i és el protocol de descobriment per defecte per tots els grups definits per l'usuari així com pel *NetPeerGroup*. Es

poden definir serveis propis de descobrimet per alliberar el PDP; en qualsevol altre cas PDP serà el servei que consultarà a d'altres *peers* buscant *advertisements*.

Hi ha diferents maneres per descobrir informació distribuida. Actualment, el binding del projecte JXTA per la plataforma J2SE utilitza una combinació de IP *multicast* per la xarxa local i *rendezvous peers* quan sortim a Internet. Hem vist el seu funcionament a l'apartat Arquitectura de Xarxa. Altres tècniques com CANs *(Content-Adressable networks)* es poden afegir per realitzar la búsqueda de recursos.

Els *peers* generen un missatge de sol·licitud de pregunta per descobrir *advertisements* dins d'un grup. Aquest missatge conté la credencial del grup on està el *peer* i l'identifica a qui rep el missatge. El missatge pot ser enviat a tots els *peers* d'una subxarxa o a un *rendezvous peer*.

Com a resultat d'una petició es poden rebre zero, una o més respostes. El missatge de resposta pot contenir un o més *advertisements*.

- **Peer Information Protocol (PIP)**

  Un cop hem trobat un *peer* dins la xarxa JXTA li podem preguntar les seves capacitats i el seu estat. PIP proporciona una sèrie de missatges per obtenir informació del *peer*.

  El missatge *ping* que forma part del protocol PIP s'envia a un *peer* per comprovar si el *peer* està "viu" i per obtenir-ne informació. En el missatge es pot especificar si es vol una resposta complerta (un *peer advertisement*) o només un simple reconeixement (estat i temps d'activitat).

- **Peer Resolver Protocol (PRP)**

  PRP permet als *peers* enviar preguntes genèriques als altres *peers* i identificar les respostes que compleixen el que s'havia demanat. Poden ser enviades a un *peer* específic o propagades mitjançant serveis de *rendezvous* dins l'àmbit d'un grup. PRP utilitza el servei de *rendezvous* per difondre una petició a múltiples *peers* i missatges *unicast* per enviar-les a un únic *peer.*

  Tant PIP com PDP estan construits sobre PRP proporcionant funcions específiques com acabem de veure. Podem veure la jerarquia de protocols a la Il·lustració 1.11.

Podem utilitzar PRP per qualsevol pregunta genèrica que pugui necessitar una aplicació.

- **Pipe Binding Protocol (PBP)**

  Els *peers* membres d'un grup utilitzen aquest protocol per llibar l'*advertisement* d'una *pipe* amb el seu corresponent *pipe endpoint*. El *link* virtual a una *pipe (pathway)* pot ser mogut per diferents capes de transport de xarxa com TCP/IP. Cada *end pipe* s'encarrega de mantenir el *link* virtual i refer-lo si és necessari.

  Podem veure una *pipe* com una cua de missatges amb nom que es pot crear, obrir, tancar, esborrar i que pot enviar i rebre missatges. Les implementacions actuals d'una *pipe* poden variar però totes utilitzen PBP per lligar la *pipe* a un *endpoint*. Durant l'operació abstracte de creació, un *peer* local lliga un *pipe endpoint* a un determinat transport.

- **Endpoint Routing Protocol (ERP)**

  Aquest protocol defineix una sèrie de missatges tipus pregunta/resposta per trobar informació sobre enrutament. Aquesta informació és necessaria per enviar un missatge des d'un *peer* (origen) fins a un altre (destí). Quan un *peer* vol enviar un missatge a un altre el primer que fa és mirar a la seva *caché* local per si hi té la ruta fins a aquest *peer*. Si no la troba envia una petició de resolució de ruta a un dels seus *peer relays* disponible. Quan aquest últim rep la petició comprova si coneix la ruta. Si és així envia la informació com una enumeració de salts.

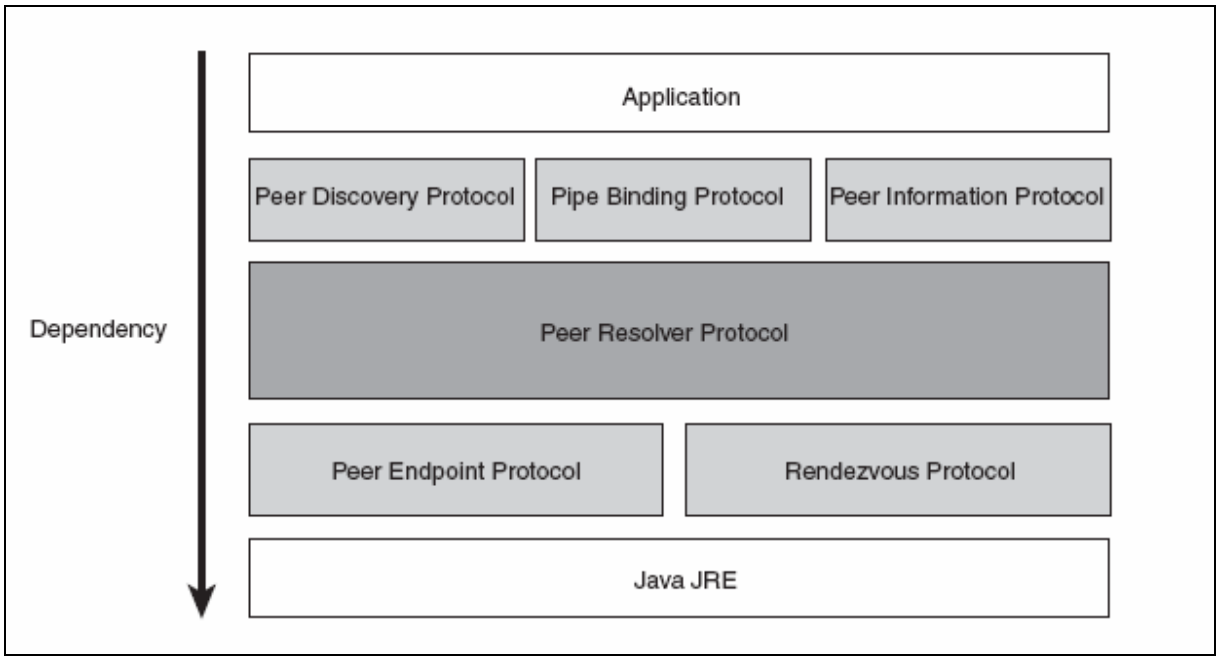
  Qualsevol *peer* pot preguntar a un *relay peer* informació sobre una ruta i qualsevol *peer* dins un grup pot convertir-se en un *relay peer*. Bàsicament el que fan és emmagatzemar informació de rutes.

- **Rendezvous Protocol (RVP)**

  És el protocol responsable de difondre missatges dins el grup de *peers*. Permet:

  - Conectar els *peers* a un servei (ser capaços de propagar missatges i rebre'ls)
  - Controlar la propagació de missatges (TTL, detecció de *loopback*, etc.)

  Els protocols PRP i PBP utilitzen aquest protocol per difondre missatges com podem veure a continuació:



Il·lustració 1.11 - Jerarquia dels protocols JXTA

1.1.4 Conclusions

Un cop hem estudiat, experimentat i avaluat les tres tecnologies anteriors, podem dir que hem après el funcionament de cadascuna i com s'adapten a les nostres necessitats. Hem vist, entre d'altres coses, com resoldre el descobriment de *peers*, com enviar missatges, com crear amb certa facilitat aplicacions distribuïdes, etc. Amb tot, podem treure'n algunes conclusions que ens ajudaran a decidir com implementem el nostre Servidor de Coneixement:

- **La plataforma ANTS** és una infrastructura que parteix d'unes idees molt semblants a les nostres. Temes com *Computer Supported Cooperative Work (CSCW)* i més concretament *Computer Supported Cooperative Learning (CSCL)* encaixen perfectament amb el nostre projecte. Per tant, en un principi el fet d'utilitzar ANTS com a plataforma sobre la qual desenvolupar-lo semblava molt interessant. A més, ens ofereix una sèrie de facilitats alhora de crear aplicacions a la seva capa d'aplicació. Recordem que el seu objectiu és fer el més transparent possible el pas d'una aplicació local a una de distribuïda.

  El millor de la plataforma és la base i les facilitats que ens proporciona per crear aplicacions distribuïdes. Especialment, aplicacions encarades a l'ensenyament ja que proporciona per exemple un mòdul de consciència que ens permet de forma molt senzilla

monitoritzar tot el que passa al sistema; principalment totes les accions que fa l'usuari o alumne.

