

Programmation d'applications P2P

- P-Grid : www.p-grid.org
 - Implantation de l'algorithme P2P structuré P-Grid
- Pastry : deux implantations freepastry (Rice University) et SimPastry/VisPastry (Microsoft Research)
 - Couche bas niveau de type P2P structuré Plusieurs applications construites au-dessus
- Neurogrid : www.neurogrid.net
 - Algorithme de recherche à la freenet
 - Outil de simulation d'algorithmes P2P
- JXTA : www.jxta.org

1

JXTA



2

Merci à S. Baehni (EPFL-Distributed Programming Laboratory) et M. Jan (IRISA Projet Paris) pour la mise à disposition de leurs présentations sur JXTA.

3

Caractéristiques P2P

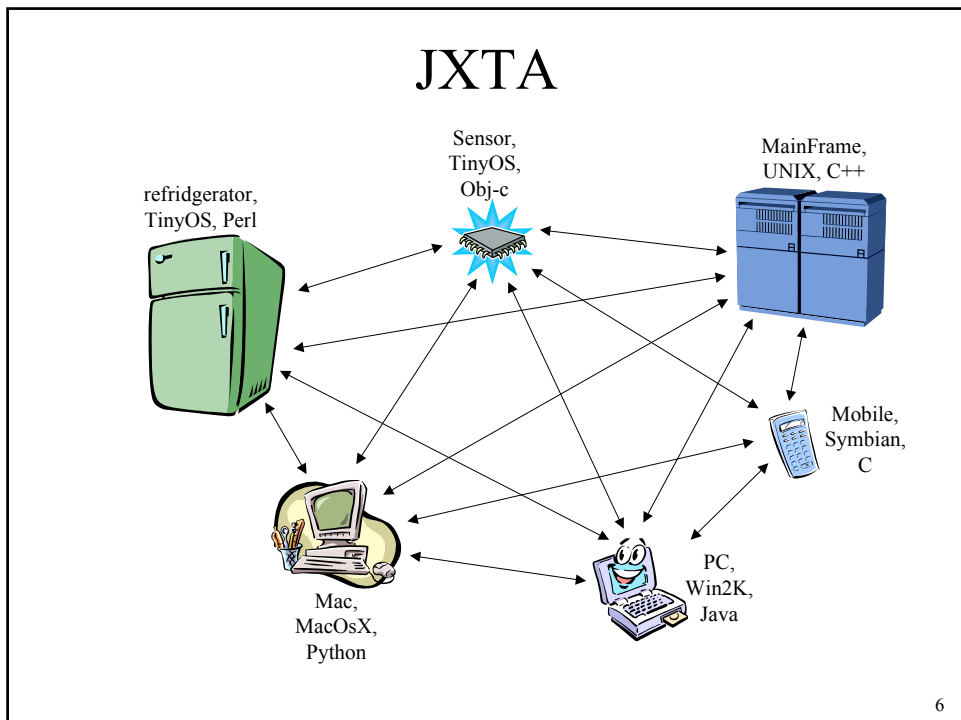
- Dynamique du réseau
 - Composition et topologie
- Découverte dynamique de peers et de ressources
- Passage à l'échelle
- Haute disponibilité (réplication)

4

Qu'est ce que JXTA?

- JXTA est un ensemble de protocoles P2P ouverts
- JXTA offre des communications via des messages XML.
- JXTA permet à n'importe quelle entité sur le réseau (téléphones cellulaires, PDAs, Serveurs et PCs) de communiquer.

5



6

JXTA

- Abréviation de juxtaposition:
 - « putting things next to each other »
- Concept de Bill Joy (Chief Scientist, Sun)
- Approche très nouvelle
 - Première présentation le 15.02.2001
 - Web-site (www.jxta.org) ouvert le 24.04.2001
- Une spécification définissant les concepts de base pour créer des applications P2P [sJxta03]
- Un projet open-source autour de la spécification (Projet JXTA)

7

Qu'est JXTA?

- La plateforme JXTA standardise la manière avec laquelle les peers:
 - Se découvrent les uns les autres
 - Publient leurs ressources réseaux
 - Communiquent entre eux
 - Coopèrent entre eux pour former des peer groups sécurisés

8

Pourquoi JXTA?

- Applications P2P sont en grande expansion.
- les programmes P2P actuels implantent généralement une seule fonction.
- les programmes P2P actuels fonctionnent sur une seule plateforme (contraire aux principes P2P).
- La plupart des applications P2P sont incapables d'échanger directement des données avec des applications similaires (Kazaa/Gnutella, ICQ/AIM/MSIM).

9

Que fournit JXTA?

- JXTA crée une plateforme commune pour les Applications P2P.
- JXTA rend le développement simple, rapide et facile (en théorie).
- JXTA permet aux développeurs de se focaliser sur les couches hautes des applications.

