

# SVILUPPO DI UN APPLICATIVO CON JXTA (Juxtapose)



**Realizzato da**  
**Ing .Marco Maltraversi e**  
**Alberto Picca**

**In collaborazione con**



**[www.futuresoftware.it](http://www.futuresoftware.it)**

**[info@futuresoftware.it](mailto:info@futuresoftware.it)**

**Progetto:** Sviluppo di un applicativo con JXTA  
**Autori:** Marco Maltraversi, Alberto Picca

**Versione:**

1.12

**Stato:**

Finale

# INDICE

<b>1. PEER-TO-PEER.....</b>	<b>3</b>
1.1 Utilizzo.....	3
1.2 Le prospettive del peer-to-peer dal punto di vista della computer science .....	4
<b>2. DESCRIZIONE DI JXTA .....</b>	<b>6</b>
2.1. Possibilità di JXTA .....	7
2.2. Tipologie di peer .....	7
2.3. I protocolli di JXTA.....	8
2.4. Gli Advertisement.....	11
2.5. Modello Jxta.....	12
2.6. Requisiti per Jxta.....	13
2.7. Sicurezza Jxta.....	13
<b>3. SVILUPPO- JXTA CHAT .....</b>	<b>14</b>
3.1. DESCRIZIONE DEL PACCHETTO SOFTWARE.....	14
3.2. INSTALLAZIONE ED USO.....	14
3.3. JAVADOC .....	19
src Class JXTAChat.....	19
goOn.....	20
FindOrCreate.....	21
FormPrincipale .....	21
currentStatusMsg.....	21
Frame.....	21
JXTAChat.....	21
prepareChat .....	21
initChat .....	21
discoveryEvent .....	21
createChatAdvertisement .....	22
createInputAndOutputPipes .....	22
stopChat.....	22
pipeMsgEvent.....	22
SetTextToSend.....	22
3.4. CONTENUTO E UTILIZZO DEL CD .....	<b>Errore. Il segnalibro non è definito.</b>

## 1. PEER-TO-PEER

Generalmente con il termine “ peer-to-peer “ ( o P2P) si intende una rete di computer o qualsiasi rete che non possiede client o server fissi, ma un numero di *nodi equivalenti* (*peer*, appunto) che fungono sia da client che da server verso altri nodi della rete.

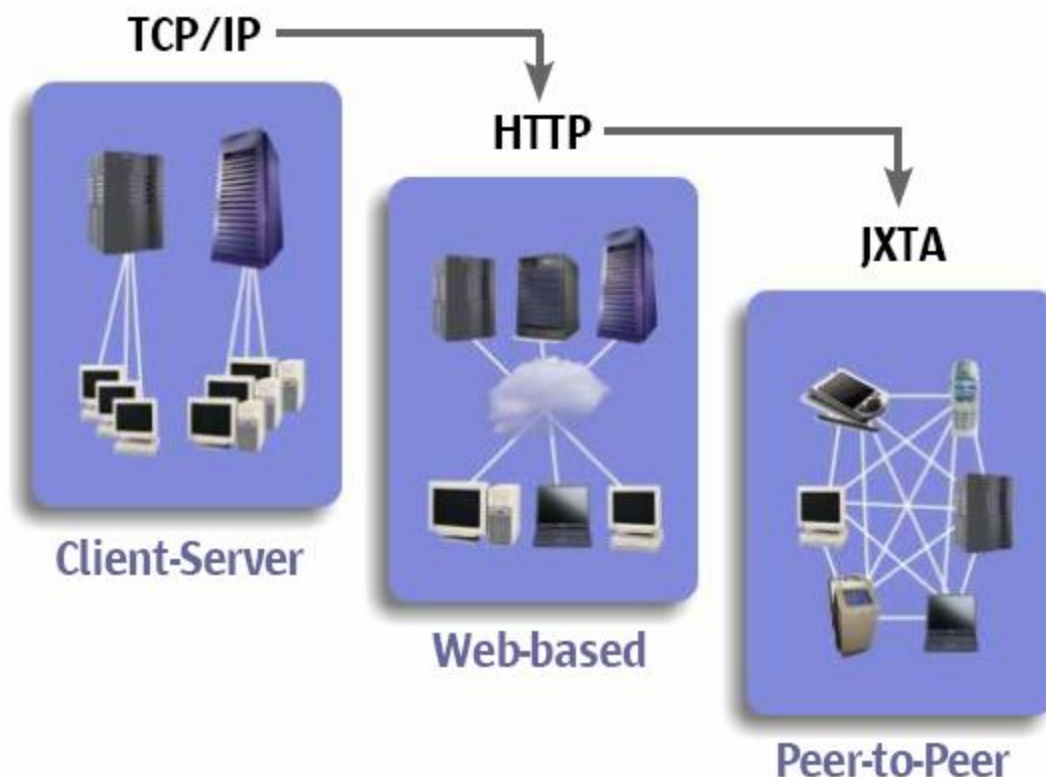
Questo modello di rete è l’antitesi dell’architettura client-server. Mediante questa configurazione, infatti, qualsiasi nodo è in grado di avviare o completare una transazione. I nodi equivalenti possono differire nella configurazione locale, nella velocità di elaborazione, nell’ ampiezza di banda e nella quantità di dati memorizzati. L’esempio classico di P2P è la rete per la condivisione di file.

### 1.1 Utilizzo

Il termine può essere tecnicamente impiegato per qualsiasi tipo di tecnologia di rete e di applicazioni che utilizzano questo modello ARPANET, applets java live chat decentralizzate.

Alcune reti e canali, come per esempio Napster, IRC usano il modello client-server per certi compiti ( per esempio la ricerca ) e il modello peer-to-peer per tutti gli altri.

