Lezione n.15
15/05/2009
JXTA:
L'applicazione RestoNet
Materiale didattico distribuito a lezione

Laura Ricci

MATERIALE DIDATTICO

- Codice dell'esempio :
  - Schema ad alto livello
  - Codice pubblicato sulla pagina del corso
- Versione di riferimento JAVA 2.4
- Materiale (esempi di applicazioni) sulla pagina https://jxta.dev.java.net/
- Tutorial JXTA sulla pagina web del corso

JXTA: UN ESEMPIO

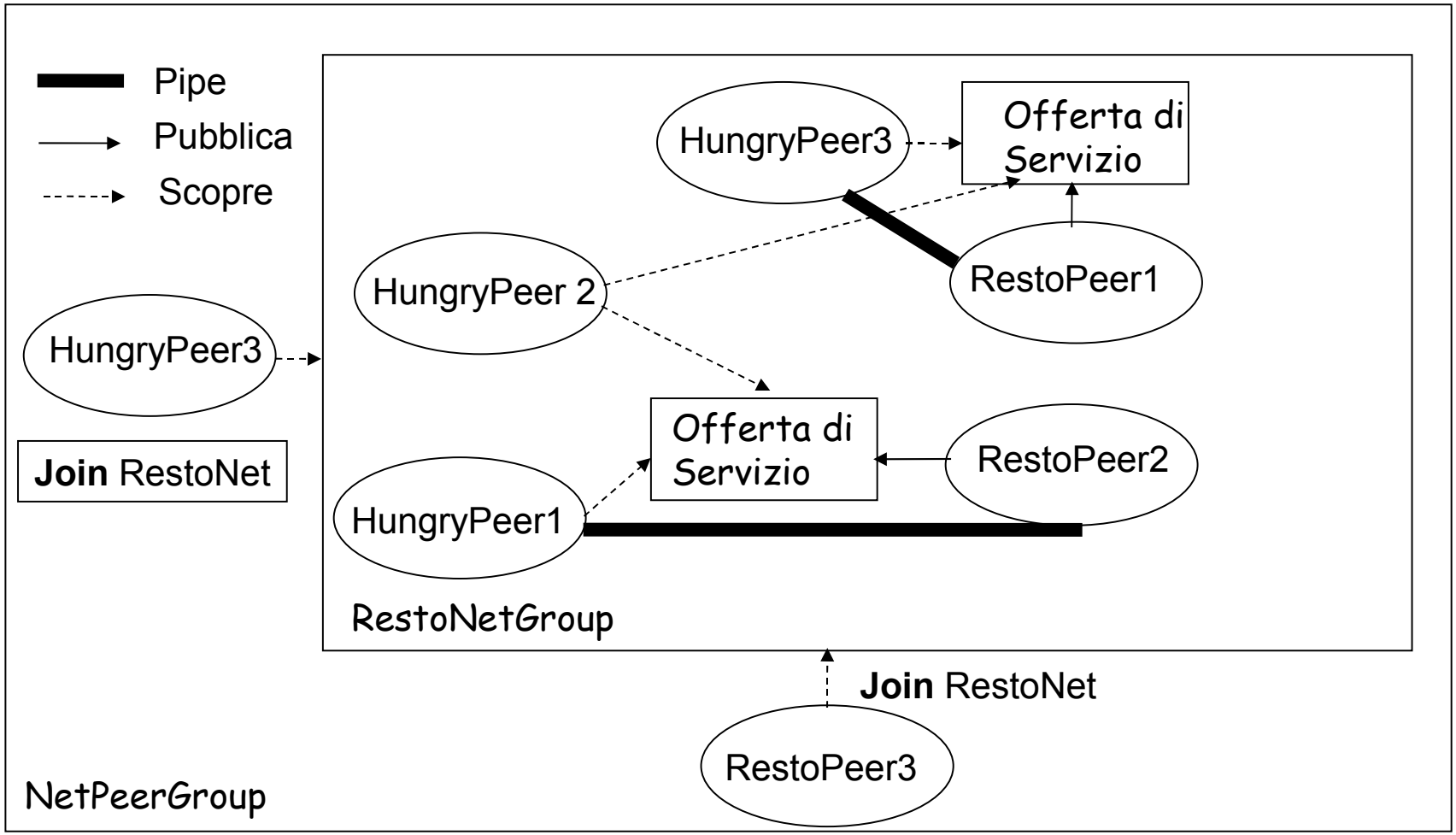
Si vuole progettare un'applicazione P2P per la ricerca distribuita di servizi di ristorazione. Il sistema prevede:

- un peergroup RestoNet a cui si uniscono i ristoratori che presentano alla comunità le proprie proposte di servizio ed i clienti che ricercano le offerte più vantaggiose per loro

- RestoPeer: forniscono servizi di ristorazione. Pubblicano le proprie offerte e rispondono alle richieste di servizio che possono soddisfare.

- HungryPeers: sono alla ricerca di un locale in cui ristorarsi. Ricercano offerte di ristorazione. Possono selezionare il ristorante in base a diversi criteri (prezzo, tipo di cibo, distanza,….)