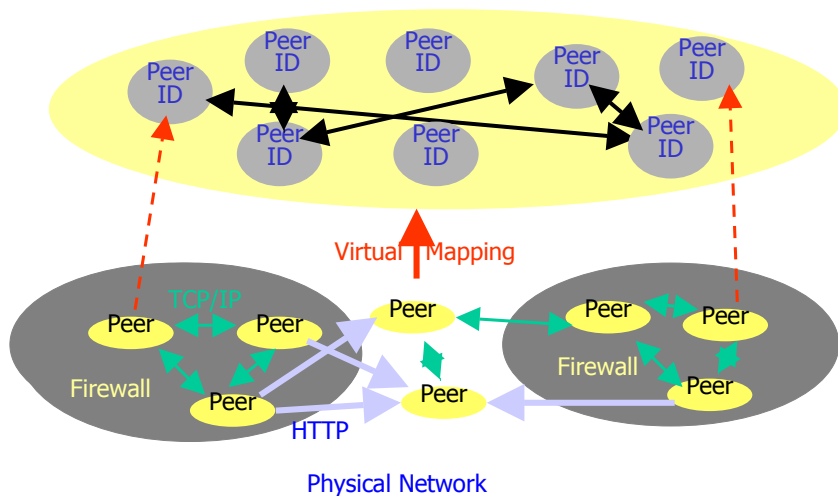
10

Quel langage de programmation et protocole?

- JXTA est indépendant des langages de programmation. C'est un framework, pas une API.
- APIs JXTA sont disponibles pour C/C++, Java, Perl, Python, Ruby, Objective C, Smalltalk et plus.
- JXTA est indépendant des protocoles de transport.
- JXTA peut être implémenté sur TCP/IP, sur HTTP, avec Bluetooth, ou n'importe quoi d'autre!

11

Réseau virtuel JXTA



12

JXTA

- 3 principaux niveaux
 - Core layer (platform)
 - Minimal, primitives essentiels communs aux réseaux P2P
 - communication basique, mécanismes de découverte
 - Sécurité basique, mécanisme d'appartenance (groupe)
 - Mécanismes basiques de surveillance (monitoring)
 - Service layer
 - Ensemble de services sur JXTA
 - pas nécessaire mais intéressant
 - cms, jxtaspaces, jxta-rmi
 - Application layer
 - Utilise le service layer
 - Liaison souple avec le service layer
 - myJxta, jxAuction, vop2p

13

JXTA

JXTA MODEL



14

JXTA

- Core layer est basé sur :
 - Concepts
 - Protocoles
- Concepts
 - Peers
 - Peer groups
 - Network Services
 - Pipes
 - Advertisements
 - Messages
 - Identifiers
 - Credentials
 - Contents

15

JXTA

- Peer
 - N'importe quelle entité de réseau utilisant JXTA
 - Possède un ID unique, permettant un adressage indépendant de la localisation physique
 - Peut avoir plusieurs adresses physiques (endpoints)
 - Pas d'hypothèse sur la disponibilité

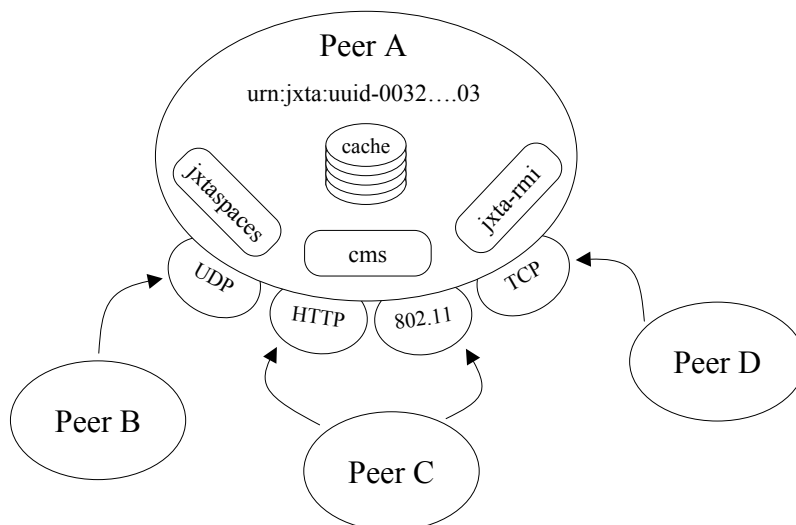
16

JXTA

- Différents types de peers
 - Minimal : juste send/receive
 - normal : + cache
 - Rendezvous : + fwd requests
 - Relay : +routing cache +firewall support
- Relay peers
 - Utilisés pour router des messages
 - Supportent des transferts de message multi-hop
 - Maintiennent dynamiquement les informations de routage
 - Gèrent en cache les messages pour les peers temporairement indisponibles
 - utilisés pour traverser les firewalls et les NAT
- Rendez-vous peers
 - utilisés pour indexer les advertisements et pour propager les messages
 - Jouent le rôle de points d'entrée pour le réseau JXTA