Quando il vocabolo “ peer-to-peer “ venne utilizzato per descrivere la rete Napster, implicava che la natura a file condivisi del protocollo fosse la cosa più importante, ma in realtà la grande conquista di Napster fu quella di posizionare tutti i computer collegati sullo stesso piano. Il protocollo “peer” era il modo giusto per realizzarlo.



## 1.2 Le prospettive del peer-to-peer dal punto di vista della computer science

Tecnicamente le applicazioni peer-to-peer dovrebbero implementare solo protocolli di *peering* che non riconoscono il concetto di "server e di "client". Tali applicazioni o reti "pure" sono in realtà molto rare. Numerose reti e applicazioni che si descrivono come peer-to-peer fanno però affidamento anche su alcuni elementi "non-peer", come per esempio il DNS. Inoltre applicazioni globali utilizzano spesso protocolli multipli che fungono simultaneamente da *client*, da *server*, e da *peer*. Reti "peers" completamente decentralizzate sono state utilizzate per svariati anni, per esempio USENET.

A partire dal 1990 la SUN aggiunse degli oggetti in linguaggio Java per velocizzare lo sviluppo delle applicazioni P2P. In questo modo i programmatori poterono realizzare delle piccole applicazioni *chat* in real time, prima che divenisse popolare la rete di Instant Messaging. Questi sforzi culminarono con l'attuale progetto JXTA.

I sistemi e le applicazioni Peer-to-peer suscitarono grande attenzione da parte dei ricercatori di *computer science*.

JXTA diverrà, così, parte integrante di Sun ONE, un'infrastruttura tecnologica basata su standard aperti e concorrente della piattaforma Microsoft.NET.

"Il Web si sta evolvendo sia in profondità sia in larghezza in un Web espanso, che rende necessario comunicare e accedere alle risorse su Internet in modo più efficiente", dichiarò Mike Clary, vice presidente del progetto JXTA, "Sun sta offrendo un approccio univoco per indirizzare la prossima fase del calcolo distribuito, un approccio che permetterà agli utenti di cercare, prendere ed usare ciò che desiderano in modo più facile e veloce". Lo slogan è "Find it, Get it, Use it".

JXTA si pone sulla scia di software come Napster e Gnutella, ma rispetto a questi quella di Sun appare come una piattaforma ben più vasta e ambiziosa che ha per obiettivo quello di trasformare il peer-to-peer nel sistema standard per utilizzare la rete: ricerca di informazioni, download di file, streaming di contenuti, fruizione di servizi.

Proprio perché si appoggia su tecnologie aperte come Java e XML, Joy spera che JXTA possa rapidamente divenire uno standard che renda interoperabili fra loro telefoni cellulari, PDA, set-top box computer desktop, server, consentendo ai dati e alle applicazioni di poter essere condivise e fatte girare in modo trasparente su ogni piattaforma che supporti Java.

Dopo un anno di sviluppo, lo scorso aprile Sun ha rilasciato agli sviluppatori tutti gli strumenti necessari per creare applicazioni basate sulla piattaforma JXTA, incluso un primo esempio di client. La casa del Sole ora dichiara che sarebbero più di 50.000 gli sviluppatori che hanno scaricato il codice sorgente del progetto e alcune decine le aziende ad aver iniziato l'evoluzione dei primi applicativi.

Il nome JXTA (abbreviazione di Juxtapose) vuole quindi essere una soluzione per integrare applicazioni e sistemi. I concetti chiave alla base della tecnologia saranno i seguenti:

- Opportunità di aprire dei canali trasmissivi tra peer in modo analogo all'impiego delle pipe del mondo UNIX;
- L'abilità di creare dinamicamente gruppi di peer e gestire le politiche di aggregazione in modo dinamico;
- La definizione di politiche per il monitoraggio dei peer (sia in termini di aggregazione che di presenza);
- La realizzazione di uno strato di sicurezza adeguato.

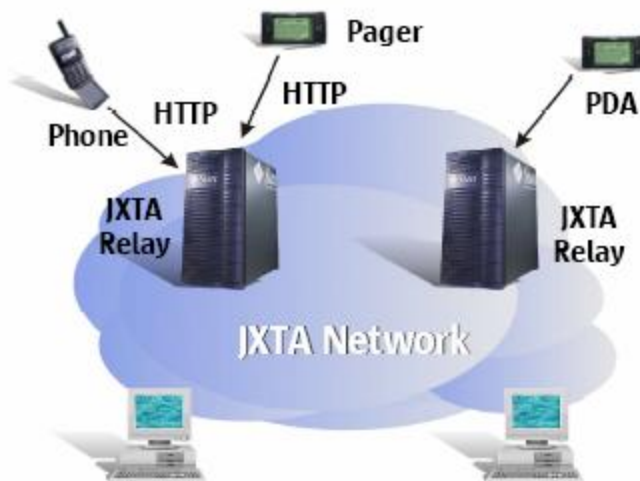
Riassumendo, l'impressione relativamente a queste prime soluzioni "commerciali" del p2p è che c'è ancora molta strada da fare, soprattutto per modellare sui concetti e i protocolli del peer to peer soluzioni per tutte le realtà applicative già esistenti e che, eventualmente, ne possano trarre beneficio.

A un livello di astrazione molto alto, Jxta è semplicemente un insieme di protocolli, dove ognuno è definito da uno o più messaggi scambiati tra i partecipanti; ogni messaggio ha un formato predefinito e può includere svariati campi.

Questi protocolli attualmente sono sei e si occupano di recuperare informazioni sui nodi, interrogarne altri per la ricerca di ulteriori, capire quali servizi offra un nodo e in quale stato (computazionale) sia, richiedere l'appartenenza a un gruppo fornendo delle credenziali (con tutti gli aspetti di security che questo comporta), inoltrare i messaggi ricevuti e richiedere informazioni per l'instradamento dei messaggi. Tutti questi protocolli, com'è ovvio, si basano sull'Xml per la formattazione dei dati.