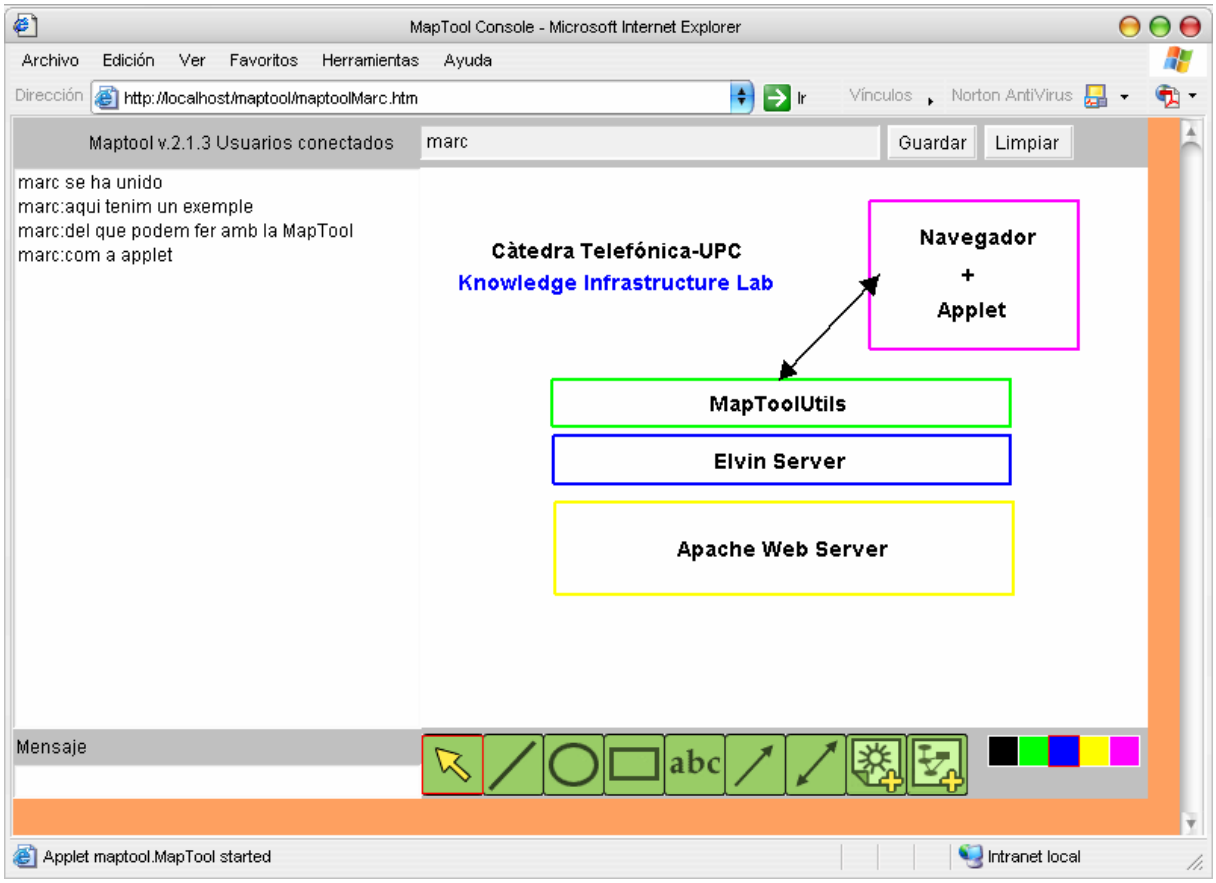
No obstant, té alguns inconvenients importants que ens desaconsellen el seu ús i que fan que no s'adapti perfectament al que nosaltres necessitem. Primer, no ens resol ni ens dóna cap ajuda per un tema vital pel nostre projecte com és el fet de trobar tots els *peers* connectats a la nostra Comunitat d'Aprenentatge Virtual. Les diferents màquines que es comuniquen mitjançant ANTS ho fan directament a partir de la IP de cadascuna; per tant, són IPs conegudes.

Per altra banda, el nostre objectiu tampoc és crear una aplicació educativa ja que el que volem fer es decanta més cap a la idea de complementar diferents mòduls o sistemes (com pot ser un LMS, una pissarra digital, un repositori, etc.) i adaptar la plataforma resultant a una xarxa P2P.

Cal tenir en compte també que les aplicacions més conegudes que s'han realitzat utilitzant ANTS com poden ser MOVE o BSCL estan basades en el model client-servidor; és a dir col·laboració via Web principalment amb un servidor central.

Per últim, el fet d'haver d'instal·lar tota la plataforma ANTS a cada *peer* tampoc és molt recomanable i podria generar conflictes alhora d'autentificacions d'usuaris.

Per tant, vistos els pros i contres, hem decidit no utilitzar la plataforma ANTS. No obstant, el que si que possiblement utilitzarem com un *pluggin* o mòdul al nostre sistema serà la Maptool també desenvolupada pels creadors d'ANTS i de la qual se n'ha fet una versió com a *applet* que es pot integrar fàcilment a qualsevol aplicació. No utilitza ANTS completament ni Orion. Només necessita el servidor de subscripció/publicació de missatges Elvin. Una captura de la MapTool com a *applet* és la següent:



Il·lustració 1.12 - Maptool com a Applet

En resum, tot i no utilitzar ANTS al complert, el seu estudi i experimentació ens ha permès conèixer tecnologies sòlides i actuals com per exemple diferents servidors de publicació/subscripció de missatges com pot ser Elvin i JMS. També, hem vist com aprofiten els serveis que ofereix un servidor d'aplicacions (en aquest cas Orion) en temes com seguretat, transaccions, escalabilitat, etc. Alhora, hem conegut idees i teories sobre com adaptar l'ensenyament a les noves tecnologies i opcions que ens ofereix la xarxa que concorden força amb les idees de la Càtedra Telefónica-UPC.

- **El projecte JXTA** es va convertir des del començament en la nostra opció preferent degut a les idees i la visió sobre P2P que són la base d'aquest projecte i una part molt important del nostre. Tots els conceptes introduïts en l'apartat Projecte JXTA semblaven adaptar-se perfectament al que necessitàvem i era la tecnologia destinada a convertir-se en la nostra plataforma (API) per la creació del nostre Servidor de Coneixement.

  Seguint les opcions que ens ofereix JXTA, la idea era crear un *peergroup* propi corresponent a la Comunitat d'Aprenentatge Virtual i que tots els participants d'aquesta entressin a aquest grup. A més, seria un grup segur amb la qual cosa podríem evitar

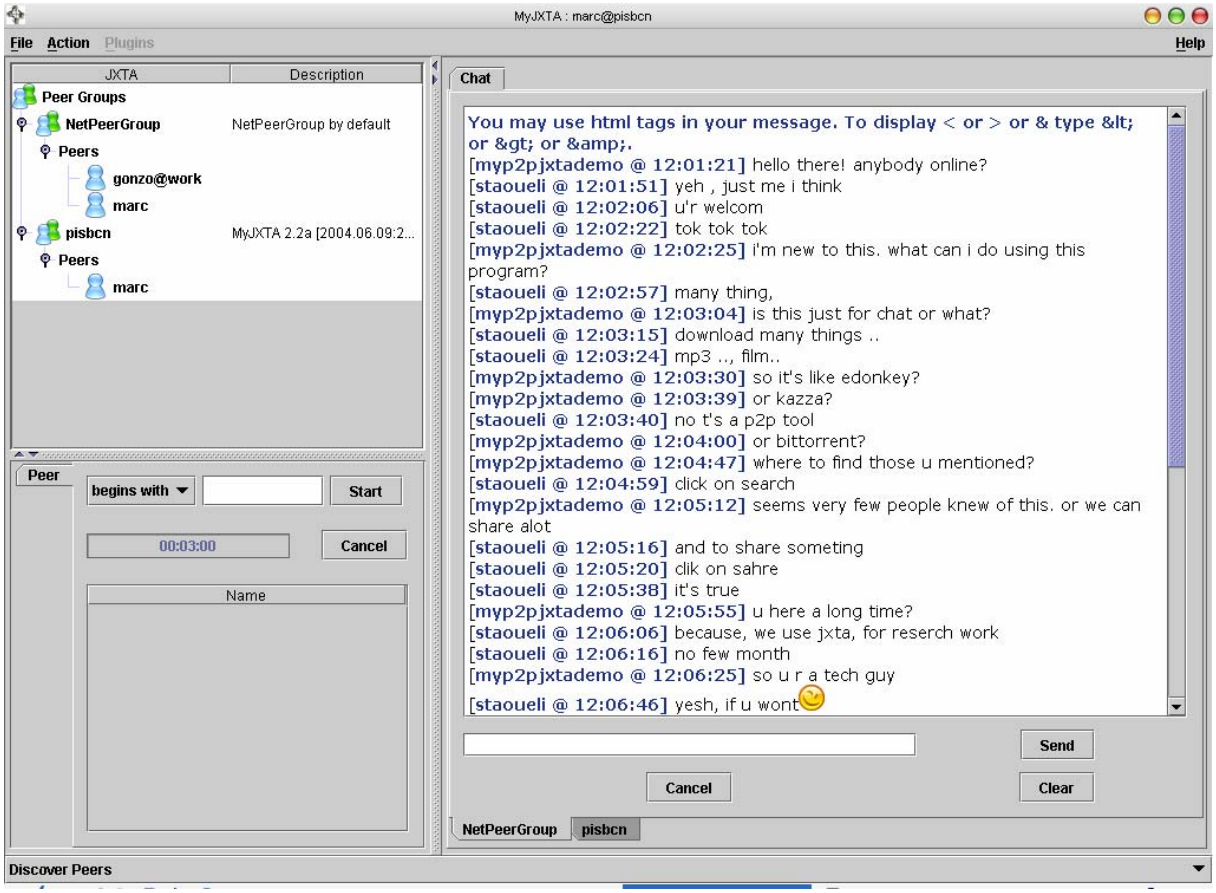
l'entrada de persones no desitjades. Dins del grup, crearíem diferents serveis que permetessin una comunicació segura i fiable entre els *peers*. Aquests es basarien en la creació de *pipes* ja siguin individuals, segures o de propagació dins del grup i una sèrie de protocols coneguts per tots els *peers*. Per tant, de manera senzilla i utilitzant JXTA trobaríem els *peers* i establiríem una serie de serveis.