L'APPLICAZIONE RESTONET



L'APPLICAZIONE RESTONET

Il RestoPeer

- effettua una ricerca per verificare se RestoNetGroup è già stato creato
- se la ricerca ha esito positivo si unisce al gruppo, altrimenti crea un nuovo gruppo, RestoNetGroup
- crea e pubblica una pipe RP da cui riceve le richieste di servizio da parte degli utenti
- Si mette in attesa di richieste e risponde ad esse

L'HungryPeer

- effettua una ricerca per verificare se RestoNetGroup è già stato creato
- se la ricerca ha esito positivo si unisce al gruppo, altrimenti termina
- ricerca annunci di ristorazione all'interno di RestoNetGroup (ricerca le pipes a cui connettersi per comunicare con i ristoranti)
- crea una pipe HP per ricevere le risposte dai ristoranti
- si connette ad un ristorante: collega l'input pipe ad un suo endpoint
- invia la richiesta ad uno dei ristoranti individuati ed attende la risposta su HP
- se la risposta non è soddisfacente si connette ad un altro ristorante

L'APPLICAZIONE RESTONET: L'IMPLEMENTAZIONE

```
try   { PeerGroup netpg = PeerGroupFactory.newNetPeerGroup ( );  }
catch (PeerGroupException e) { System.exit(1); }
DiscoveryService hdisco = netpg.getDiscoveryService ( );
Enumeration ae = null;
int count = 3;
while (count-- >0)
        {try
             {ae = hdisco.getLocalAdvertisements(DiscoveryService.GROUP,
                              "Name","RestoNet");
           if (ae !=null) break;
           hdisco.getRemoteAdvertisements(null, DiscoveryService.GROUP,
                              "Name","RestoNet",1,null)
           try {Thread.sleep(timeout)  } catch (InterrruptedException ie) {   }
           }
          catch (Exception e)  {System.exit(1);}
         }
        if (ae == null)  <creazione del nuovo gruppo>
            else   <richiesta di partecipazione al gruppo>}
```

CREAZIONE DI UN NUOVO GRUPPO

```
PeerGroupID groupID= (PeerGroupID) IDFactory.newPeerGroupID());

PeerGroup restoNet = null;

ModuleImplAdvertisement implAdv =
        netpg.getAllPurposePeerGroupImplAdvertisement(  );

restoNet = netpg.newGroup(groupID,implAdv,"RestoNet","Gruppo Ristoranti");
```

La creazione del nuovo gruppo richiede:

- un identificatore unico per il gruppo
- la definizione dei servizi offerti dal gruppo
- il nome e la descrizione del gruppo

CREAZIONE DI UN NUOVO GRUPPO

- È necessario garantire che tutti i RestoPeer attribuiscano al gruppo lo stesso identificatore

- Se l'applicazione assicura che un solo peer P possa creare il gruppo,
  - P può impostare groupID a null. JXTA genera automaticamente un nuovo identificatore per il gruppo
  - P può invocare la funzione IDFactory.newPeerGroupID( ) che genera un identificatore unico per il gruppo

- IDFactory contiene metodi per creare identificatori unici per gruppi,peers pipes, moduli

- Se più peers possono creare in parallelo lo stesso gruppo è necessario che l'identificatore attribuita al gruppo sia consistente, altrimenti posso creare advertisements diversi per lo stesso gruppo

CREAZIONE DI UN NUOVO GRUPPO

Creazione dell'identificatore unico del gruppo:

- Quando un nuovo peer P ricerca un advertisement per RestoNet e non riesce a reperirlo
  - È possibile che esistano comunque dei peer che appartenengono a RestoNet
  - in questo caso, la ricerca può aver dato esito negativo perché P non ha atteso la risposta per un intervallo di tempo sufficiente, oppure problemi di routing non hanno consentito di raggiungere quel peer

- in questo caso è possibile che P generi un advertisement per un gruppo che è già stato creato in precedenza

- P deve utilizzare lo stesso identificatore generato per il gruppo al momento della sua creazione

CREAZIONE DI UN NUOVO GRUPPO

- L'identificatore del gruppo è costruito a partire da una stringa predefinita "cablata" nel codice. Poiché tutti i RestoPeers eseguono lo stesso codice, tutti attribuiscono lo stesso identificatore al gruppo

- L'identificatore unico può essere ottenuto mediante metodi 'off line' rispetto all'applicazione

- Esempio: un applicativo genera un insieme di identificatori unici, mediante il metodo newPeerGroupID( ) e li scrive all'interno di un file

- Ogni RestoPeer legge dal file l'identificatore e, se deve creare il gruppo RestoPeer utilizza quell'identificatore

CREAZIONE DI UN NUOVO GRUPPO

- JXTA permette di associare al nuovo gruppo un insieme di servizi, descritti mediante un insieme di module advertisements
- i servizi fanno riferimento a implementazioni (codice) disponibili a tutti i membri del gruppo
- i membri del gruppo possono scaricare l'implementazione di un servizio ed utilizzarlo (invocarne le funzioni)
- È possibile associare al nuovo gruppo solo i servizi base offerti da JXTA (peer discovery service, pipe binding service….)
- La funzione getAllPurposePeerGroupImplAdvertisement( ) crea un advertisement standard che contiene solo i riferimenti ai servizi base