Ciò che la piattaforma definisce, quindi, è semplicemente come debba avvenire la comunicazione fra nodi, senza dare indicazioni implementative di alcun genere, linguaggio di sviluppo compreso. In molti pensano che l'architettura peer-to-peer sia intrinsecamente legata alla violazione delle leggi sul diritto d'autore, partendo da esempi famosi quali Napster fino ad arrivare alla rete Gnutella.

## 2. DESCRIZIONE DI JXTA



JXTA è un 'interessante materializzazione della mente geniale di Bill Joy, il padre di Java, che già in passato aveva contribuito alla creazione di tecnologie distribuite, come il famosissimo NFS. Presentata intorno al 2001, sembra che JXTA si appresti a diventare la prossima rivoluzione tecnologica legata alla rete.

JXTA è un insieme di protocolli P2P aperti che permette ad ogni dispositivo connesso sulla rete, dai cellulari ai PDA, dai Pc ai server, di comunicare e collaborare come peer. I protocolli di JXTA sono indipendenti sia dai protocolli di trasporto dati che dal linguaggio di programmazione utilizzato, poichè si basano su metadati (dati che descrivono altri dati). Esistono, infatti, implementazioni in C/C++, Perl, Python e naturalmente Java. Da quando la rete ha incominciato a crescere a causa dell'aumento dei dispositivi in grado di connettersi ad una rete con un protocollo qualsiasi, dal Bluetooth al TCP/IP, la filosofia P2P è divenuta popolare. Sebbene il termine "P2P" sia spesso legato a concetti spiacevoli (come la pirateria e la violazione del diritto d'autore), JXTA rappresenta un'evoluzione tecnologica notevole, che non è di esclusivo appannaggio della condivisione dei file, ma può abbracciare numerosi servizi, permettendo, ad esempio, la creazione di gruppi di lavoro e servizi esclusivi. Lo scopo principale di JXTA è quello di provvedere alle funzionalità base dei sistemi P2P. In aggiunta, però, JXTA sviluppa concetti come:

- *Interoperabilità*: la possibilità per ogni peer di fornire servizi diversificati e poterli cercare.
- *Indipendenza dalla piattaforma*: JXTA è progettato per essere indipendente dal linguaggio di programmazione, dai protocolli di trasporto e piattaforma di sviluppo.
- *Ubiquità*: JXTA è stato realizzato per essere accessibile da qualsiasi dispositivo digitale (cellulari, PDA, smartphone) e non solo dai PC.

JXTA crea una rete virtuale nella quale ogni nodo equivalente (*peer*) è in grado di condividere con altri nodi equivalenti (*peers*) una serie di risorse, anche se queste sono protette da un firewall. JXTA prevede, inoltre, un'organizzazione assolutamente decentralizzata, ove non esiste questa differenza sostanziale di importanza tra peer. Anche la gestione delle risorse non è affidata ad un'unità in

particolare, ma è distribuita: quando un peer necessita di una data risorsa, di una certa informazione o di un qualche servizio, invia in rete una richiesta "di aiuto", una *Discovery Query*. Questo è proprio il grande vantaggio di JXTA: la possibilità di un discovery dinamico delle risorse disponibili e dei peer esistenti.

## **2.1. Possibilità di JXTA**

Con l'utilizzo di JXTA è possibile realizzare applicazioni che permettono di:

- Trovare con ricerche dinamiche gli altri peer sulla rete, anche se questi sono protetti da firewall o NAT.
- Condividere senza difficoltà informazioni e documenti.
- Creare gruppi dinamici e fornire servizi.
- Monitorare l'attività dei peer in remoto.
- Comunicare in modo sicuro con gli altri peer della rete.

## **2.2. Tipologie di peer**

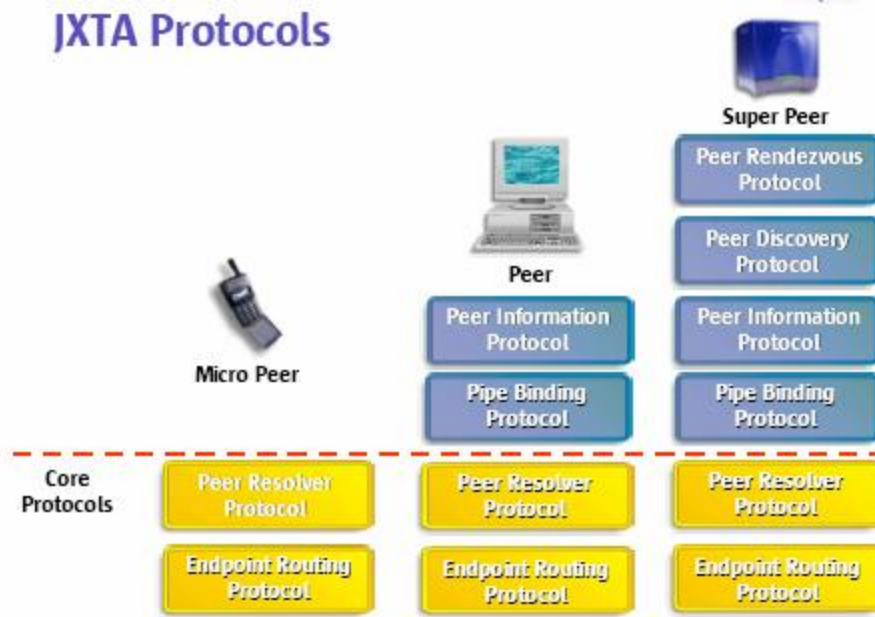
La rete JXTA è una rete adattativa composta da peer connessi.

Le connessioni sono temporanee e l'instradamento tra peer è non deterministico.