Per tal, de comprovar el funcionament de JXTA i la viabilitat del sistema que havíem pensat, s'han realitzat petits programes destinats a:

- trobar *peers*
- crear / trobar / entrar a un *peergroup*
- anunciar / trobar *advertisements*
- crear / trobar / utilitzar serveis
- comunicació amb *pipes* (normals, bidireccionals, etc.)

A part, d'aquests petits programes, també hem instal·lat i provat programes realitzats per membres de la comunitat JXTA (o pels mateixos creadors del projecte) com per exemple:

- ***MyJXTA***: aplicació que permet *chat* i intercanvi de fitxers entre *peers*
- ***MyJXTA2***: nova versió de *MyJXTA* destinada a substituir l'anterior que aprofita les noves funcionalitats de la nova API de JXTA 2.3. S'ha canviat també la interfície i ara com ara permet *chat*, creació de grups, entrar / sortir de grups, una llista de presència de membres a cada grup i intercanvi de fitxers. Aquesta és una de les aplicacions JXTA que més hem provat ja que ha estat desenvolupada per programadors de JXTA i utilitzant les últimes versions de la plataforma. El seu aspecte és el següent:



Il·lustració 1.13 - MyJXTA 2

- ***Mindshare***: aplicació P2P per grups petits que treballen conjuntament. Permet compartir fitxers com fa molta gent en una LAN però sense la necessitat d'un servidor centralitzat. Gareth Farrington està realitzant aquesta aplicació com a tesis i espera que en el futur les aplicacions amb JXTA siguin tant populars com Gnutella.
- ***Shell***: és una consola que permet fer consultes sobre l'estat de la xarxa JXTA; *peers* connectats, grups, informació sobre *advertisements*, sobre *rendezvous* i *relays servers*, etc. Totes aquesta informació es pot obtenir amb simples comandes a l'estil del *shell* de Unix.
- ***JXTAView****:* aplicació que busca *peers* i *peergroups* dins la xarxa JXTA i els mostra en una interfície gràfica en forma d'arbre.

Provant els programes anteriors hem trobat alguns problemes amb JXTA que ens han fet replantejar la seva utilització per aquest projecte. Per començar, vam provar diferents programes dels anteriors en una LAN. En una xarxa local, els programes funcionàvem perfectament, es trobaven els *peers,* els serveis, els *advertisements*, etc. Això no tenia

gaire dificultat ja que amb la opció de *multicast* activada que permet escollir la pantalla de configuració de la plataforma JXTA, es trobaven tots els recursos sense problemes. Així doncs, dins una xarxa local les proves van ser satisfactòries i JXTA ens oferia moltes facilitats per comunicar els diferents PC's connectats. Totes aquestes immillorables perspectives van canviar força però quan les proves van ser a la xarxa global o Internet.

Explicarem doncs, els principals problemes trobats utilitzant JXTA a Internet:

1. Totes les aplicacions que han de trobar *peers* dins d'un grup *(MyJXTA, MyJXTA2, Mindshare, JXTAView, etc.)* poden tardar molt de temps a trobar-los o fins i tot poden no trobar-los. No tenim cap seguretat de que els *peers* que mostra a la llista siguin tots els existents ni tampoc de que tots els que hi ha a la llista estiguin actualment *on-line*.

2. Provant una petita aplicació que publica un servei mitjançant *advertisements* i una altra que el busca i l'utilitza no ha estat possible que la segona trobés el servei desitjat quan aquest existia a la xarxa JXTA.

3. Utilitzant el *MyJXTA2* s'ha comprovat que amb:

i. dos *peers* en dos PCs diferents
ii. amb diferent IP pública
iii. tots dos utilitzant aquest programa
iv. estan al mateix grup *(NetPeerGroup)*

en un principi hi havia missatges que arribaven a un i no a l'altra. Passats uns minuts, els missatges van convergir i tots dos *peers* rebien els mateixos.

I és que com ja hem comentat en l'apartat Projecte JXTA la majoria de cerques de recursos JXTA són **asíncrones** i ens poden tornar **zero, un o més resultats** amb la qual cosa no tenim mai la seguretat que la consulta s'ha realitzat correctament i ens ha retornat tots els recursos buscats que hi ha a la xarxa. D'altra banda, tampoc sabem mai el temps que podem necessitar per trobar-los i aquest pot tardar des de segons a varis minuts depenent del nombre de salts o *hops* que hagi de fer per la xarxa JXTA. No obstant, la documentació diu que al final, tots els recursos acaben convergint i trobant-se. Un altre problema, és el de la fiabilitat. Alguns dels protocols o serveis com per exemple les *propagated pipes* no inclouen fiabilitat per la qual cosa no ens poden assegurar que els missatges enviats arriben correctament a tots els destinataris. Per

solucionar això s'hauria d'implementar un servei/protocol a sobre que controlés tot l'enviament i recepció de missatges.

Per tant, el gran problema que tenim amb JXTA és la fiabilitat, la confiança que ens pot oferir la seva utilització. Si bé aquesta no és complerta, podem dir que les aplicacions funcionen. És a dir: es pot conversar amb *MyJXTA2* amb d'altres persones connectades a Internet, es poden intercanviar fitxers, etc. Però no podem predir el temps que tardaran dos persones a trobar-se, trobar tots els serveis i comunicar-se sense problemes gràcies a la xarxa JXTA i sense cap servidor central.

També és important destacar la importància que hem vist que tenen els *rendezvous peers* dins la xarxa. El fet d'estar connectat a un de fiable, potent i amb un ample de banda suficient pot facilitar molt la comunicació. Per exemple, el fet que dos *peers* estiguin connectats al mateix *rendezvous*[1] farà que es trobin amb molta més rapidesa que si estan en *rendezvous* diferents. Podríem dir que aquests fan com uns pseudo-servidors i que si varis *edge peers* estan connectats al mateix tota la cerca de recursos i comunicació és més fàcil.

Per últim comentarem algunes consideracions respecte a l'estat actual de la plataforma. Els desenvolupadors de la plataforma actualitzen les versions amb certa freqüència millorant-ne l'eficiència, la seguretat, resolent *bugs*, afegint-hi serveis, etc. L'última versió (durant aquest estudi) per J2SE (la plataforma JAVA és en la que es concentren tots els esforços) és **JXTA J2SE 2.3 "Jambalaya"**. Està prevista una nova versió anomenada **"Yaprakh"** pel setembre del 2004. Pel que fa a la documentació, hi ha varis llibres que parlen exclusivament de JXTA però tots son anteriors a l'any 2003 i com acabem de dir amb les constants actualitzacions de la plataforma, seria convenient algun nou llibre actualitzat. Per tant, per obtenir informació, documentació i una especificació actualitzada, tot s'ha de consultar via web (www.jxta.org) o llistes de correu d'usuaris i desenvolupadors.

Com a conclusió a tot el que acabem de veure de JXTA podríem dir que la curva d'aprenentatge per crear aplicacions P2P amb JXTA no és ni molt menys tant senzilla com podria semblar de bon principi si es vol realitzar una aplicació segura, fiable i escalable. Tot i els inconvenients, JXTA està en constant millora i pot tenir molta

[1] Recordem que un *edge peer* sempre que entra a la xarxa JXTA es connecta a un *rendezvous peer*

importància en el futur sobretot si **Sun**[2] hi posa interès i esforços en el seu desenvolupament. No obstant, la companyia està força aturada en aquest projecte si ho comparem amb els esforços realitzats en d'altres camps. Aquesta postura és força descoratjadora si tenim en compte l'interès generat per Sun i Intel sobre la tecnologia P2P amb l'èxit de Napster i quan semblava que P2P era el pròxim gran *boom*. El fet que importants empreses com Verizon o Nokia entre d'altres utilitzin JXTA i que el director de tecnologies avançades de Sun, Juan Carlos Soto, asseguri que cada cop es veuran més i més productes de Sun que inclouran la tecnologia JXTA pot accelerar-ne l'interès i la seva implantació definitiva. A part d'aquestes importants empreses es pot destacar que el projecte JXTA s'ha baixat de la seva Web més de 2 milions de vegades i que més de 16.000 persones s'han registrat com a membre de **JXTA.org**.

Vista doncs la situació actual de JXTA no podem basar la nostra aplicació –un dels punts vitals de la qual és trobar els membres de la comunitat- en una plataforma que no ens pot assegurar aquest descobriment de *peers.* Ens queda doncs la última opció que tractarem a continuació.