RICHIESTA DI PARTECIPAZIONE AD UN GRUPPO

Se il gruppo RestoNet esisteva ed il peer è riuscito a reperire un advertisement che lo definisce

- se il gruppo implementa politiche di sicurezza, il peer che intende partecipare al gruppo deve essere autenticato
- altrimenti è sufficiente istanziare un nuovo gruppo a partire dall'advertisement reperito
  - (PeerGroupAdvertisement) restoNetAdv =
    (PeerGroupAdvertisement) ae.nextElement( );
  - restoNet = netpg.newGroup(restoNetAdv)

JXTA: IL MECCANISMO DELLE PIPES

- Le pipes consentono agli HungryPeers ed ai RestoPeers di comunicare all 'interno del gruppo RestoNet

- Pipe Unidirezionali: occorre creare
  - una pipe tramite cui l'HungryPeer invia messaggi al RestoPeer
  - una pipe tramite cui il RestoPeer invia le risposte all'HungryPeer

- Ogni RestoPeer
  - crea e pubblica un pipe advertisement
  - collega l'input pipe ad un suo endpoint

- Gli HungryPeers
  - ricercano in RestoNet le pipes create dai RestoPeers
  - utilizzano le pipes reperite per comunicare le loro richieste ai RestoPeers

PIPE ADVERTISEMENTS

- Per evitare di pubblicare una pipe con un nome già esistente, si effettua una ricerca di quella pipe, in base al suo nome
- L'assenza di conflitti non può essere comunque garantita con certezza
- Se due peers distinti scegliessero lo stesso nome per due pipes diverse un restopeer potrebbe ricevere messaggi destinati ad un altro restopeer.
- Per effettuare la ricerca del pipe advertisement
  - si utilizza il DiscoveryService associato a RestoNet.
  - la propagazione della query è limitata ai peers appartenenti a RestoNet
  - meccanismo di scoping fornito dai gruppi

PIPE ADVERTISEMENTS

Il RestoPeer controlla se questa pipe è già stata pubblicata

```
DiscoveryService      disco= restoNet.getDiscoveryService ( );
Name = "Il Gambero Rosso";
Enumeration ae = null;
 int count = 3;
 while (count -- > 0) {
    try
     { ae = disco.getLocalAdvertisements( DiscoveryService.ADV, "name",
                                          "RestoNet:RestoPipe:"+ Name);
     if ((ae != null) && ae.hasMoreElements( )) break;
     disco.getRemoteAdvertisements( null, DiscoveryService.ADV,"name"
                                    "RestoNet:RestoPipe:"+ Name,1,null);
     try { Thread.sleep (time-out);} catch (InterruptedExeception e) { }
catch(IOException e) { }
}
```

CREAZIONE DEI PIPE ADVERTISEMENTS

- Se il pipe advertisement non viene individuato, ne viene creato uno nuovo e viene pubblicato nella cache locale e propagato nella rete

```
PipeAdvertisement myAdv
    = (PipeAdvertisement) AdvertisementFactory.newAdvertisement
                (PipeAdvertisment.getAdvertisementType( ));
myAdv.setPipeID(.....);
myAdv.setName("REstoNet:RestoPipe:"+Name);
myAdv.setType(PipeService.UnicastType)
disco.publish(myAdv, DiscoveryService.ADV, PeerGroup.DEFAULT_LIFETIME,
                PeerGroup.DEFAULT_EXPIRATION)
disco.remotePublish(myAdv, DiscoveryService.ADV,
                PeerGroup.DEFAULT_EXPIRATION)
```

CREAZIONE DEI PIPE ADVERTISEMENTS

- **public static** Advertisement newAdvertisement(**String** advertisementType)
  - costruisce un'istanza di un advertisement che possiede il tipo specificato come parametro (nel nostro caso un PipeAdvertisement)
- SetPipeID, setName, setType sono metodi della classe PipeAdvertisement che attribuiscono valori ai diversi campi dell'advertisement
- Generazione di un identificatore unico per la pipe con i metodi visti in precedenza

PDP: PEER DISCOVERY SERVICE

- Permette ad un peer di
  - specificare e pubblicare i tipi di servizio che può offrire
  - ricercare (Discovery) servizi all'interno della rete JXTA
- Esempio: pubblicazione di un servizio di piping mediante il quale scambiare i dati con gli altri peers
- Ogni servizio viene descritto mediante un advertisement
- Le query vengono automaticamente propagate sulla rete dal supporto
- L'insieme di risposte ad ogni query vengono salvate nella cache locale di un peer, per consentire una successiva ricerca più efficiente dello stesso servizio

PDP: PUBBLICAZIONE DEGLI ADVERTISEMENTS

- ogni advertisement viene indicizzato utilizzando il servizio SRDI (Shared Resource Distributed Indexes) usando un numero predefinito di chiavi (es: identificatore, nome, dell'advertisement)

- ogni peer P può pubblicare un advertisement A
  - nella sua cache locale
  - mediante invio a tutta la rete