Ci sono quattro tipologie di peer all'interno della rete JXTA:

1. Peer semplice:  
può mandare e ricevere messaggi ma non può conservarne o instradare quelli di altri peer. Sono essenzialmente i peer con risorse limitate.
2. Peer pienamente funzionale:  
può mandare e ricevere messaggi e solitamente mantiene in memoria gli advertisement ; tuttavia non inoltra richieste da parte di altri peer.
3. Peer rendezvous:  
può mandare e ricevere messaggi, mantiene in memoria gli advertisement e inoltra richieste da parte di altri peer.
4. Peer relay:  
fornisce un meccanismo client/server che permette la comunicazione con peer inaccessibili poichè dietro NAT/firewall.

## 2.3. I protocolli di JXTA



JXTA definisce una serie di formati di messaggi XML o protocolli per la comunicazione tra i peer. I peer usano questi protocolli per effettuare ricerche, informare e trovare risorse di rete, comunicare e instradare i messaggi. I protocolli JXTA sono divisi in due principali categorie: *Core Specification Protocol - Peer Resolver Protocol*:

Il PRP (che fa parte insieme all'ERP dei CoreSpecificationProtocols ed è, quindi, uno dei protocolli JXTA a livello più basso) definisce, appunto, la struttura dei messaggi (queri e response) per la comunicazione tra i vari peer. Questo non vuol dire che il PRP definisca un certo tipo di servizio per la ricerca di *advertisement* (*Discovery*) o per la ricerca di informazioni sulla rete, etc..., ma ci si aspetta che certi tipi di servizi (come il *DiscoveryService*, ad esempio) usino questa struttura definita dal PRP.

### Endpoint Routing Protocol:

L'Endpoint Routing Protocol è uno dei due Core Protocols, la cui implementazione è obbligatoria per ogni applicazione JXTA. Esso si occupa (come si evince dal nome) del routing dei messaggi, ossia di trovare una sequenza di nodi intermedi (*hops*) che consenta ad un pacchetto dati di giungere dal mittente fino al destinatario. L'uso di tale protocollo è dovuto alla natura intrinseca di JXTA come rete distribuita e dinamica, in cui i peer possono aggiungersi, disconnettersi, usare firewall che rendano impossibile una comunicazione bidirezionale o semplicemente non avere canali di comunicazione diretti fra di loro. Questa dinamicità ha ovviamente lo svantaggio di rendere impossibile l'utilizzo di tabelle di instradamento statiche e note a tutti i peer.

## *Standard Service Protocol*

### *Peer Discovery Protocol:*

Il PDP è un protocollo disegnato per permettere la pubblicazione e la ricerca (*Discovery*) di servizi all'interno della rete JXTA : ogni peer di un gruppo (ovvero ogni peer facente parte della rete JXTA) ha l'opportunità di offrire dei servizi (per esempio un servizio di piping con il quale scambiare dati) a tutta la rete JXTA; le informazioni concernenti i vari servizi devono essere pubblicate nella rete tramite gli *advertisement*.

Grazie al PDP un peer può quindi:

- specificare, e quindi pubblicare, i tipi di servizio che può offrire;
- cercare nella rete JXTA un certo tipo di servizio, ovvero una risorsa messa a disposizione dagli altri utenti della rete.

### *Peer Information Protocol:*

In JXTA la possibilità di ottenere queste informazioni sullo stato di un peer remoto (e non) è offerta dal Peer Information Protocol. Non essendo un core protocol, la sua implementazione è facoltativa: per questo motivo non è garantito che un dato peer risponda alla nostra richiesta di informazioni su di lui (esso potrebbe non aver previsto la gestione del PIP). Tralasciando questa eccezione, il principio di funzionamento del PIP è molto semplice: un peer che vuole ottenere informazioni invia un PeerInfoQueryMessage ad un altro peer. Questi, quando riceverà tale richiesta, controllerà di essere realmente il destinatario designato: in caso positivo risponderà al mittente comunicandogli i dati richiesti; in caso negativo si limiterà a propagare.

### *Pipe Binding Protocol:*

Il PBP permette la creazione di pipe per lo scambio di dati, messaggi ed informazioni in genere. E' quindi "il protocollo finale", quello utile, perchè definisce un canale di comunicazione tra i peer. Ovviamente, facendo parte degli *Standard Service Protocol*, è un protocollo di alto livello, costruito sopra l'Endpoint Routing Protocol : l'informazione fisica è trasportata da quest'ultimo protocollo (tramite i protocolli transport TCP/IP, UDP e HTTP), ma il PBP si preoccupa di definire e di specificare il mezzo "virtuale" di comunicazione tra i peer. Per concludere, i vari messaggi di ricerca di *advertisement*, di informazioni sui peer e comunque tutti i messaggi di servizio sono trasportati dal Peer Resolver Protocol (che ingloba direttamente i messaggi *Discovery* ed *Information*); invece i messaggi veri e propri, quelli che

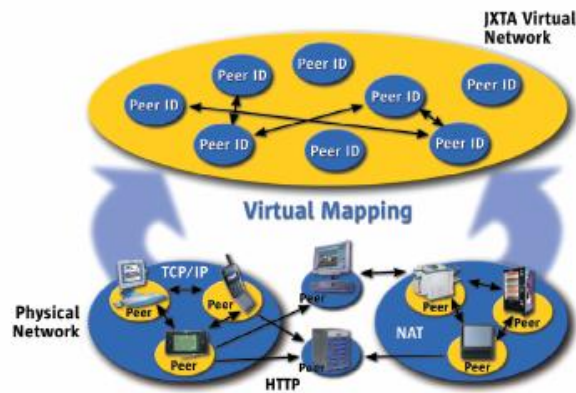
riguardano lo scambio di dati tra i peer, sono gestiti grazie alle pipe create dal PBP.