- La opció d'incloure un **Servidor de Descobriment de peers ("discovery server")** a la infrastructura és la opció segurament menys innovadora però la única –ara mateix- que ens ofereix la seguretat de trobar tots els *peers* connectats en un temps acceptable. Com a punt negatiu tenim el fet de necessitar un servidor central amb tot el que això implica respecte a:
  - punt singular de fallada
  - necessitat d'estar *on-line* 7x24
  - escalabilitat
  - el cost augmenta com més *peers* es connecten
  - IP fixe
  - haurem de definir un "protocol" per la comunicació directe entre *peers*. Aquesta haurà de ser directament via *sockets* un cop es coneguin les IPs. Aquest "problema" amb JXTA ja el teniem resolt.

[2] El Projecte JXTA™ va començar el 2001 com a projecte de recerca a Sun Microsystems sota la supervisió de Bill Joy i Mike Clary per ordenar l'espai P2P.

Ara doncs, hem de decidir quina tecnologia utilitzem per implementar aquest servidor de descobriment. Tenim diferents opcions a escollir depenent de les característiques o de la disponibilitat de gestió sobre el PC que faria de servidor.

La implementació que hem provat en l'apartat Simple Discovery Server per avaluar la tecnologia consistia en un servidor amb un servidor web (IIS), una base de dades Accés i un mòdul ASP. No obstant, la implementació definitiva no té perquè tenir aquesta configuració. Per exemple, una altra opció seria un servidor web Apache, una base de dades MySQL i un mòdul de PHP. En aquest supòsit treballaríem només amb *software* "open source" que és un dels requeriments que ens hem imposat. Tractarem la configuració definitiva més endavant.

2. Infrastructura per l'aprenentatge en xarxa

Aquesta nova infrastructura sorgeix perquè des del punt de vista del nostre laboratori –el "Knowledge Infrastructure Lab"- els sistemes que trobem avui en dia no ens permeten assolir els objectius marcats dins del laboratori. Dit d'una altra forma, no podem implementar els conceptes d'Espai d'Aprenentatge en una xarxa P2P amb les tecnologies que trobem avui en dia. Per tant, per implementar els conceptes introduïts en el Working Report WR-2004-01 proposem les infrastructures tecnològiques següents.

2.1 Servidor de Coneixement

Nosaltres veiem aquest servidor com *un PC domèstic amb software específic de client-servidor que permet interactuar amb d'altres servidors de coneixement mitjançant el paradigma P2P i que implementa les funcionalitats definides en l'Espai d'Aprenentatge.*

En realitat el nom de **Servidor de Coneixement** el veiem com una evolució del **Servidor Personal.** Per aquest motiu mantenim el nom de servidor. No obstant, en aquest cas segurament seria més adequat anomenar-lo **Client-Servidor de Coneixement** ja que com hem comentat ara no només fa tasques de servidor sinó també de client.

Un **Servidor de Coneixement** engegat i amb connexió a Internet es converteix en un **Peer de Coneixement** (o simplement **peer**)

Fins al moment hem parlat d'idees d'aprenentatge en xarxa, e-Learning, hem vist sistemes per facilitar la creació d'entorns e-Learning i hem estudiat diferents opcions per crear la xarxa P2P de Peers de Coneixement. És en aquest punt doncs, on hem de començar a ajuntar tot el que hem vist per definir com serà l'arquitectura d'aquest Servidor de Coneixement que resumint-ho en tres grans trets ha de:

I. implementar com a mínim els serveis bàsics que defineix l'Espai d'Aprenentatge
II. permetre una comunicació P2P entre Servidors de Coneixement
III. realitzar una integració perquè l'usuari vegi tots els serveis i eines com un únic espai

2.1.1 Possibles arquitectures del Servidor de Coneixement

L'arquitectura final que proposarem és resultat de la unió, complementació de diferents sistemes ja existents i d'altres que implementarem nosaltres agrupat tot per un **mòdul de coneixement**. Totes les eines, plataformes o tecnologies que apareixen són "open source".

En les dos arquitectures que proposarem a continuació només hi mostrarem els diferents mòduls que integrem i el mòdul de coneixement. Serà a l'arquitectura final on també afegirem altres serveis o eines que implementarem nosaltres.

Arquitectura I

La primera arquitectura que proposem es basa en el següent:

- Pel que fa a les eines que possibiliten l'aprenentatge en xarxa hem escollit:
  a. per intercanviar coneixement → **ATutor** [12]
  b. per intercanviar informació → **BSCW** [22]
  c. per la comunicació síncrona → **Maptool (permet xat)**
  (la Maptool és una pissarra digital desenvolupada sobre ANTS)

- **Apache** [18]: Un servidor Web necessari pel funcionament de l'ATutor, el BSCW i la Maptool des d'un navegador.

- **ANTS** [19] [20]: Subconjunt de la plataforma ANTS explicada anteriorment i necessària pel correcte funcionament de la Maptool. Permet la creació de sessions amb la Maptool, control d'usuaris, xat, enviament de missatges, etc.

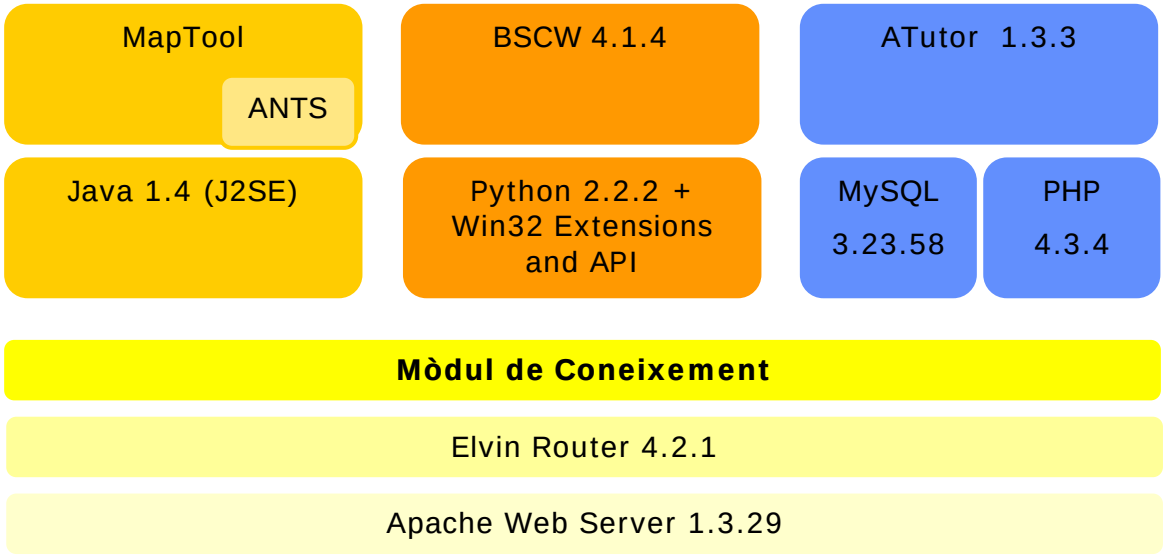
- **ELVIN Router** [21]: Un Servidor de publicació-subscripció de missatges (enviament de missatges un a molts, un a un). És necessari perquè ANTS pugui comunicar-se amb tots els membres que participen en la sessió de pissarra digital.

- **Mòdul de Coneixement**: Aquest l'haurem de desenvolupar nosaltres i s'ocuparà de:

  a) Trobar la resta de participants *(peers)* de la comunitat d'aprenentatge connectant-se amb el *Discovery Server.*
  b) La comunicació entre els Peers de Coneixement. Per tant, tindrà un servidor per rebre peticions d'altres Peers i respondre-les i un client per enviar peticions i rebre'n la resposta.
  c) Crear la interfície gràfica que permeti treballar de forma senzilla, agradable i transparent amb els altres mòduls establint una comunicació amb ells i permetent accedir-hi i trobar-ne informació.

Aquest mòdul es desenvoluparà en JAVA i constituirà l'aplicació principal que gestionarà i encapsularà el nostre Servidor de Coneixement. També inclourà altres funcionalitats educatives com el fòrum, el xat o el taulell d'anuncis com veurem a l'arquitectura final.