17

JXTA



18

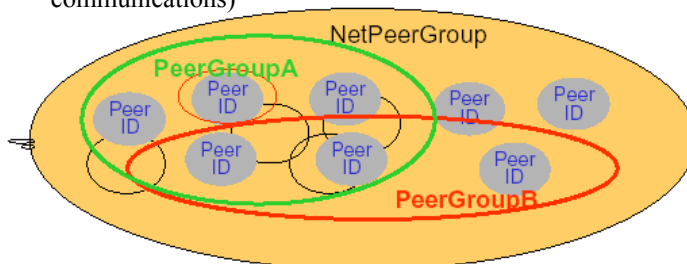
JXTA

- Peer group
 - Collection de peers partageant un intérêt commun
 - Crée un environnement sécurisé, délimité et surveillé
 - fournit un ensemble de core services
 - Discovery service
 - Membership service
 - Pipe service
 - Resolver service
 - Monitoring service
 - un peer peut se joindre à plusieurs peer groups
 - Tous les peers appartiennent par défaut au « NetPeerGroup »

19

Peer Groups

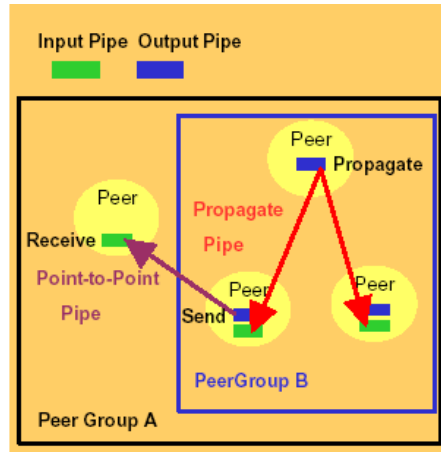
- Pourquoi?
 - Intérêt commun
 - Domaine sécurisé et protégé
 - Permet la surveillance
 - Définit une portée d'opérations (découverte, recherche, communications)



20

Pipe

- Canal de communication abstrait entre 2 ou plus peers
 - Uni-directionnel
 - Asynchrone
 - Non fiable
 - Propagate pipe (many-to-many, dans la portée d'un peer group)
 - Enhanced pipe (sécurisé, bidirectionnel)



21

Modèle de Communication Pipe

- Connecte services indépendamment des localisations des peers
- Liaison dynamique
 - À la la création du pipe ou pour chaque message échangé
- Construire des connections à haute-disponibilité
 - Reconnexion automatique
- Construction de services complexes par pipelines

22

JXTA

- Advertisements
 - Toutes les ressources réseaux ont des advertisements
 - Peer advertisement
 - Peer group advertisement
 - Pipe advertisement
 - Module (service) advertisement
 - Content advertisement
 - ...
 - Métadonnées pour décrire des ressources
 - Représentées avec XML
 - Ont un age local et un age distant

23

JXTA

• Peer Advertisement

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PA>
<jxta:PA xmlns:jxta="http://jxta.org">
  <PID> urn:jxta:uuid-59616261646162...C88E08168F0E4BDA903 </PID>
  <GID> urn:jxta:uuid-FF1D3F7597584238B2F89F9E87AD51F602 </GID>
  <Name> Seb </Name>
  <Svc>
    <MCID> urn:jxta:uuid-DEADBEEFDEAFB...0000000805 </MCID>
    <Parm>
      <Addr> tcp://128.178.73.43:9701/ </Addr>
      <Addr> jxtatls://uuid-59.....03/TlsTransport/jxta-
        WorldGroup </Addr>
      <Addr> jxta://uuid-59.....903/ </Addr>
      <Addr> http://JxtaHttpClientuuid-59.....903/ </Addr>
    </Parm>
  </Svc>
</jxta:PA>
```

24

JXTA

- Messages
 - Unité d'échange de données entre peers
 - deux représentations
 - XML/Binary
 - Dépend du protocole de transport utilisé
 - Ensemble de couples nom/valeur (element)
 - Element
 - Ajouté lors du passage entrant par la pile de protocoles
 - Supprimé lors du passage sortant
 - XML car
 - Langage neutre
 - auto-descriptif
 - Permet de vérifications syntaxiques

25

JXTA

- Messages (XML format)