### Rendezvous Protocol:

I RendezVous Peer sono particolari peer che si distinguono dagli altri per la loro capacità di effettuare il routing e la propagazione dei messaggi. Nelle righe sopra abbiamo già definito come la prima capacità sia implementata dall'Endpoint Routing Protocol.

Tramite il RendezVous Protocol è quindi possibile, ad esempio, utilizzare funzioni di pseudo-broadcast qualora questo non sia supportato dal protocollo di trasporto utilizzato. Il RVP si occupa di stabilire il meccanismo tramite il quale i peer sono in grado di utilizzare o fornire loro stessi il servizio di propagazione.

## 2.4. Gli Advertisement



I protocolli JXTA si basano sull'utilizzo dell'XML per ottenere informazioni autodescrittive. Se da un lato questo permette una notevole interoperabilità ed un utilizzo generale di tale tecnologia, dall'altro appare costoso in termini di numero di messaggi scambiati e occupazione di banda. L'overhead prodotto, infatti, è notevole, come abbiamo potuto sperimentare nel progetto sviluppato. Ciascun advertisement è pubblicato con un ciclo di vita che specifica la disponibilità delle risorse associate. I protocolli JXTA definiscono varie tipologie di annunci:

**Peer Advertisement:** descrivono il peer stesso.

**Peer Group Advertisement:** descrivono il peer group.

**Pipe Advertisement:** descrivono i canali di comunicazione (pipe) che permettono l'effettivo scambio di informazioni.

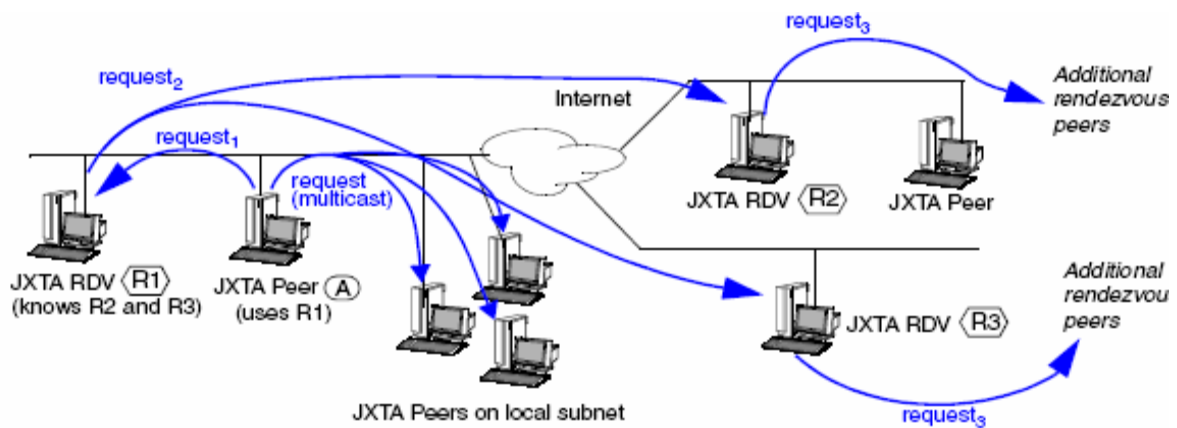
**Module Class Advertisement:** lo scopo principale è documentare l'esistenza di un modulo che fornirà un servizio.

**Module Spec Advertisement:** lo scopo primario è quello di fornire riferimenti alla documentazione necessaria per consentire la creazione di implementazioni conformi alla specifica di modulo.

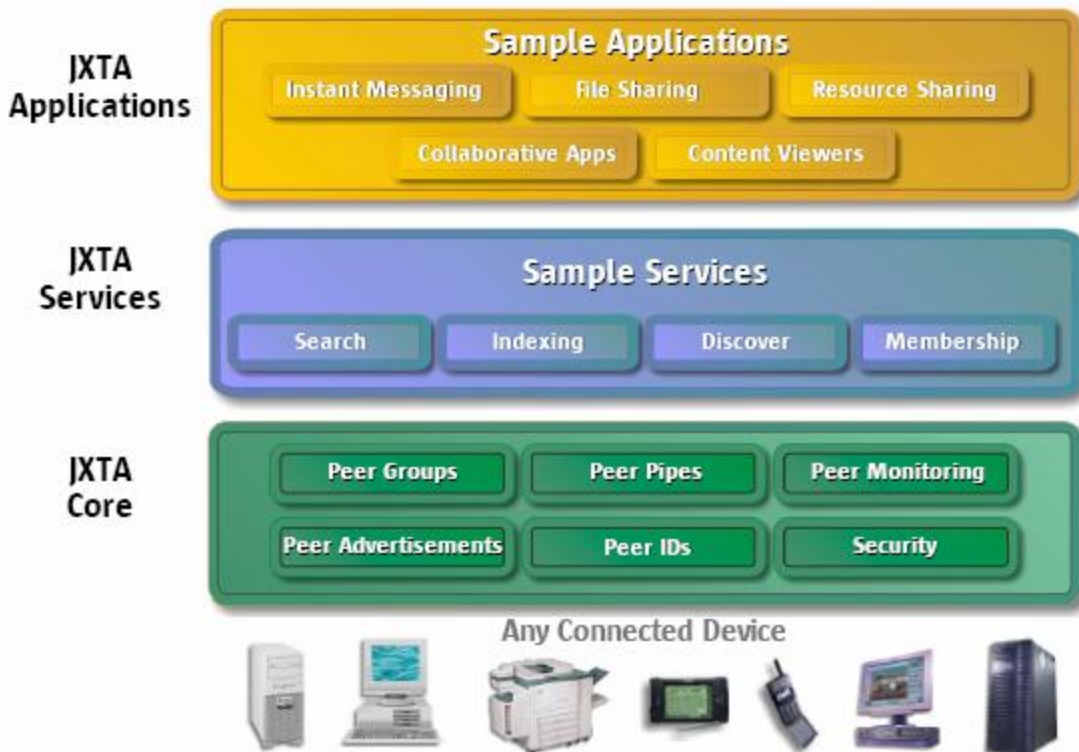
**Module Impl Advertisement:** definisce un' implementazione di una specifica di modulo.