En la següent il·lustració tenim un esquema dels diferents mòduls que acabem de comentar:



Il·lustració 2.1 - Arquitectura del Servidor de Coneixement (I)

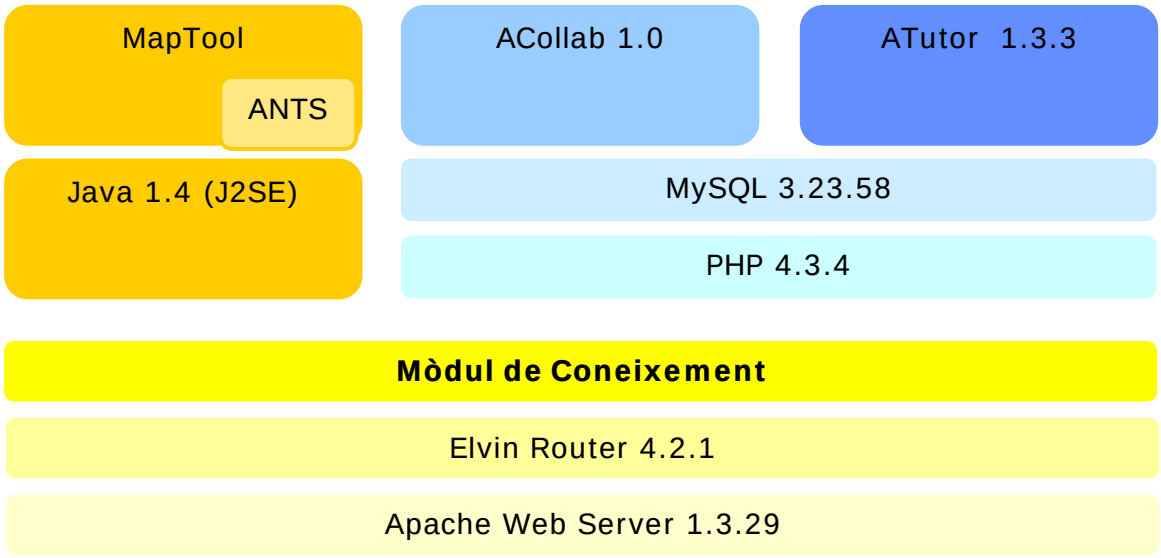
Part del que volem fer nosaltres ho fa el software anomenat Synergeia [23]. Aquest ha estat desenvolupat dins del projecte de recerca ITCOLE fundat per la Unió Europea entre 2001-2003. Combina un component asíncron com és el BSCL (BSCW amb algunes extensions) i un component síncron com és el MapTool (part del sistema ANTS). Synergeia però es basa en el paradigma client-servidor i per tant s'allunya de la nostra idea P2P alhora que no disposa de cap LMS.

Arquitectura II

La segona arquitectura que proposem és idèntica que l'anterior amb l'excepció que canviem l'eina que ens proporciona l'intercanvi d'informació. Si abans utilitzàvem el BSCW, ara ens hem decantat per l'**ACollab** [12]. Les seves funcionalitats bàsiques són molt semblants però tenen algunes diferències a tenir en compte:

- ACollab està desenvolupada pel mateix grup que ATutor per la qual cosa el tema d'interfícies és més coherent i més uniforme que si utilitzem el BSCW. Alhora, ATutor i ACollab estan programats amb PHP mentre que BSCW amb Python

- Mentre el BSCW és àmpliament conegut i utilitzat (sobretot en àmbits universitaris), l'ACollab està en una fase força més embrionària i el seu ús encara és una incògnita



Il·lustració 2.2 - Arquitectura del Servidor de Coneixement (II)

2.1.2 Arquitectura final del Servidor de Coneixement

Un cop vistes i estudiades les dos opcions anteriors podem resoldre que ens quedem amb la segona per les raons que exposarem tot seguit.

La raó de deixar de banda BSCW i escollir ACollab es deu bàsicament a la major facilitat d'integració dels dos components:

→ **ATutor**: Intercanvi de coneixement

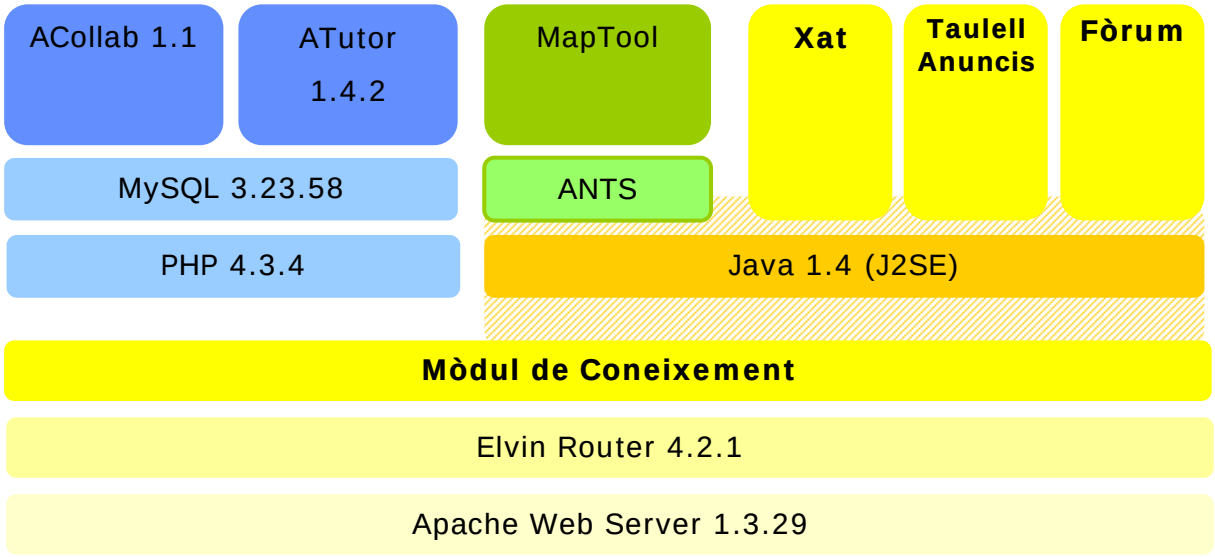
→ **ACollab**: Intercanvi d'informació

D'aquesta manera no barregem tantes tecnologies i no ens caldrà instal·lar un intèrpret de Phyton a cada client. ATutor i ACollab compartiran la mateixa Base de Dades i utilitzaran el mateix mòdul de PHP. Des del punt de vista de disseny gràfic i funcionament també és millor aquesta opció ja que ATutor i ACollab estan desenvolupades pel mateix grup amb la qual cosa tenim un entorn visual i una estructura molt similars facilitant així l'aprenentatge del seu funcionament per part de l'usuari.

Tot i que ACollab no està tant desenvolupat i contrastat com el BSCW creiem que per la proposta que fem aquí és suficientment fiable. ACollab [12] és ideal per grups que treballen

a distància desenvolupant documentació, col·laborant en recerca o escrivint conjuntament articles. També per educadors on-line que vulguin afegir activitats d'aprenentatge en grup al seu curs d'ATutor.

Així doncs, ara ja podem agrupar tots els mòduls que formaran el nostre Servidor de Coneixement, tant els que ja estan implementats com les funcionalitats que afegirem nosaltres. L'arquitectura del Servidor de Coneixement quedarà com mostrem a continuació:



Il·lustració 2.3 – Arquitectura final del Servidor de Coneixement

Podem observar que els serveis o eines educatives que tindrà el Servidor de Coneixement, com a implementació dels requisits educacionals bàsics i/o opcionals de l'Espai d'Aprenentatge, són:

Servei	Mòdul que ho implementa
Intercanvi d'informació	ACollab 1.1
Intercanvi de coneixement	ATutor 1.4.2
Comunicació síncrona: pissarra digital	Maptool
Comunicació síncrona: xat	Mòdul de coneixement – Xat
Taulell d'anuncis	Mòdul de coneixement – Taulell Anuncis
Fòrum	Mòdul de coneixement – Fòrum [3]

Taula 2.1 - Serveis de l'Espai d'Aprenentatge implementats al Servidor de Coneixement

[3] El fòrum estarà emmagatzemat al Discovery Server mentre que al Servidor de Coneixement només hi haurà un visualitzador de missatges

A part d'aquests mòduls o submòduls educatius, al Servidor de Coneixement també hi haurem d'afegir tres funcionalitats bàsiques anteriorment comentades com són:

Funcionalitat	Mòdul que ho implementa
Interfície gràfica que representi un únic espai	Mòdul de coneixement – Interfície gràfica
Trobar la resta de Peers de Coneixement	Mòdul de coneixement – Localització
Permetre comunicació P2P	Mòdul de coneixement – P2P(client/servidor)

Il·lustració 2.4 - Funcionalitats complementàries del Servidor de Coneixement

A partir de les taules i del que hem dit anteriorment, podem resumir què haurem d'implementar en el nostre Mòdul de Coneixement:

- Una interfície gràfica única que integri i que permeti treballar de forma senzilla, agradable i transparent per l'usuari amb tots els mòduls que ofereixen serveis per l'aprenentatge: ATutor, ACollab, Maptool, Xat, Taulell d'Anuncis i Fòrum.
- Una funcionalitat que ens permeti trobar a la resta de participants *(peers)* de la comunitat d'aprenentatge.
- Funcionalitats per fer de client i de servidor. Dit d'una altra forma, ha de poder enviar peticions a d'altres Peers de Coneixement alhora que acceptar i respondre peticions que li arribin.
- Els submoduls que hem vist a l'esquema i que també hem identificat com a serveis necessaris dins l'Espai d'Aprenentatge. Aquests són:
  - Un Xat
  - Un Taulell d'Anuncis
  - Un Fòrum
- També s'afegirà un *log* de tot el que passa al Servidor de Coneixement per tenir-ne un control.