```
<?xml version="1.0"?>
<!DOCTYPE Message>
<Message version="0">
  <Element name="jxta:SourceAddress" mime_type="text/plain"> tcp://123.456.205.212
</Element>
  <Element name="stuff" encoding="base64" mime_type="application/octet-stream">
AAECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISljJCUmJygpKissL.S4vMD
EyMzQ1Njc4OT07PD0+P0BBQkNERUZHSElKS0xNTk9QUVJTVFVWV1hZWlhcXV5fYGFi
Y2RIZmdoaWprbG1ub3BxcnN0dXZ3eH16e3x9fn+AgYKDhIWGh4iJiouMjY6PkJGSk5SVlpeY
mZqbnJ2en6ChoqOkpaanqKmq q6ytrq+wsbKztLW2t7i5uru8vb6/wMHCw8TFxsc=
  </Element>
</Message>
```

26

JXTA

- Identifier
 - Identifie de manière unique une entité et sert à la localiser
 - Pas de serveur centralisé
 - Présenté comme URN
 - Propriétés :
 - Non ambigu
 - Unique
 - Canonique
 - Opaque
 - Tableau de 64 bytes
 - Exemple :
 - `urn:jxta:uuid-59616261646162...C8168F0E4BDA903`

27

JXTA

- Credential
 - Utilisé pour identifier un émetteur
 - Fournit :
 - Confidentialité
 - Autorisation
 - Intégrité des données
 - répudiabilité
- Contenu (codat)
 - N'importe quelle donnée (même un processus exécutable)
 - Partagé entre peers et peergroups
 - identifié de manière unique
 - Connu par un Content advertisement

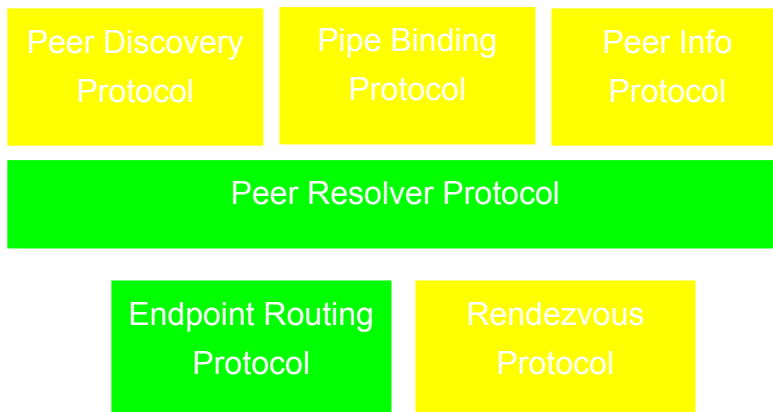
28

JXTA

- Protocoles
 - Peer Resolver Protocol (PRP)
 - Peer Discovery Protocol (PDP)
 - Peer Information Protocol (PIP)
 - Pipe Binding Protocol (PBP)
 - Endpoint Routing Protocol (ERP)
 - Rendezvous Protocol (RVP)
- Indépendant les uns des autres
- On implémente ceux dont on a besoin

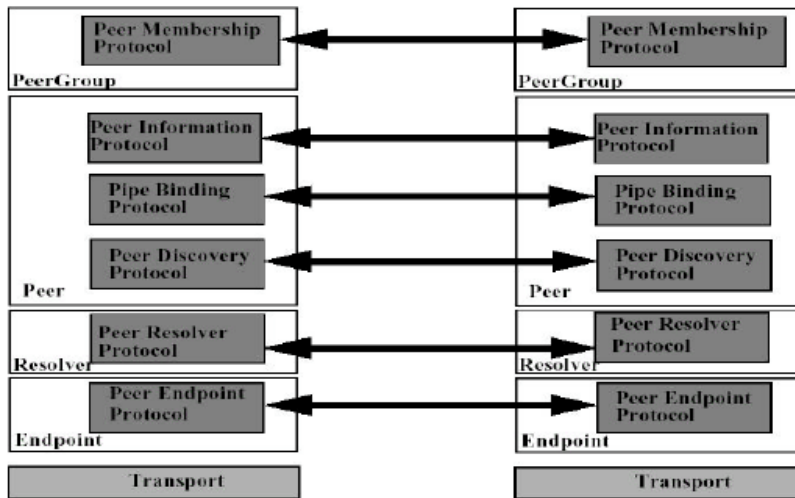
29

JXTA pile des protocoles



30

Protocoles JXTA



31

JXTA

- Peer Resolver Protocol (PRP)
 - Utilisé pour enregistrer des handlers sur des messages spécifiques
 - associe une requête/réponse au handler spécifique
 - chaque requête a un ID unique
 - Ne suppose pas un protocole de transport particulier
 - Utilisé par PDP et PIP

32

JXTA

- Peer Discovery Protocol (PDP)
 - Permet de publier les ressources d'un peer
 - Permet de découvrir n'importe quelle ressource publiée
 - Effectué dans le contexte d'un groupe
 - propagation par IP multicast dans un LAN (implémentation Java)
 - utilise rendezvous/relay dans un WAN (RVP)
 - utilise TTL pour prévenir flooding