- Pubblicazione locale

```
public void publish (Advertisement adv) throws IOException

public void publish (Advertisement adv, long lifetime,
                     long expiration) throws IOException
```

PDP: PUBBLICAZIONE LOCALE

Il peer P pubblica localmente l'advertisement A:

- A viene memorizzato nella cache locale
- viene creato una chiave che viene memorizzata mediante il servizio SRDI, in modo che l'advertisement possa essere reperito dagli altri peers
- **long** lifetime tempo di vita (millisecondi), dell'advertisement. Dopo questo tempo l'advertisement viene eliminato dalla cache locale
- **long** expiration intervallo di tempo (millisecondi) in cui l'advertisement rimane valido nelle caches degli altri peers
- se i valori di lifetime e di expiration non vengono indicati, si considerano i valori definiti per default (definiti nell'interfaccia DiscoveryService, esempio 1 anno e due ore)

PDP: PUBBLICAZIONE REMOTA

Pubblicazione Remota

```
public void remotePublish (Advertisement adv) throws IOException

public void remotePublish (Advertisement adv, long expiration) throws
                                                    IOException
```

- il peer P pubblica in remoto l'advertisement A:
  - A viene propagato sulla rete e memorizzato dai peers del gruppo a cui appartiene P che lo ricevono
  - si può specificare un valore per l'expiration time (tempo massimo di validità dell'advertisement nelle caches degli altri peers).
  - si può specificare l'identificatore di un peer su cui si vuole pubblicare A

CREAZIONE DELLA PIPE

- La creazione della input pipe prende in input l'advertisement della pipe myAdv
- myAdv è stato creato precedentemente oppure è stato ritrovato mediante il processo di Discovery
- Creazione della input pipe
  - PipeService pipes = restoNet.getPipeService ( );
  - PipeIn = pipes.createInputPipe (myAdv)
- Gli hungry Peers ricercano advertisement di pipes creati dai RestoPeer
  - la chiave di ricerca è il nome della pipe
  - gli hungry peers creano l'output pipe e la collegano ad un proprio endpoint
  - il meccanismo delle pipe consente agli hungryPeers di invocare un servizio offerto dai RestoPeers

RICEZIONE DI MESSAGGI DALLA PIPE

- I messaggi inviato dall'HungryPeer al RestoPeer deve contenere
  - la richiesta, ad esempio il tipo servizio richiesto (esempio: l'hungry peer richiede patatine fritte, invia il suo nome e la quantità di patatine richieste)
  - l'advertisement della pipe che dovrà essere utilizzata dal RestoPeer per inviare il messaggio di risposta all'HungryPeer. In questo modo il RestoPeer può rispondere immediatamente, senza dover ricercare la pipe dell'Hungry Peer nella rete
- Messaggio = lista ordinata di uno o più message elements
- Ogni message element può essere a sua volta un documento strutturato

RICEZIONE DI MESSAGGI DALLA PIPE

Message msg = pipeIn.waitForMessage( );

- Il messaggio è un documento XML
- E' necessario individuare i diversi campi del messaggio mediante i nomi degli attributi
- Un campo del messaggio conterrà l'advertisement (hungrypipe) della pipe utilizzata per rispondere al cliente
- Un altro campo conterrà la descrizione della richiesta del cliente

RICEZIONI DI MESSAGGI DA UNA INPUT PIPE

Per ricevere un messaggio da una input pipe

- Ricezione sincrona dei messaggi
- Message waitForMessage( ) **throws** InterruptedException
  - blocca l'esecuzione del peer fino al momento in cui viene ricevuto un messaggio. Restituisce il messaggio ricevuto
  - l'attesa bloccante può essere interrotta da un altro thread

- Ricezione asincrona dei messaggi
  - si associa un PipeMsgListener alla pipe, al momento della sua creazione
  - si implementa il metodo pipeMsgEvent del listener, che definisce il comportamento del peer al momento della ricezione dei messaggi

INVIO DELLA RISPOSTA

Il RestoPeer, per inviare la risposta all'HungryPeer

- Utilizza l'advertisement della pipe utilizzata da HungryPeer per ricevere le risposte per creare un' outputpipe

```
pipeOut = pipes.createOutputPipe(hungryPipe, timeout)
```

- Crea il messaggio di risposta msg (può utilizzare la classe Structure Document)

- Invia il messaggio di risposta (prezzo delle patatine, offerte speciali,…)sulla Output Pipe

```
pipeOut.send(msg)
```

HUNGRYPEER: IMPLEMENTAZIONE

- ricerca un advertisement per il gruppo RestoNet
- se non lo trova, termina, altrimenti ricerca un certo servizio presso un qualsisi ristorante del gruppo RestoNet
- l'accesso ai servizi di ristorazione avviene per mezzo del meccanismo delle pipe
- ricercare un servizio equivale a ricercare una pipe che permetta di comunicare con quel servizio.
- individuata in RestoNet l'advertisement di una pipe pubblicato da un RestoPeer, collega l'output pipe ad un proprio endpoint ed invia la richiesta
- attende la proposta di servizio e, nel caso risulti insoddisfacente, contatta un altro RestoPeer

HUNGRYPEER: IMPLEMENTAZIONE

- Ricerca del servizio di ristorazione:

  Enumeration ae=
  disco.getLocalAdvertisements
  (DiscoveryService.ADV, "name", "RestoNet:RestoPipe:*")

  disco.getRemoteAdvertisements
  (null, DiscoveryService.ADV, "name", "RestoNet:RestoPipe:*", 5, null)

- si utilizza una wild-card per ricercare una qualsiasi pipe di tipo RestoPipe, pubblicata all'interno del gruppo RestoNet

PDP: RICERCA DI SERVIZI REMOTI

int getRemoteAdvertisements (**String** peerid, **int** type, **String** attribute, **String**value, **int** threshold)

ricerca gli advertisement pubblicati dai peer appartenenti al peergroup e li memorizza nella cache locale del peer

peerid: indica l'identificatore del peer su cui si vuole effettuare la ricerca.

- peerid=null, la query viene propagata a tutti i peer del peergroup
- propagazione: mediante multicast sulla sottorete locale, memdiante rendez vouz peer per la propagazione ai peer remoti

type tipo dell'advertisement

attribute + value: consentono di restringere la ricerca.

threshold: indica il numero massimo di advertisements che possono essere restituiti.

PDP: RICERCA DI SERVIZI REMOTI

int getRemoteAdvertisements (**String** peerid, **int** type, **String** attribute, **String** value, **int** threshold, **DiscoveryListener** listener)

DiscoveryListener: permette la ricerca asincrona dei servizi remoti.

Ricerca asincona:

- creazione di un oggetto che implementa l'interfaccia DiscoveryListener
- l'interfaccia definisce al suo interno un metodo, discoveryEvent, che
  - viene automaticamente invocato al momento della ricezione di un advertisement
  - deve essere implementato per definire le operazioni da eseguire al momento della ricezione di un advertisement
  - l'implementazione del metodo può accedere ad un oggetto DiscoveryEvent, che riferisce un DiscoveryResponseMsg che contiene i dati relativi all'advertisement individuato
- Associazione del DiscoveryListener al servizio di Discovery

PDP: RICERCA ASINCRONA DI SERVIZI REMOTI

Esempio: l'HungryPeer effettua una ricerca asincrona dei servizi (pipes) in RestoNet

```
public class HungryPeer implements DiscoveryListener
    ……
    disco.addDiscoveryListener (this);
    disco.getRemoteAdvertisements (…….)
    ……
    public void discoveryEvent(DiscoveryEvent e) {
            DiscoveryResponseMsg msg = e.getResponse();
            Enumeration e = msg.getResponses( );
            while (e.hasMoreElements( ) ){
                <memorizza l'advertisement reperito>}
```

PDP:RICEZIONE ASINCRONA DEI MESSAGGI

```
PipeMsgListener  pipeListener = new PipeListener ( )
            { public void pipeMsgEvent (PipeMsgEvent e)
                    try
                            { Message msg = e.getMessage ( ) ;
                            PipeID pipeID = e.getPipeId( );
                            System.out.println("Message"+
                            msg.getString("DataTag")+"pipe"+ (String)
                                                        pipeID;)
                            }
                    catch (Exception e)
            }

……
createInputPipe(pipeAdv, pipeListener)
……
```

JXTA CORE SERVICES E RISPETTIVI PROTOCOLLI

Discovery Service: pubblicazione e la ricerca (Discovery) di servizi e risorse all'inteno della rete	**Peer Discovery Protocol (PDB)**
Pipe Service: creazione e pubblicazione di pipes per lo scambio diretto di informazioni tra i peers	**Peer Binding Protocol (PBP)**
Peer Information Service: permette di monitorare lo stato degli altri peers partecipanti al gruppo	**Peer Information Protocol (PIP)**
Rendez Vouz Service: permette di contattare un rendez-vouz peer per la propagazione delle queries	**Rendez Vouz Protocol (RVP)**
End Point Service: gestisce l'invio dei messaggia basso livello	**END Routing Protocol (ERP)**
Peer Resolver Service: permette di inviare queries generiche a peers singoli a o tutti i peers di un gruppo	**Peer Resolve Protocol (PRP)**
MemberShip e Access Control: implementa meccanismi per il controllo dell'accesso di un peer ai peergroups	