2.2 Accés remot al Servidor de Coneixement personal

Aquest laboratori, com ja hem dit, parteix del "Personal Server Lab" i intentem no perdre de vista les idees allà presents ja que les considerem interessants i complementaries a les d'aquest laboratori. En el "Personal Server Lab" s'introduïa el concepte d'**Espai Personal** com *"un racó nostre a la xarxa, amb els nostres documents, agenda, etc. al que* ***podem accedir des de qualsevol lloc i d'una manera ràpida, senzilla i assequible****. D'aquesta*

manera podem parlar del nostre espai a la xarxa". I s'implementava aquest Espai Personal en un Servidor Personal.

Un dels punts clau en aquest Espai Personal és que s'ha de permetre accedir-hi considerant la mobilitat física dels usuaris que li permeten els nous dispositius mòbils basats en tecnologies com GPRS, WiFi, Bluetooth, UMTS/3G, etc.

Agafant doncs aquesta idea pel nostre laboratori, nosaltres també *volem permetre que el propietari de l'Espai d'Aprenentatge pugui accedir al seu Servidor de Coneixement des de qualsevol punt del planeta amb una connexió a Internet amb qualsevol de les tecnologies existent, inalàmbriques o no, des d'un altre PC, des d'una PDA, etc. Per això, en la nostra proposta d'infrastructura pel coneixement en xarxa afegirem al Servidor de Coneixement un servidor que permeti un accés remot.* La idea és la següent:

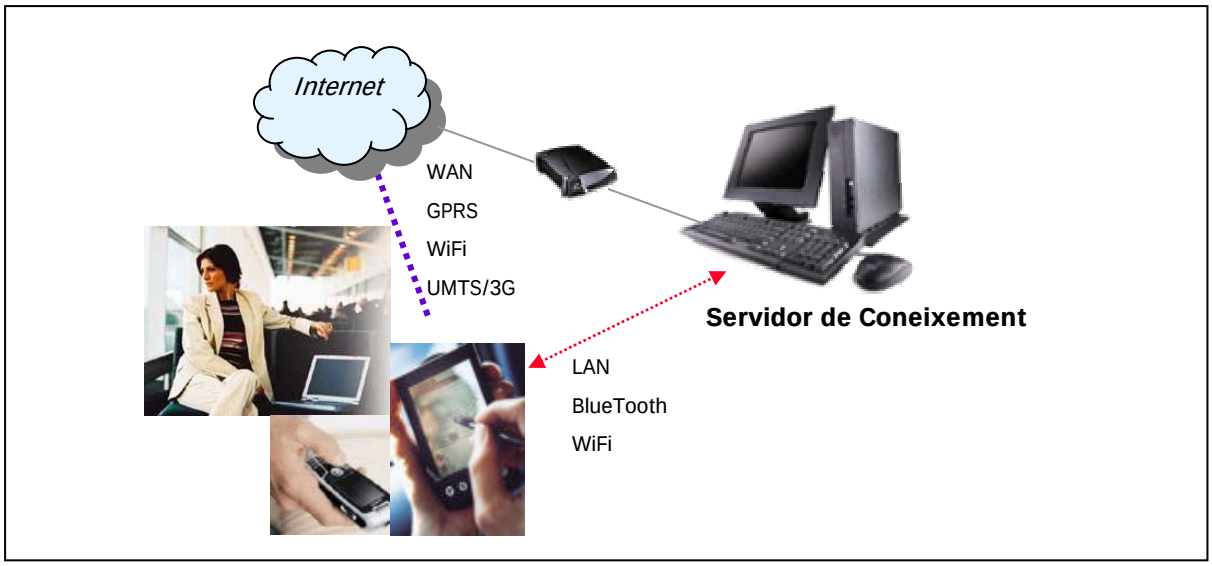


Ilustración 2-1 - Accés remot al Servidor de Coneixement

Nosaltres ho resoldrem instal·lant al nostre Servidor de Coneixement un administrador remot que ens permeti accedir-hi i controlar-lo. Així, en certa manera, tanquem el cercle i unim les idees dels dos laboratoris. Hem estudiat diferents opcions:

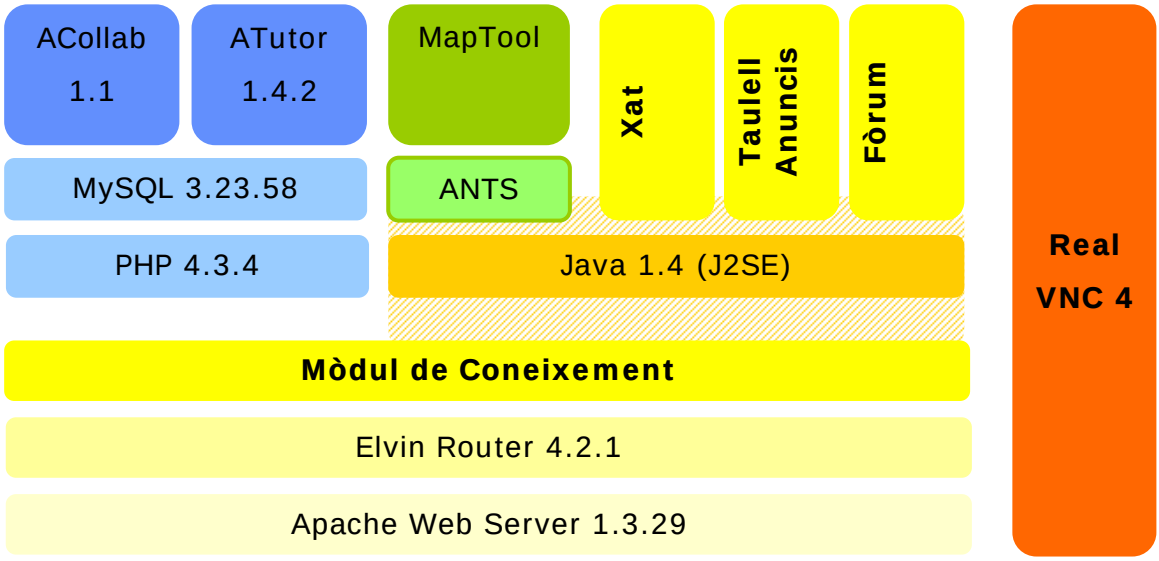
- ✓ Remote Desktop de Windows
- ✓ Remote Administrator (RAdmin) 2.2

  http://www.radmin.com
- ✓ Remotely Anywhere 6

  http://www.remotelyanywhere.com
- ✓ Real VNC 4

  http://www.realvnc.com

L'opció finalment escollida és el Real VNC bàsicament perquè té una versió de lliure distribució (amb menys funcionalitats que l'edició empresarial) i s'adapta perfectament a les nostres necessitats. com veurem en el punt següent. Per la proposta que fem en aquest projecte, creiem que és perfectament vàlida. Abans d'explicar de forma més detallada què és VNC mostrarem com queda l'arquitectura del Servidor de Coneixement un cop hi hem afegit el nostre administrador remot:



Il·lustració 2.5 - Arquitectura final del Servidor de Coneixement amb Real VNC

Hem de comentar que també és un element bàsic pel Servidor de Coneixement el tenir un navegador tot i que no l'hem inclòs a l'esquema ja que avui en dia tots els Sistemes Operatius ja en porten un.

2.2.1 Real VNC

VNC són les sigles de "Virtual Network Computing". És un *software* de sistema remot que permet veure i interactuar amb un ordinador (el "servidor") utilitzant un programa molt simple (el "visor") en un altre ordinador o dispositiu mòbil amb connexió a Internet. En altres paraules, permet a una persona des d'un ordinador remot controlar-ne un altre a través de la xarxa com si hi estigués assegut al davant.

VNC es diferencia d'altres sistemes de control remot en tres punts principals:

1. És multi-plataforma. Un escriptori d'un ordinador Linux es pot visualitzar en un Windows, en un Solaris o en d'altres arquitectures. Hi ha un visor Java que permet que qualsevol

escriptori pugui ser visualitzat amb qualsevol navegador que tingui el *pluggin* de Java **sense haver d'instal·lar cap *software* addicional**. Hi ha un servidor per Windows, que et permet visualitzar un escriptori remot d'un PC amb Windows en qualsevol d'aquestes plataformes utilitzant exactament el mateix visor. La simplicitat del protocol facilita la portabilitat cap a noves plataformes.

2. És elegant i simple. El visor per Windows, per exemple, té una mida aproximada de 150Kb i es pot executar directament des de qualsevol dispositiu d'emmagatzamament (una memòria *flash*, un disquet, etc.). El visor de Java complert ocupa menys de 100Kb i es triga menys temps a baixar-lo de la xarxa que algunes imatges de pàgines web. Aquest visor es descarrega automàticament des del navegador sense que l'usuari hagi d'instal·lar-se ni configurar res. Només s'ha de tenir el *pluggin* de Java del navegador activat.
3. És gratis! Es pot baixar des de http://www.realvnc.com/download.html, utilitzar-lo, i redistribuir-lo sota els termes llicències GNU General Public License.