 - vérifie id des messages pour éviter la récursivité

33

JXTA

- Peer Information Protocol (PIP)
 - Utilisé pour récupérer de l'information sur un peer
- Pipe Binding Protocol (PBP)
 - Permet d'établir un canal de communication virtuel entre deux ou plusieurs peers
 - Lie les extrémités (input/output(s)) de la connexion
 - Utilise le PRP

34

JXTA

- Endpoint Routing Protocol (ERP)
 - Utilisé pour dynamiquement trouver une route pour envoyer un message à un autre peer
 - Soit il y a une route directe, soit il faut passer par un autre peer
 - Dans un message reçu il y a la route d'envoi
 - Une route peut inclure d'autres routes
 - Utilise le ERP (requêtes envoyées aux autres routeurs)
 - utilise le relay service pour les peers inatteignables (firewall)

35

JXTA

- RendezVous Protocol (RVP)
 - utilisé pour propager un message dans un peergroup
 - propage les requêtes à tous les clients connectés et aux rendezvous
 - utilisé par le Peer Resolver Protocol pour propager les messages

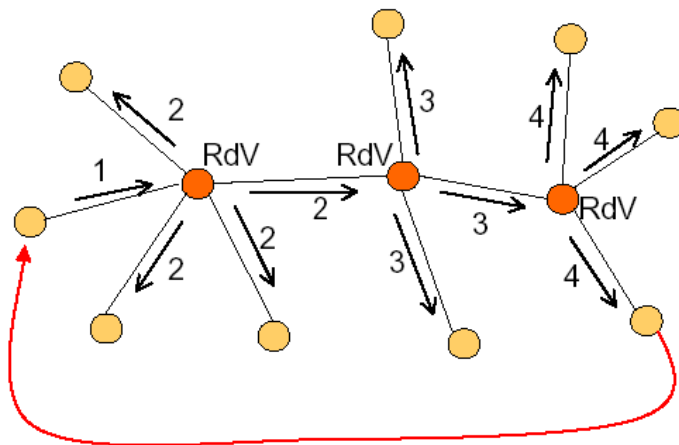
36

Réseau de Rendezvous

- En JXTA 1.0, tous les peers sont impliqués dans la propagation des messages resolveur, discovery, et propagate pipe à l'intérieur d'un PeerGroup
- En JXTA 2.0, la résolution et la propagation sont faites par deux nouveaux concepts:
 - Rendezvous super-peer network et Rendezvous Walker Service
 - Le Shared Resource Distributed Index (SRDI) Service indexe les advertisement sur le rendezvous network

37

Peer Discovery Protocol (V1.0)



38

Nouveau mode de propagation

- Les requêtes se propagent seulement entre rendezvous super-peers
- Les peers reçoivent seulement des requêtes concernant leurs propres advertisements
- Les Rendezvous super-peers s'auto-organisent
- JXTA fournit un framework adaptable

39

Shared Resource Distributed Index

- les peers publient leurs advertisements (de manière répliquée avec un niveau donné) dans le réseau Rdv via des Distributed Hash Tables (DHT)
- Les répliques sont dans le même "voisinage"
- Les DHTs sont maintenues par les Rendezvous peers
- Les requêtes sont envoyées au bon Rdv (ou à une réplique) en utilisant la même fonction de hachage
- Si cela ne marche pas (le réseau de RDV est non cohérent), on cherche de manière itérative