**Rendezvous Advertisement:** descrivono un peer che agisce come rendezvous per un dato gruppo.

**Peer Info Advertisement:** portano con sé informazioni sullo stato di peer e sulle sue statistiche.



## 2.5. Modello Jxta



- **JXTA Core**
  - Assicura uno *uniform peer addressing scheme*
  - Definisce il ruolo dei *relay peers*
  - Funzionalità per creare e cancellare peer group, pubblicizzarli, permettere ad altri peer di trovarli, iscriversi e dis-iscriversi
- **JXTA Services**
  - Funzionalità per indexing, searching, file sharing, ecc.
- **JXTA Applications**
  - Usano le funzionalità di Core e Services: es. JXTA Shell

## 2.6. Requisiti per Jxta

Attualmente il progetto JXTA richiede una piattaforma che supporti JRE (Java Run-Time Environment) oppure SDK (Software Development Kit) versione 1.3.1 o superiore. Pertanto sono supportati i sistemi operativi più diffusi quali i sistemi operativi Solaris, Microsoft Windows 95/98/2000/ME/NT 4.0/XP, Linux, e Macintosh. Per quanto riguarda l'hardware è preferibile un pentium 3 450 Mhz con 64mb di ram (configurazione minima)

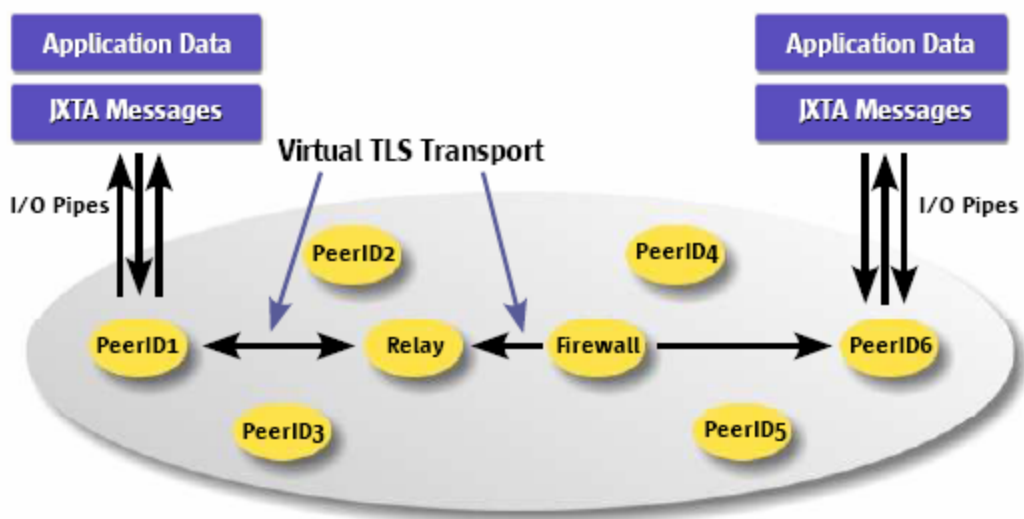
## 2.7. Sicurezza Jxta

*Jxta assicura la sicurezza tramite:*

- *Trasporto di TLS layer security (Strato Sicurezza)*
- *L'indipendenza di trasporto dei protocolli end to end di JXTA*
- *Certificati digitali e certificato di autorità*

*JXTA può offrire rapidamente una forte sicurezza*

1. *TLS Endpoint Trasporto*
2. *Semplice biblioteca crittografica*
3. *Sicurezza peer*
4. *Ogni peer ha il suo proprio certificato di radice*
5. *Certificato credenziale in ogni JXTA di comunicazione di protocollo*
6. *Struttura di autenticazione*
7. *Schema di login*



### 3. SVILUPPO- JXTA CHAT

La nostra applicazione consiste in una chat che implementa le funzionalità della piattaforma jxta le cui potenzialità sono state largamente discusse in precedenza.

Si tratta di una chat grafica in cui è possibile sia creare un nuovo gruppo, sia verificare la presenza di un peer-group già esistente.

In ogni istante è possibile interrompere l'esecuzione premendo il pulsante "stop" dell'applicazione per poter accedere alla ricerca di un nuovo utente o alla creazione di un nuovo gruppo.

L'interfaccia grafica risulta semplice e di facile utilizzo.

Per un corretto funzionamento dell'applicazione bisogna avere installato sul proprio computer la JVM (Java Virtual Machine) e seguire le seguenti istruzioni da noi indicate.

Verificare, inoltre, di avere tutte le librerie richieste, disporre di un compilatore java, di un collegamento alla rete internet e di configurare in modo opportuno la console jxta.

#### 3.1. DESCRIZIONE DEL PACCHETTO SOFTWARE

Per far funzionare la nostra applicazione sono necessarie le seguenti librerie

##### Librerie



#### 3.2. INSTALLAZIONE ED USO

E' fondamentale settare nella variabile di ambiente CLASSPATH il classpath di tutti i file jar di j2sdk e di jxta

A questo punto siamo pronti per lanciare il class "main" del progetto .

E' consigliabile scaricare un compilatore java (es: Netbeans o simili) e caricare come progetto il sorgente con le librerie e compilare ed eseguire l'applicativo. Se non si dispone di un compilatore java usare i comandi dos.

È possibile scrivere, compilare ed eseguire un programma Java in ambiente Windows, ovvero senza usare ambienti di sviluppo aggiuntivi.

Copiare tutto il contenuto del file sorgente all'interno di una cartella del proprio computer, ad esempio "C:\".