L'ús que preveiem nosaltres del Real VNC està principalment encarat a:

- Visualitzar i controlar el Servidor de Coneixement des d'un ordinador remot. Per fer-ho, l'opció perfecte que ens ofereix el Real VNC és accedir des del **navegador** de l'ordinador remot al nostre Servidor de coneixement. D'aquesta manera, no tenim que instal·lar cap *software* addicional a l'ordinador remot ni tant sols el visor de 150Kb. Per accedir-hi d'aquesta forma, és tant senzill com posar la següent *url* al navegador:

  `http://pcmarc:5800`

  El servidor VNC porta un petit servidor web que escolta pel port *5800+número de display* esperant connexions *http*. En aquest cas doncs (típic si el servidor és un Windows) ens connectem al *host* pcmarc i al *display* 0. Llavors, l'*applet* que es descarrega et demana el *password* per poder visualitzar l'escriptori remot.

- Visualitzar i controlar el Servidor de Coneixement des d'un dispositiu mòbil. Aquesta és la versió que ofereix més flexibilitat de control del nostre Servidor de Coneixement. Des de qualsevol dispositiu mòbil ens agradaria poder controlar-lo. Així, ho provarem des d'una PDA per la qual hi ha un petit client d'aproximadament 100Kb. En veurem els resultats a l'apartat **¡Error! No se encuentra el origen de la referencia.**.

2.3 Super-Peer

Tot i que en una etapa inicial hem parlat que volem crear una **xarxa d'iguals (P2P)** en la qual cap usuari sigui diferent dels altres, ens trobem amb la necessitat d'afegir a la nostra xarxa un peer amb unes funcionalitats diferents a les dels altres. La primera necessitat que tenim de crear aquest super-peer es deu al problema, ja explicat, de localitzar a la resta de peers dins de la xarxa P2P. Per tant, per poder superar aquest problema existent en el paradigma P2P s'ha decidit suavitzar una mica la condició d'igualtat dins la nostra xarxa i permetre l'existència d'un peer amb unes funcionalitats diferents.

Aquest Super-Peer del que estem parlant s'ocuparà de tenir registrat en tot moment quins Peers de Coneixement estan connectats a la Comunitat Virtual d'Aprenentatge i de respondre a certes peticions d'informació sobre aquests. A aquesta part del Super-Peer l'anomenarem **Discovery Server**.

Justificada la presència d'un Super-Peer dins la nostra infrastructura, el que farem és aprofitar aquest per implementar-hi també l'Espai d'Aprenentatge *Guest* que hem vist en el Working Report WR-2004-01. L'anomenarem el **Guest Server** i "conviurà" dins del Super-Peer amb el Discovery Server.

En resum, tindrem un ordinador que formarà part de la nostra infrastructura d'aprenentatge en xarxa que tindrà les següents característiques bàsiques:

- Estar disponible 7x24
- Allotjar el Discovery Server
- Allotjar el Guest Server

2.3.1 Discovery Server

El Discovery Server o Servidor de Localització s'ocuparà doncs de les següents funcions:

- ✓ Mantenir en una base de dades tots els membres que formen part de la infrastructura d'aprenentatge en xarxa.
- ✓ Mantenir en una base de dades tots els temes i missatges del **Fòrum**. Hem decidit guardar-los en aquest Super-Peer perquè creiem que per pròpia definició, el Fòrum ha de permetre afegir-hi nous missatges així com respondre'n d'altres o simplement consultar-los en qualsevol moment i sense que aquests depenguin de si un Peer de Coneixement està connectat (dins la Comunitat d'Aprenentatge Virtual) o no.

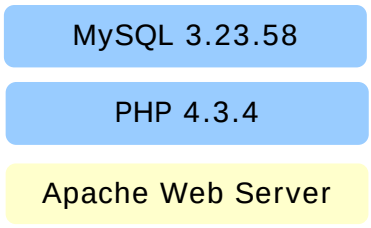
- ✓ Rebre les següents peticions **d'actualització** dels Peers de Coneixement:
  a. Entrada/Sortida de la Comunitat d'Aprenentatge Virtual
  b. Canvi de l'estat d'un Peer de Coneixement (disponible, no disponible o no present)
  c. Engegar o parar una sessió de pissarra digital per part d'un Peer.
  d. Afegir un nou tema o missatge al Fòrum.

  Per cada petició que li vagi arribant haurà de guardar o actualitzar la informació corresponent en una base de dades.

  Alhora, per cada petició que rebi de les tres primeres (a, b, c) haurà de notificar-la a tota la resta de peers presents a la Comunitat d'Aprenentatge Virtual.

- ✓ Respondre a les següents peticions de Peers de Coneixement:
  a. Llista de Peers de Coneixement de la Comunitat Virtual d'Aprenentatge
  b. Llista de temes i missatges del Fòrum

Per poder realitzar totes aquestes tasques, la configuració escollida està composta per una base de dades MySQL, un mòdul dinàmic de PHP i un servidor web Apache.



Il·lustració 2.6 - Configuració del Super-Peer pel Discovery Server

2.3.2 Guest Server

Nosaltres veiem aquest servidor com *una part del Super-Peer amb software específic de servidor que implementa les funcionalitats definides en l'Espai d'Aprenentatge Guest.*

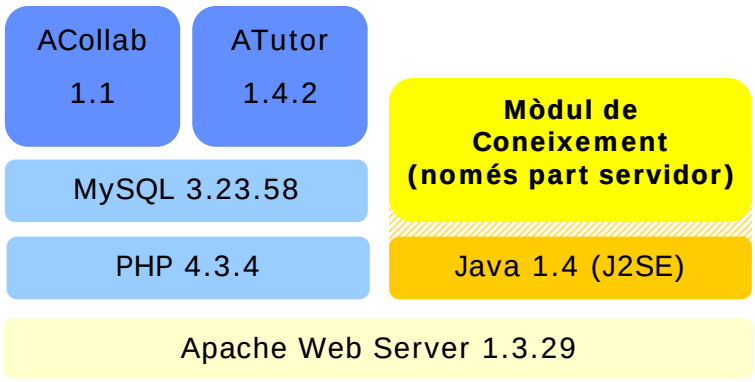
Un usuari que es connecti al **Servidor Guest** (o entri a l'**Espai d'Aprenentatge Guest**) es convertirà en un **Guest de Coneixement** (o simplement **guest**)

Si aquest servidor ha d'implementar les funcionalitats definides a l'Espai d'Aprenentatge Guest ha de:

i. implementar com a mínim els serveis bàsics que defineix l'Espai d'Aprenentatge Guest

ii. crear un espai compartit a la xarxa on puguin accedir tots els usuaris

L'arquitectura que proposem, seguint les idees del que hem fet al Servidor de Coneixement, constarà del que veiem al següent esquema:



Il·lustració 2.7 - Configuració final del Super-Peer

Podem observar que els serveis o eines educatives que tindrà el Guest Server, com a implementació dels requisits bàsics de l'Espai d'Aprenentatge Guest, són:

Servei	Mòdul que ho implementa
Intercanvi d'informació	ACollab 1.1
Intercanvi de coneixement	ATutor 1.4.2
Creació d'un espai a la xarxa Localització Peers de Coneixement i serveis que ofereixen aquests	Aplicació Web amb PHP + MySQL