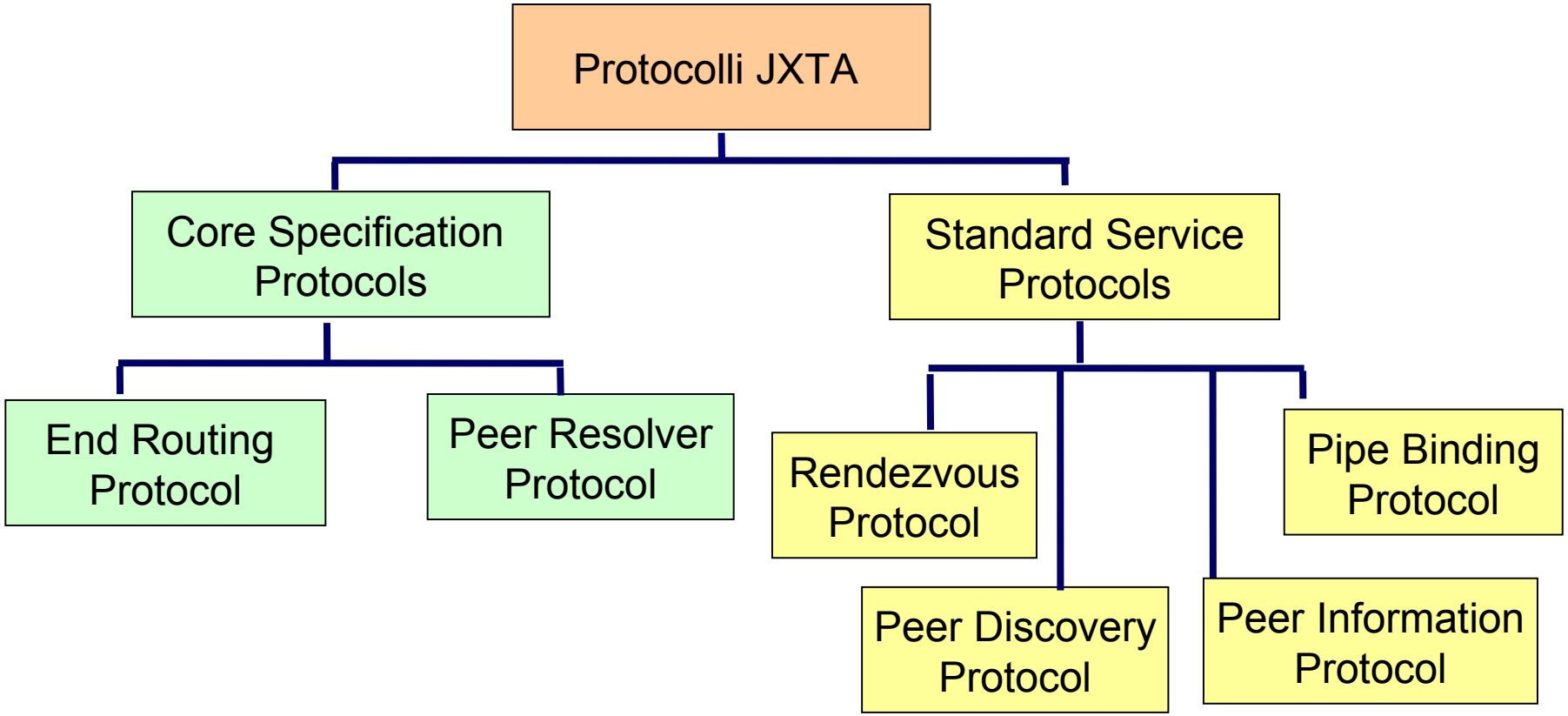
JXTA CORE SERVICES

- Alcuni servizi sono automaticamente associati al gruppo creato

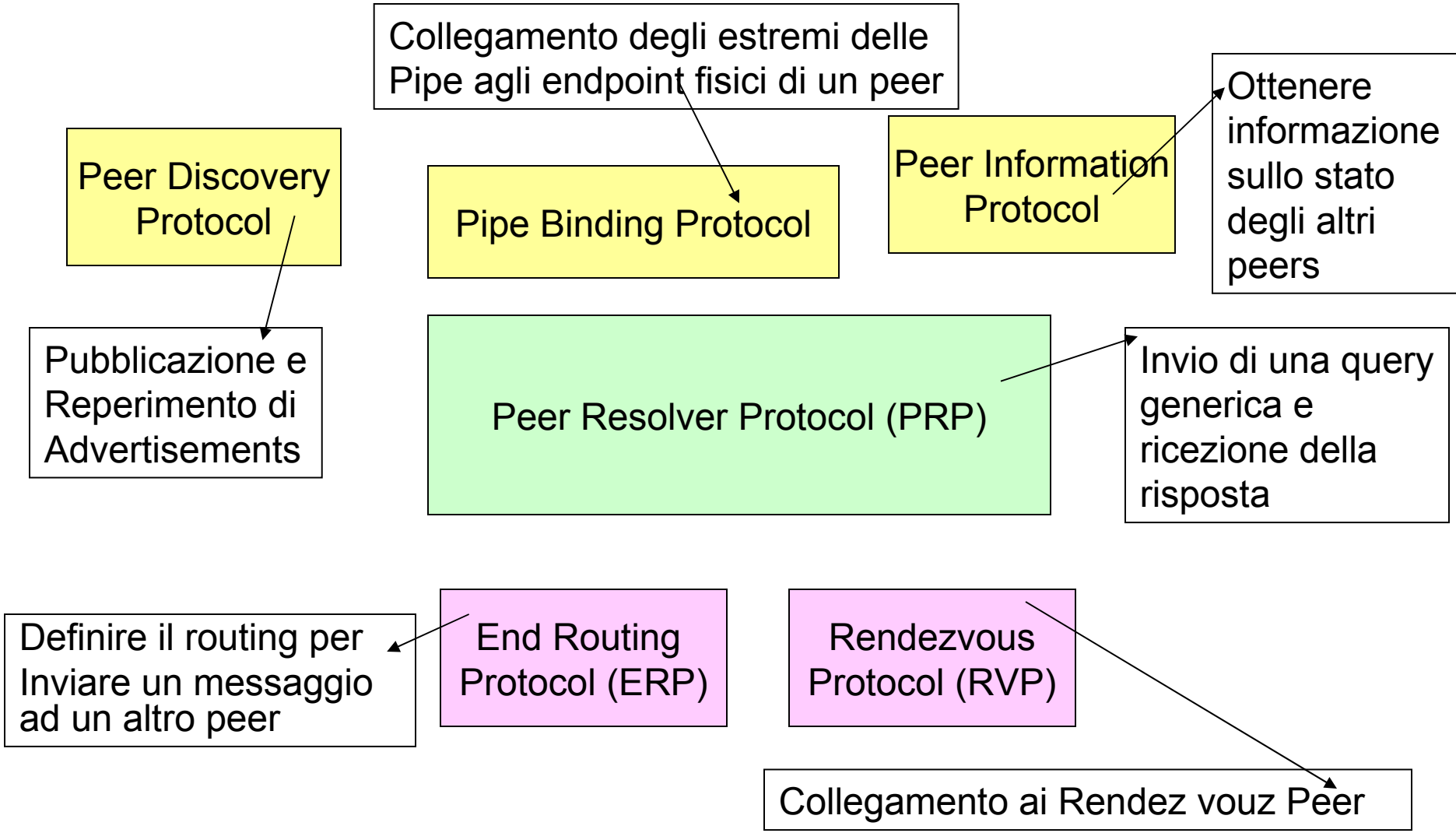
- I riferimenti a questi servizi possono essere reperiti mediante invocazioni a metodi della classe PeerGroup

```
DiscoveryService     discos= restoNet.getDiscoveryService ( );
PipeService          pipes  = restoNet.getPipeService ( );
RendezVouzService    rends  = restoNet.getRendezVouzService( );
PeerInfoService      pinfos = restoNet.getPeerInfoService( );
MembershipService    membs= restoNet.getMembershipService ( );
ResolverService      resvs  = restiNet.getResolverService( );
EndPointService      endps  = restoNet.getEndPointService ( );
```