- Les fonctions de hachage sont modifiables

40

DHT faiblement cohérente

- Les peers peuvent apparaître et disparaître de manière dynamique
- Le coût de mise à jour des DHTs peut excéder le gain attendu en lecture
- Une recherche itérative coûte cher en lecture mais rien en mise à jour
- Principe : mixer les deux approches

41

Rendezvous Peer View (RPV)

- chaque rendezvous peer maintient une liste ordonnée (via leur ID) de rendezvous peers
- Il n'y a pas de mécanisme pour garantir la cohérence des RPV entre les différents RVP
- Périodiquement les RVP échange leur RPV avec un nombre aléatoire de RVP dans leur liste (de même un message "alive" avec leur prédécesseur et successeur)

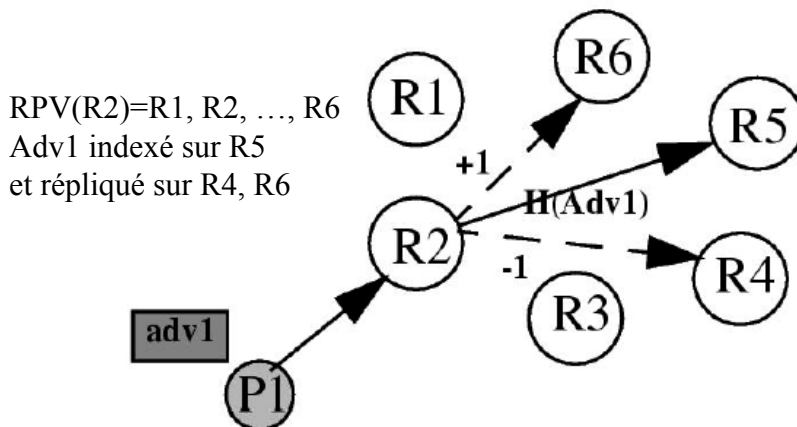
42

Recherche des Rendezvous Peers

- chaque peers maintient une liste de ses rendezvous peers
- Les peers découvrent et stockent les Rdv advertisements
- Recherche dans le peergroup
- Sinon utilise des Seeding Rdvs
- Si rien n'est trouvé, un peer peut s'auto-déclarer RDV

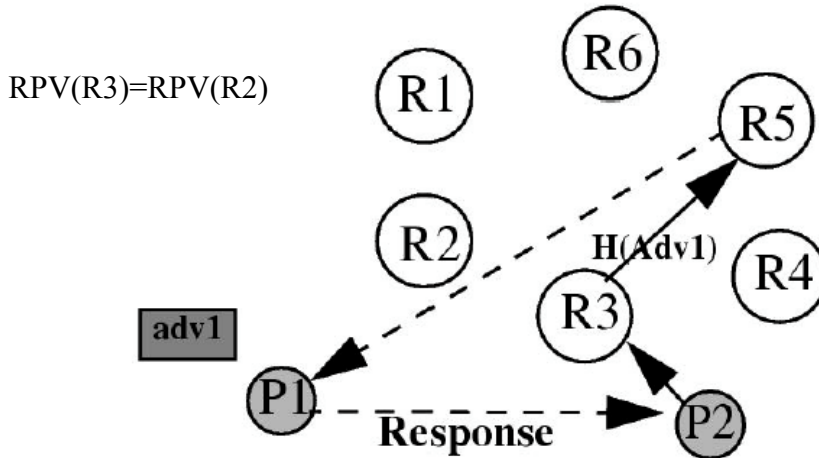
43

Publier un advertisement



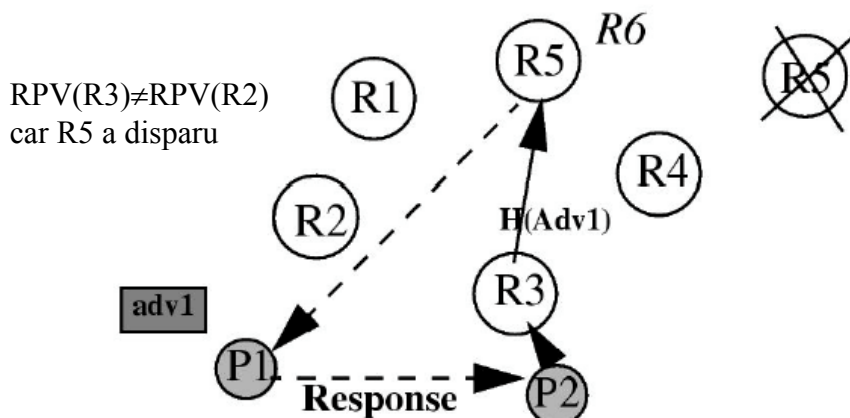
44

Recherche d'un advertisement (cas cohérent)



45

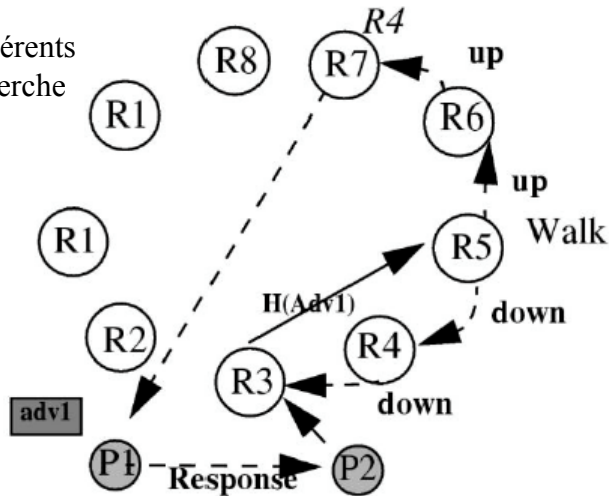
Recherche (cas faiblement cohérent)



46

Recherche (cas incohérent)

RPV sont incohérents
on fait une recherche
séquentielle
bidirectionnelle



47

JXTA

- Service layer
 - Services au dessus du core layer
 - Fournit par qui veut bien
 - pas nécessaire mais intéressant (pour aider le programmeur)
 - 21 services proposés :
 - jxta-rmi
 - jxta-wire (maintenant dans le core layer)
 - ...
 - Créé en utilisant le Module(Spec/Class/Impl)Advertisement

48

JXTA

- Application layer
 - Regroupe les applications enregistrées développées avec JXTA
 - 20 applications bien connues :
 - shell
 - myJxta
 - vop2p
 - ...

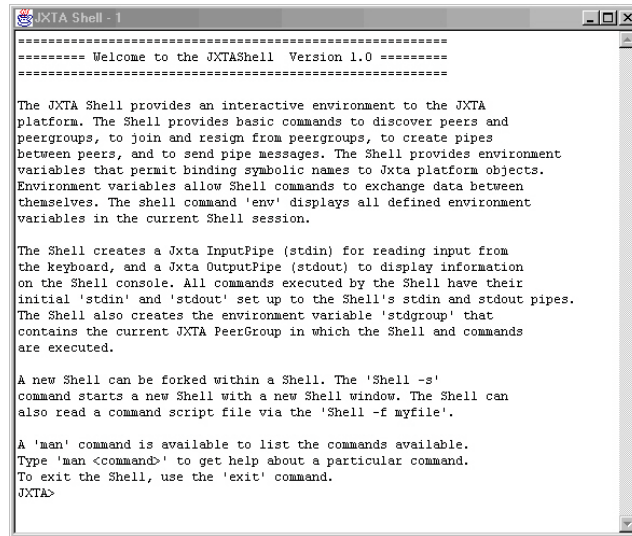
49

JXTA

- Shell
 - Ligne de commande permettant d'interagir directement avec la plateforme JXTA
 - création d'advertisements
 - Parler à d'autres peers
 - ...
 - interface ligne de commande
 - Permet de lancer des scripts
 - Permet d'ajouter dynamiquement de nouvelles commandes
 - Aller voir <http://shell.jxta.org>

50

JXTA