Digitare al prompt dei comandi i seguenti comandi per esempio:

```
> c:  
> cd \code\src
```

La compilazione del file avviene tramite il comando **javac** (Java Compiler):

```
> javac Main.java
```

Se non viene visualizzato alcun messaggio, allora il programma è stato compilato correttamente ed è stato generato il file Main.class. Se si sono verificati errori durante la compilazione questi verranno segnalati e il programma dovrà essere corretto.

Ora occorre compilare la classe JxtaChat.class, quindi:

```
> javac JXTAChat.java
```

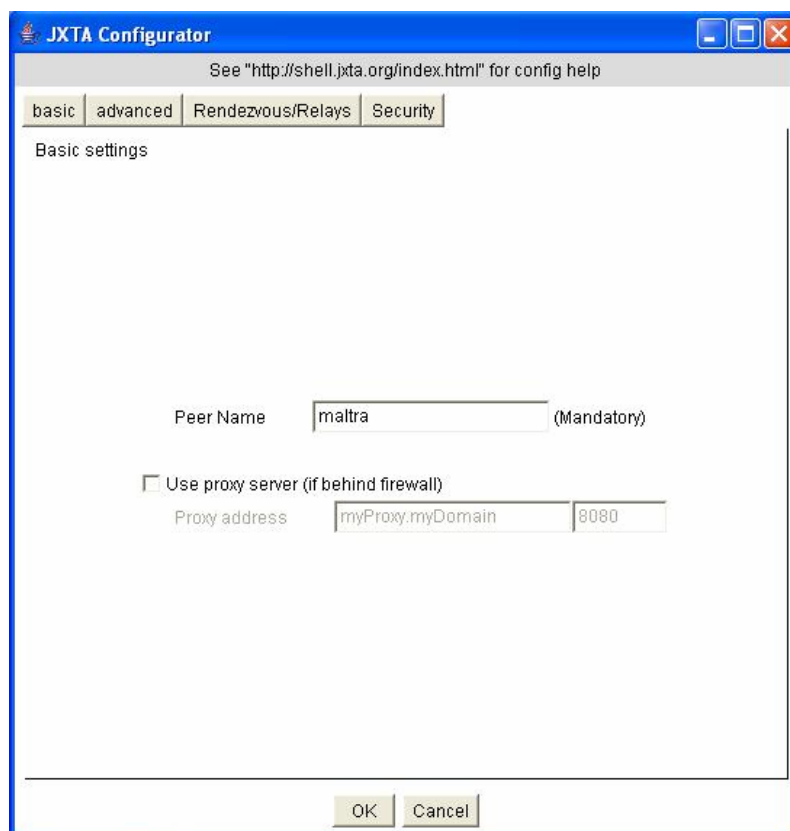
L'esecuzione di un programma Java avviene mediante il comando java (Java Interpreter):

```
> java Main
```

Si noti che non va aggiunta l'estensione .class e che le lettere maiuscole/minuscole sono significative nell'uso di questo comando, cioè devono corrispondere esattamente al nome della classe che è stata definita.

**Se sorgono problemi nella compilazione utilizzare i file precompilati (.class) presenti nel progetto.**

Una volta lanciato l'engine, apparirà una schermata di configurazione (JXTA Configurator). E' necessario inserire un nome, che identificherà il peer in rete, nella tab *Basic Settings*. In *Advanced* è possibile settare il livello di verbose della console (*Trace Level*) e passare alle impostazioni fondamentali per un corretto funzionamento dell'applicazione.



La scheda successiva (*Rendezvous/Relays*) permette di impostare l'utilizzo di relay e rendezvous. E' possibile scegliere di essere un rendezvous e impostarne la lista di rendezvous di boot, che vengono interrogati come ultima risorsa in caso non ne venissero trovati altri, oppure impostare i relay da utilizzare necessariamente in caso di presenza di firewall/NAT.

**Per utilizzare in maniera corretta JxtaChat è fondamentale selezionare "Use only configured rendez-vous" e "Act as a rendezvous".**

In caso che la chat non funzionasse con i peer presenti configurare la console così:

IL PRIMO PEER configurarlo come rendezvous (nella finestra rendezvous/relays selezionare SOLO "act as a rendezvous" e "use only configured rendezvous"; e poi in Advanced abilitare le incoming connections di HTTP)

IL SECONDO PEER, configurarlo come edge e imporre che si connetta al primo peer che è stato attivato (nella finestra rendezvous/relays non selezionare nulla ma scrivere nei campi: Available TCP rendez-vous - l'indirizzo IP del primo peer che è stato attivato, e la porta 9701; Available HTTP rendez-vous - l'indirizzo IP del primo peer che è stato attivato, e la porta 9700; e poi cliccare + in entrambi per farli memorizzare; e poi in Advanced abilitare le incoming connections di HTTP)

**JXTA Configurator** See "http://shell.jxta.org/index.html" for config help

basic | advanced | Rendezvous/Relays | Security

Experienced Users Only

Trace Level

**TCP Settings**

Enable if direct LAN connection.

Enabled  Multicast

Manual

Enable Outgoing connections

Enable Incoming Connections

(Optional) Public address

**HTTP Settings**

Must enable if behind Firewall or NAT

Enabled

Manual

Enable Outgoing connections

Enable Incoming Connections

(Optional) Public address

OK Cancel

Una volta configurato la jxta console partirà l'applicativo che si presenta così:



Da qui si può creare o trovare un nuovo gruppo e chattare con i vari utenti.

Il codice sorgente è stato appropriatamente commentato per una migliore comprensione da parte dell'utente inoltre è disponibile un piccolo sito web di supporto per capire in modo rapido le funzionalità di jxta e le sue potenzialità. Il javadoc contiene l'insieme delle procedure implementate nella jxtaChat.