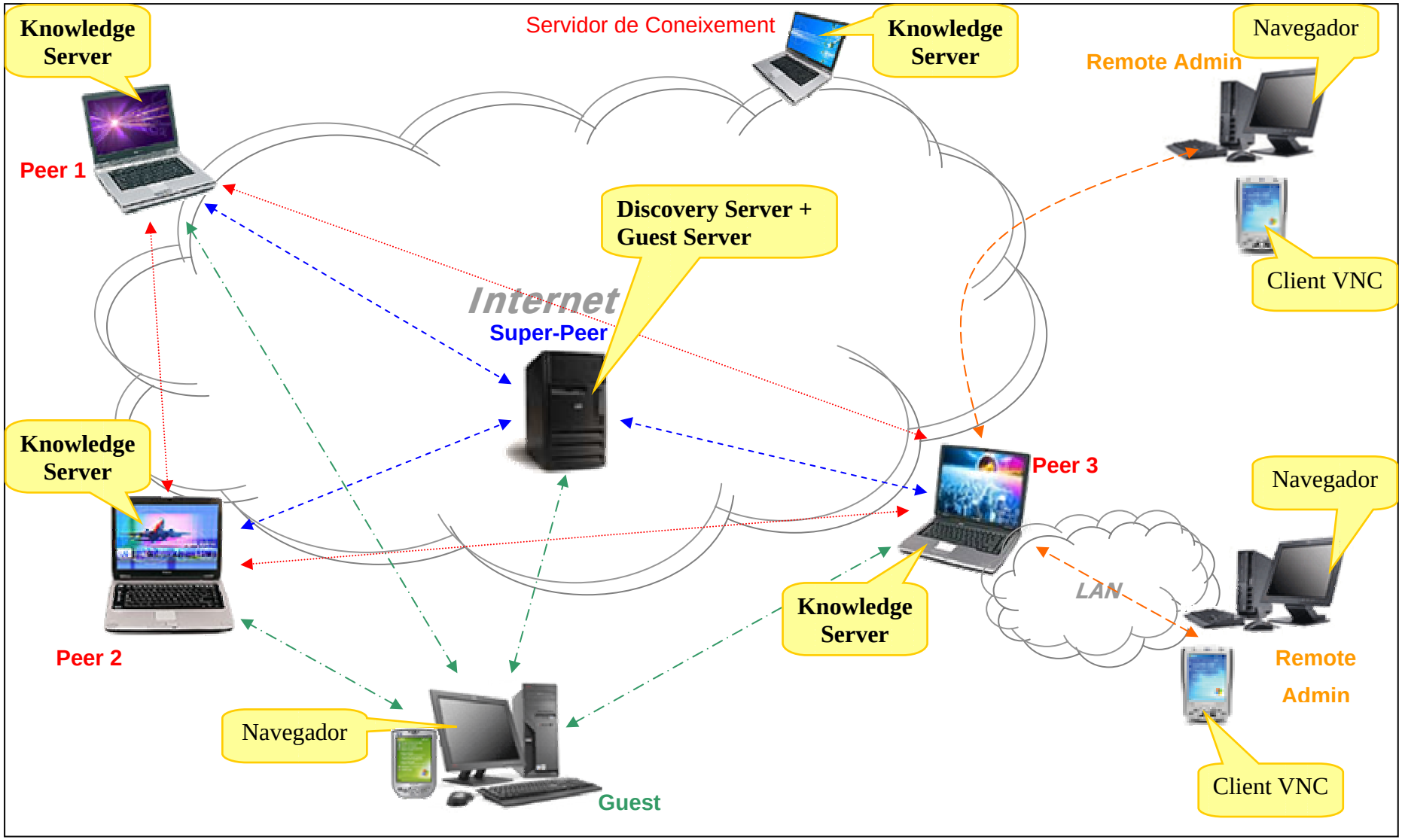
Il·lustració 2.8 - Serveis de l'Espai d'Aprenentatge Guest implementats al Guest Server

Si comparem aquesta configuració amb la de la Il·lustració 2.6 - Configuració del Super-Peer pel Discovery Server veiem que només hi hem hagut d'afegir quatre elements o mòduls nous que són ACollab, ATutor, J2SE i una part del Mòdul de Coneixement desenvolupat pel Servidor de Coneixement. D'aquesta manera, ja tenim la configuració definitiva que necessitem pel Super-Peer.

2.4 Visió global

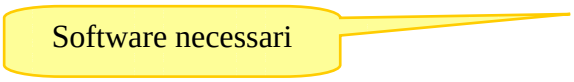
En la il·lustració següent mostrem una visió global de tota la infrastructura per l'aprenentatge en xarxa que proposem en la qual hi apareixen tots els conceptes que acabem d'explicar. Hi podem trobar:

- *un Servidor de Coneixement* : no està dins la Comunitat d'Aprenentatge Virtual
- *tres Peers de Coneixement* : Servidors de Coneixemnt on-line
- *un Guest de Coneixement* : membre de la comunitat que no disposa de Servidor de Coneixement personal
- *el Super-Peer* : allotja el Guest Server i el Discovery Server
- *accés remot LAN* : accés remot a un Servidor de Coneixement personal dins d'una xarxa local amb un ordinador o una PDA
- *accés remot WAN* : accés remot a un Servidor de Coneixement personal des de qualsevol punt d'Internet amb un ordinador o una PDA



Il·lustració 2.9 - Visió global de la Infrastructura d'Aprenentatge en Xarxa

Software necessari en cada element



Aquesta vinyeta associada a cada element ens mostra el software que necessita cada element de la nostra xarxa d'aprenentatge en xarxa. Així, podem observar que:

- un *guest* (equivalent a *guest de coneixement*) només necessita un navegador. Tenint en compte que avui en dia tots els SO porten navegador, podem afirmar que un *guest* es podrà connectar des de qualsevol punt amb connexió a Internet (des d'un PC o des d'una PDA).
- un *peer* (equivalent a *peer de coneixement*) necessita tenir instal·lat al seu PC personal el Servidor de Coneixement (Knowledge Server) amb tots els mòduls que aquest incorpora i que hem vist a l'apartat Arquitectura final del Servidor de Coneixement. Igualment pel Servidor de Coneixement que podem observar, que com ja hem comentat, no deixa de ser un *peer* no connectat a la Comunitat d'Aprenentatge Virtual.
- el Super-Peer necessita l'arquitectura necessària per allotjar el *Guest Server* i el *Discovery Server* que hem vist al punt Super-Peer.
- pels accessos remots a un Servidor de Coneixement només ens cal un navegador des d'un PC mentre que per realitzar-ho des d'una PDA hi hem d'instal·lar un petit client de l'administrador remot que utilitzem: el Real VNC.

Comunicació entre els diferents elements

Tots els elements que apareixen, tenen connexió a Internet directa o estan dins una LAN. La manera com realitzin aquesta connexió ens és indiferent, ja sigui per GPRS, WiFi, UMTS/3G, Bluetooth, ADSL, cable, etc. El que ens interessa és el protocol que s'utilitza en les diferents comunicacions entre elements de la nostra xarxa d'aprenentatge.

Símbol	Comunicació entre...	Protocol
	peer to peer	TCP/IP sockets
	peer to discovery server discovery server to peer	HTTP TCP/IP sockets
	peer to guest server	TCP/IP sockets
	guest to super-peer guest to peer	HTTP
	remote admin to knowledge server	VNC protocol

Taula 2.2 - Protocols utilitzats en les comunicacions entre elements de la xarxa d'aprenentatge

Comunitat d'Aprenentatge Virtual

Dins de la il·lustració anterior podem identificar-hi la Comunitat d'Aprenentatge Virtual que com diu la seva definició està formada pel *subconjunt de la nostra infrastructura per l'aprenentatge en xarxa corresponent a tots els membres d'aquesta que estan on-line*. Aquest subconjunt, el formarien doncs: el peer1, el peer2, el peer3 i el guest.

Podem destacar també la presencia d'un Servidor de Coneixement no actiu o no engegat per la qual cosa en aquest moment no estaria formant part de la CAV encara que s'hi podria afegir en qualsevol moment.

3. Referències

[10] BLACKBOARD. [En línia]. < http://www.blackboard.com >. [Consulta: març 2004]

[11] WEBCT Learning Environment. [En línia]. < http://www.webct.com >. [Consulta: març 2004]

[12] ATUTOR Learning Content Management System. [En línia]. < http://www.atutor.ca >. [Consulta: març 2004]

[13] ILIAS open source. [En línia]. < http://www.ilias.uni-koeln.de/ios/index-e.html >. [Consulta: març 2004]

[14] Future Learning Environment. [En línia]. < http://fle3.uiah.fi/ >. [Consulta: març 2004]

[15] COLLOQUIA. [En línia]. < http://www.colloquia.net >. [Consulta: març 2004]

[16] GROOVE Networks. [En línia]. < http://www.groove.net >. [Consulta: març 2004]

[17] Dreamtech Software India, *Peer-to-peer application development : cracking the code*, New York : Hungry Minds, 2002. ISBN 0-7645-4904-9

[18] The APACHE Software Foundation. [En línia]. < http://apache.org >. [Consulta: març 2004]

[19] GARCIA LOPEZ, P. “Plataforma ANTS: Una arquitectura software basada en componentes para el desarrollo de aplicaciones distribuidas de trabajo colaborativo”. A. F. Gomez Skarmeta (dir.). Tesis doctoral. Universidad de Murcia. Facultad de Informática, Murcia, 2002.

[20] ANTS. [En línia]. < http://ants.etse.urv.es >. [Consulta: abril 2004]

[21] MANTARA Software. [En línia]. < http://www.mantara.com >. [Consulta: abril 2004]

[22] BSCW. [En línia]. < http://bscw.gmd.de >. [Consulta: març 2004]

[23] STAHL, G. *Groupware Goes to School* [En línia]. < http://www.cis.drexel.edu/faculty/gerry/cscl/papers/ch11.pdf >. [Consulta: març 2004]

[24] FLENNER, R. [et al.] *Java P2P Unleashed*, Indianapolis : Sams, cop. 2003. ISBN 0-672-32399-0

[25] SUN Microsystems. *Project JXTA V2.0: Java Programmer's Guide* [En línia]. 2003. < http://www.jxta.org/docs/JxtaProgGuide_v2.pdf > [Consulta: maig 2004]

[26] GRADECKI, J.D. *Mastering JXTA*, Indianapolis : Wiley, 2002. ISBN 0-471-25084-8

[27] GARCÍA LOPEZ, P., GÓMEZ SKARMETA, A. (2003). "ANTS Framework for Cooperative Work Environments". *IEEE Computer*, p. 56-62, Març (ISBN 0018-9162).

[28] GARCÍA LOPEZ P., GOMEZ SKARMETA A., RALLO MOLLA R. (2001). "ANTS: a new Collaborative Learning Framework". *Proc. Of the European Conference on Computer-Supported Collaborative Learning 2001* (ISBN: 90-5681-097-9). Maastricht (Holanda)