```
JXTA Shell - 1
=====
----- Welcome to the JXTAShell Version 1.0 -----
=====

The JXTA Shell provides an interactive environment to the JXTA
platform. The Shell provides basic commands to discover peers and
peergroups, to join and resign from peergroups, to create pipes
between peers, and to send pipe messages. The Shell provides environment
variables that permit binding symbolic names to Jxta platform objects.
Environment variables allow Shell commands to exchange data between
themselves. The shell command 'env' displays all defined environment
variables in the current Shell session.

The Shell creates a Jxta InputPipe (stdin) for reading input from
the keyboard, and a Jxta OutputPipe (stdout) to display information
on the Shell console. All commands executed by the Shell have their
initial 'stdin' and 'stdout' set up to the Shell's stdin and stdout pipes.
The Shell also creates the environment variable 'stdgroup' that
contains the current JXTA PeerGroup in which the Shell and commands
are executed.

A new Shell can be forked within a Shell. The 'Shell -s'
command starts a new Shell with a new Shell window. The Shell can
also read a command script file via the 'Shell -f myfile'.

A 'man' command is available to list the commands available.
Type 'man <command>' to get help about a particular command.
To exit the Shell, use the 'exit' command.
JXTA>
```

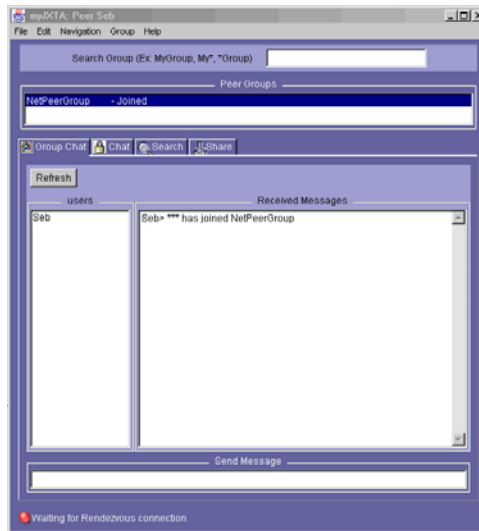
51

JXTA

- myJxta
 - Démonstration des possibilités offertes par JXTA
 - Instant Messaging
 - Partage de fichiers
 - Secure chat
 - Aller voir <http://myjxta.jxta.org>

52

JXTA



53

Let's get started !

- Hello JXTA !
- Create a service
- Discover and use a service

54

Hello JXTA !

```
import net.jxta.peergroup.PeerGroup;
import net.jxta.peergroup.PeerGroupFactory;
import net.jxta.exception.PeerGroupException;
import net.jxta.discovery.DiscoveryService;
import net.jxta.protocol.PeerAdvertisement;
public class SimpleJxtaApp {
    public static void main(String args[]) {
        try { // Create, and Start the default jxta NetPeerGroup
            PeerGroup netPeerGroup = PeerGroupFactory.newNetPeerGroup();
            // Obtain the peer advertisement
            PeerAdvertisement myPeerAdv = netPeerGroup.getPeerAdvertisement();
            //Get the discovery service
            DiscoveryService discovery = netPeerGroup.getDiscoveryService();
            //Publish the peer advertisement
            discovery.remotePublish(myPeerAdv, discovery.PEER,
            discovery.DEFAULT_EXPIRATION);
        } catch (PeerGroupException pge) {
            pge.printStackTrace();
        }
    }
}
```

55

Creating and Publishing a Service

- Create a service advertisement
- Publish the service advertisement
- Resolve a pipe for receiving messages
- Read messages from the pipe

56

Create and publish a module class advertisement

```
try {
    // First create the Module advertisement associated with the service
    ModuleClassAdvertisement mcadv = (ModuleClassAdvertisement)
        AdvertisementFactory.newAdvertisement(
            ModuleClassAdvertisement.getAdvertisementType());

    mcadv.setName("JXTAMOD:JXTA-EX1");
    mcadv.setDescription("JXTA module tutorial");

    ModuleClassID mcID = IDFactory.newModuleClassID();
    mcadv.setModuleClassID(mcID);

    // We now have the Module Class advertisement, let's publish it
    discovery.publish(mcadv,
        discovery.ADV,
        discovery.DEFAULT_LIFETIME,
        discovery.DEFAULT_EXPIRATION);
    // publish in the network
    discovery.remotePublish(mcadv, discovery.ADV,
        discovery.DEFAULT_EXPIRATION);
}
```

57

Create a Module Spec Advertisement

```
ModuleSpecAdvertisement mdadv = (ModuleSpecAdvertisement)
    AdvertisementFactory.newAdvertisement(ModuleSpecAdvertisement.getAdvertisementType());

// provide meta-data describing the service
mdadv.setName("JXTASPEC:JXTA-EX1");
mdadv.setVersion("Version 1.0");
mdadv.setModuleSpecID(IDFactory.newModuleSpecID(mcID));
mdadv.setSpecURI("http://www.jxta.org/Ex1");

PipeAdvertisement pipeadv = null;
try {
    // read in a pre-cooked pipe service advertisement
    FileInputStream is = new FileInputStream("pipeserver.adv");
    pipeadv = (PipeAdvertisement) AdvertisementFactory.newAdvertisement(
        new MimeMediaType("text/xml"), is);
    is.close();
} catch (Exception e) {
    System.out.println("failed to read/parse pipe advertisement");
    e.printStackTrace();
}
```

58

Edit and Publish the Module Spec Advertisement