Il progetto è stato realizzato da Marco Maltraversi e Alberto Picca per il corso Reti di Calcolatori.

### 3.3. JAVADOC

**src**

#### **Class JXTAChat**

java.lang.Object

└ **src.JXTAChat**

#### **All Implemented Interfaces:**

net.jxta.discovery.DiscoveryListener, java.util.EventListener, net.jxta.pipe.PipeMsgListener

---

```
public class JXTAChat
extends java.lang.Object
implements net.jxta.pipe.PipeMsgListener, net.jxta.discovery.DiscoveryListener
```

Questa classe rappresenta la pura implementazione del cuore della Chat. E' stata realizzata avvalendosi della libreria open-source "JXTA". Questo progetto é stato sviluppato per il corso di Reti di calcolatori

#### **Field Summary**

java.lang.String	<a href="#"><u>currentStatusMsg</u></a> Variabile che permette di indicare i messaggi di stato che devono essere mostrati all'utente.
boolean	<a href="#"><u>FindOrCreate</u></a> Boolean che permette di indicare se occorre creare un nuovo gruppo oppure occorre effettuare una ricerca
src.MainForm	<a href="#"><u>FormPrincipale</u></a> GUI frame dell'applicazione.
javax.swing.JFrame	<a href="#"><u>Frame</u></a>
boolean	<a href="#"><u>goOn</u></a> Boolean usato per indicare lo stato della chat goOn = true --> chat attiva goOn = false --> chat non attiva

## Constructor Summary

[JXTAChat\(\)](#)

Main constructor.

## Method Summary

[createChatAdvertisement\(\)](#)

Metodo usato per creare un adv Il nome associato all'adv creato é composto da: prefisso + indirizzo IP della macchina + data corrente In questo modo possiamo rendere univoco un adv

[createInputAndOutputPipes](#)(net.jxta.protocol.PipeAdvertisement chatAdv)

Metodo usato per creare una pipe di ingresso e di uscita per l'adv

[discoveryEvent](#)(net.jxta.discovery.DiscoveryEvent event)

Metodo usato per ricevere un nuovo evento legato al Discovery

[initChat\(\)](#)

Metodo che deve essere richiamato dopo prepareChat().

[pipeMsgEvent](#)(net.jxta.pipe.PipeMsgEvent pipeMsgEvent)

Questo metodo viene utilizzato per gestire i messaggi inoltrati e ricevuti nella chat.

[prepareChat\(\)](#)

Metodo usato per preparare la chat

[SetTextToSend](#)(java.lang.String[] line)

Metodo usato per associare alla variabile privata 'lineToSend' il valore passato come parametro.

[stopChat\(\)](#)

Metodo usato per fermare la chat.

## Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### goOn

```
public boolean goOn
```

Boolean usato per indicare lo stato della chat goOn = true --> chat attiva goOn = false --> chat non attiva

## FindOrCreate

public boolean **FindOrCreate**

Boolean che permette di indicare se occorre creare un nuovo gruppo oppure occorre effettuare una ricerca

---

## FormPrincipale

public src.MainForm **FormPrincipale**

GUI frame dell'applicazione.

---

## currentStatusMsg

public java.lang.String **currentStatusMsg**

Variabile che permette di indicare i messaggi di stato che devono essere mostrati all'utente.

---

## Frame

public javax.swing.JFrame **Frame**

## Constructor Detail

### JXTAChat

public **JXTAChat**()

Main constructor. Metodo per la costruzione della Chat. Permette la creazione del Peer-group e l'inizializzazione dei componenti.

## Method Detail

### prepareChat

public void **prepareChat**()

throws java.lang.Exception

Metodo usato per preparare la chat

#### Throws:

java.lang.Exception - : eventuale exception

---

### initChat

public void **initChat**()

throws java.lang.Exception

Metodo che deve essere richiamato dopo prepareChat() . Permette la creazione di un nuovo ChatAdvertisement, oppure inizia la ricerca di un ChatAdvertisement.

#### Throws:

java.lang.Exception - : Se scatta una peer-group Exception

---

### discoveryEvent

public void **discoveryEvent**(net.jxta.discovery.DiscoveryEvent event)

Metodo usato per ricevere un nuovo evento legato al Discovery

#### Specified by:

discoveryEvent in interface net.jxta.discovery.DiscoveryListener

#### Throws:

java.lang.Exception - : Se scatta errore durante la ricerca

---

### **createChatAdvertisement**

public net.jxta.protocol.PipeAdvertisement **createChatAdvertisement**()

Metodo usato per creare un adv Il nome associato all'adv creato é composto da: prefisso + indirizzo IP della macchina + data corrente In questo modo possiamo rendere univoco un adv

**Returns:**

PipeAdvertisement: puntatore all'adv creato.

---

### **createInputAndOutputPipes**

public void

**createInputAndOutputPipes**(net.jxta.protocol.PipeAdvertisement chatAdv)

throws java.lang.Exception

Metodo usato per creare una pipe di ingresso e di uscita per l'adv

**Throws:**

java.lang.Exception

---

### **stopChat**

public void **stopChat**()

Metodo usato per fermare la chat. Questa procedura riporta il sistema ad uno stato iniziale, nel quale l'utente potrà ancora scegliere se crea un peer-group oppure se cercarne uno.

---

### **pipeMsgEvent**

public void **pipeMsgEvent**(net.jxta.pipe.PipeMsgEvent pipeMsgEvent)

Questo metodo viene utilizzato per gestire i messaggi inoltrati e ricevuti nella chat.

**Specified by:**

pipeMsgEvent in interface net.jxta.pipe.PipeMsgListener

---

### **SetTextToSend**

public void **SetTextToSend**(java.lang.String[] line)

Metodo usato per associare alla variabile privata 'lineToSend' il valore passato come parametro.

---