JXTA: I PROTOCOLLI



JXTA: I PROTOCOLLI



IL PROTOCOLLO PRP:PEER RESOLVER PROTOCOL

- Il Peer Resolver Protocol è uno di protocolli più a basso livello di JXTA
- Un Resolver Service viene associato automaticamente ad ogni gruppo creato da JXTA, come servizio di default
- Consente di inviare una generica query agli altri peer della rete
- Definisce la struttura dei messaggi di query e di query response inviati sulla rete
- Le query sono definite mediante XML
- Viene utilizzato sia del PDP, (Peer Discovery Protocol) che dal PIP (Peer Information Protocol)
- Ogni discovery query inviata dal PDP viene trasformata in una Resolver Query che incapsula la query effettuata a livello di Discovery Service
- Esempio: una getRemoteAdvertisement viene tradotto in una ResolverQuery e viene inviata sulla rete

PEER RESOLVER PROTOCOL

- Il Peer Resolver Protocol offre l'interfaccia QueryHandler
- Interfaccia QueryHandler definisce i metodi
  - processQuery (ResolverQueryMsg query) definisce le operazioni che devono essere effettuate quando si riceve una query
  - processResponse(ResolverResponseMsg response) definisce le operazioni che devono essere effettuate quando si riceve la risposta ad una query
- I peer devono
  - Ottenere un riferimento al ResolverService associato al gruppo su cui vuole inviare la query
  - definire una classe QI che implementi l'interfaccia QueryHandler e dare un'implementazione dei metodi processQuery e processResponse
  - Registrare la classe QI presso il Resolver Service
  - Inviare e ricevere query

PEER RESOLVER PROTOCOL: UN ESEMPIO

Un esempio:

- il peer1 invia (metodo sendQuery) al peer2 una query (ProductQuery) per eseguire il prodotto tra due numeri
- Il peer2 risponde a questa query calcolando il prodotto richiesto. Questo calcolo viene eseguito nel metodo processQuery che implementa il metodo corrispondente dell'interfaccia QueryHandler
- Il peer1 riceve la risposta la elabora (ad esempio, stampa il risultato) mediante il metodo processResponse dell'interfaccia QueryHandler

PEER RESOLVER PROTOCOL: UN ESEMPIO

Il formato dei messaggi:

/*** Messaggio ProductQuery

<product: ProductQuery xmlns:product="hhtp://jxta.org">
<firstNumber> </firstNumber>
<secondNumber> </secondNumber>
</product: ProductQuery>

/****Messaggio ProductResponse

<product: ProductResponse xmlns:product="http://jxta.org">
<result> </result>
</product:ProductResponse>

PEER RESOLVER PROTOCOL: UN ESEMPIO

- Per registrare l'handler al Resolver Service

```
ResolverService resolver = netPeerGroup.getResolverService();
ProductHandler = new ProductHandler();
resolver.registerHandler("ProductHandler",handler);
```