```
// Include the pipe advertisement within the service advertisement
StructuredTextDocument paramDoc = (StructuredTextDocument)
    StructuredDocumentFactory.newStructuredDocument
        (new MimeMediaType("text/xml"),"Parm");

StructuredDocumentUtils.copyElements(paramDoc, paramDoc, (Element)
    pipeadv.getDocument(new MimeMediaType("text/xml")));

mdadv.setParam((StructuredDocument) paramDoc);

// now that we have the Module advertisement, let's publish
// it in my local cache and into the NetPeerGroup.
discovery.publish(mdadv, discovery.ADV
    discovery.DEFAULT_LIFETIME,
    discovery.DEFAULT_EXPIRATION);
discovery.remotePublish(mdadv, discovery.ADV,
    discovery.DEFAULT_EXPIRATION);

// we are now ready to start the service by creating the input pipe endpoint
// clients can resolve and bind to.
pipe.createInputPipe(pipeadv, this);
```

59

Process an incoming message

```
public void pipeMsgEvent ( PipeMsgEvent event ){

    Message msg = null;
    try {
        msg = event.getMessage();
        if (msg == null) {
            return;
        }

        // get the data element and print it
        String ip = msg.getString("DataTag");
        if (ip != null) {
            System.out.println("Server: received a message: " + ip);
        } else {
            System.out.println("Server: received an empty message");
        }
    }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

60

Discover and Connect to A Service

- Discover Service Advertisements
- Extract Pipe Advertisement from Service Advertisement
- Resolve the Pipe for sending messages
- Send a message to the Pipe

61

Discover and Connect to a Service

```
// lets look in the local cache first
enum = discovery.getLocalAdvertisements(DiscoveryService.ADV
    , "Name"
    , "JXTASPEC:JXTA-EX1");
if ( ! enum.hasMoreElements() ) {

    // We could not find anything in our local cache
    // let's look remotely
    discovery.getRemoteAdvertisements(null
        , DiscoveryService.ADV
        , "Name"
        , "JXTASPEC:JXTA-EX1",1, null);
}
..
// Extract the pipe advertisement
StructuredTextDocument paramDoc =
    (StructuredTextDocument) mdsadv.getParam();
Enumeration elements = paramDoc.getChildren("jxta:PipeAdvertisement");
..
// Bind an output pipe using the advertisement
System.out.println( "attempting to create a OutputPipe" );
pipe.createOutputPipe( pipeAdv, this );
```

62

Discover and Connect to A Service

```
public void outputPipeEvent( OutputPipeEvent event ) {  
    System.out.println( " Got an output pipe event" );  
    //let's grab our output pipe  
    OutputPipe op = event.getOutputStream();  
    Message msg = null;  
  
    //let's create a message and send it on the pipe  
    try {  
        System.out.println( "Sending message" );  
        msg = pipe.createMessage();  
        msg.setString( "DataTag", "Hello JXTA" );  
        // send the message  
        op.send( msg );  
    } catch ( IOException e ) {  
        System.out.println( "failed to send message" );  
        e.printStackTrace();  
        System.exit( -1 );  
    }  
    op.close();  
    System.out.println( "message sent" );  
}
```

63

JXTA

- Caractéristiques du projet JXTA
 - Membres: 16000
 - Licence du type Apache pour contribuer
 - Projets : environ 100
 - Sun team: 25
 - Aller voir www.jxta.org

64

Conclusion

- JXTA
 - Framework pour créer facilement des applications P2P interoperables
 - En constante amélioration
 - La seule (?) base de développement existante aujourd'hui

65

Critiques!

- JXTA est neuf, change rapidement et est faiblement documenté
- On n'a pas besoin de standards dans un monde où il faut faire encore plein de recherche!
- Pourquoi du XML!
- JXTA arrive trop tard, on a déjà KaZaa et les autres!
- puisque JXTA est trop jeune, il est trop tôt pour juger de son succès

66

Le futur de JXTA

- JXTA dans J2SE Micro Edition
- Améliorations de l'implémentation Java
- Implémentation dans d'autres langages (C/C++/C#, Perl, Python, etc)
- Accepté par l'industrie

67

Références

- [BAE02] S. Baehni, P2P, JXTA, Type-based Publish&subscribe, présentation téléchargeable sur la page web de l'auteur à l'EPFL-LPD
- [BBD02] Nitin Borwankar, Daniel Brookshier et al., *JXTA: Java P2P Programming*, Sams publishing, March 2002
- [Jan 03] M. Jan, JXTA Overview, présentation téléchargeable sur le site http://www-src.lip6.fr/projets/datagraal/Reunions/jxta_datagraal.pdf
- [jxta02] JXTA web site, <http://www.jxta.org>
- [Li02] Sing Li, *JXTA Peer-to-Peer Computing with Java*, Wrox, January 2002
- [sJxta03] Specification of JXTA, <http://spec.jxta.org/v2.0/docbook/JXTAProtocols.html>
- [Wi02] Brendon J. Wilson, *JXTA*, New Riders Publishing, May 2002

68