- Per creare e spedire una query
  - creare il documento XML
  - creare una ResolverQuery
  - spedire la query mediante il ResolverService

PEER RESOLVER PROTOCOL: UN ESEMPIO

Invio delle queries:

```
.....

Element e = null;
e = doc.createElement("firstNumber", 6);
doc.appendChild(e);
e = doc.createElement("secondNumber", 5);
doc.appendChild(e);
String xml = serializeDoc(doc);
ResolverQuery message = null;
message = new ResolverQuery( ):
message.setQuey(xml);..... message.setHandlerName(ProductHandler),.....
resolver.sendQuery(null, message);

.....
```

IL PROTOCOLLO ERP: ENDPOINT ROUTING PROTOCOL

- Si occupa del routing dei messaggi
- Consente ad un peer di trovare il cammino verso un qualsiasi altro peer
- Necessario per la comunicazione tra peers che non possono essere collegati direttamente a causa di
  - Firewalls
  - NATs
  - Utilizzo di diversi protocolli di rete
- Ogni peer conserva nella propria cache le informazioni relative al routing ricavate mediante l'ERP

IL PROTOCOLLO ERP: ENDPOINT ROUTING PROTOCOL

- Quando un peer P deve comunicare con peer Q, P ricerca nella propria cache le informazioni sul cammino da percorrere per raggiungere Q
- Se non possiede queste informazioni interroga l'ERP, mediante una RouteQuery inviata al rendezvous peer connesso a P
- Il rendez vous peer controlla a sua volta nella propria cache e controlla se la query può essere risolta con le informazioni possedute
- In caso negativo, il rendezvous peer inoltra la richiesta ad altri rendesvous peers
- La route verso Q viene quindi memorizzata nella cache locale di P

IL PROTOCOLLO ERP: ENDPOINT ROUTING PROTOCOL

- I rendez vous peers sono peer speciali in grado di svolgere funzioni di routing
- Possono essere interpellati dagli edge peers al fine di trovare una route valida per mettere in comunicazione due peers
- Ogni rendez vous peer, in base alle proprie conoscenze ed in base all'interrogazione di altri rendezvous noti, restituisce al richiedente una sequenza di intermediari (che possono essere sia edge peers che rendezvous peers) per stabilire la connessione desiderata
- L'invio vero e proprio del messaggio è gestito dai protocolli di trasporto implementati dall'architettura(TCP, HTTP, TLS, Beep, ServerHtttp, Bluetooth)

IL PROTOCOLLO RVP: RENDEZ VOUS PROTOCOL

- Ogni peer, nella propria cache locale, memorizza anche gli advertisements che descrivono i rendezvous peers
- Questi advertisements vengono suddivisi in lotti (di 5 advertisements) ed il rendesvouz tenta di contattare sequenzilamente questi rendesvouz
- Se dopo un certo intervallo di tempo non è stata stabilita alcuna connessione, si passa al lotto successivo
- Prima si contattano i rendezvous in rete locale
- Poi i seeding rendezvous (rendezvous ritenuti affidabili ed il cui indirizzo è fornito con l'applicazione
- Se ancora non si ha risposta il peer stesso tenta di diventare un rendez vous

RENDEZVOUS SUPERPEERS

- Classificazione dei peers
  - Edge Peers
  - Rendez-Vouz Peers
  - Relay Peers

- Rendezvous Peers: peers che si sono resi disponibili ad effettuare

l 'instradamento degli advertisements sulla rete

- Un prerequisito per diventare rendezvous peer per un gruppo è essere membro di quel gruppo

•Al momento del bootstrap un rendez vous peer deve inizializzare la propria RPV (Rendezvous Peer View)

RPV= conoscenza che un rendezvous ha degli altri rendezvous dello stesso gruppo

SEEDING RENDEZVOUS

- Seeding Rendezvous
  - Rendez vous che sono stabilmente attivi sulla rete
  - Un rendezvous peer è configurato mediante una lista predefinita di seeding rendezvous
  - insieme di rendezvous che permettono il bootstrap di un rendezvous peer sulla rete

- I seeding rendez vous agiscono da rendezvous solo per NetPeerGroup

- All'inizio il rendezvous conosce solamante un insieme di seeding rendezvous

- dopo il bootstrap, i seeding rendezvous vengono utilizzati come punto di distribuzione di advertisements di rendezvous che si sono connessi alla rete

RENDEZVOUS SUPERPEERS

- E' compito dell'utente definire i rendezvous peer per gruppi diversi dal NetPeerGroup

- Come un peer P può diventare rendezvous peer:

  - Invocazione esplicita al metodo startRendezvous( )
  - Con invocazione del metodo setAutoStart(true, period) P dichiara la sua disponibilità a diventare un rendezvous.
  - A seconda del rapporto edge/rendezvous peer presenti nella rete in quel momento, P può assumere o meno la funzione di rendez vous peer
  - Lo stato della rete viene ricontrollato ogni period millisecondi. JXTA quindi decide dinamicamente se 'promuovere' un peer da edge a rendez vous o farlo 'regredire' da rendezvous ad edge peer

RENDEZVOUS SUPERPEERS

- un peer che è diventato un rendez vous mediante il meccanismo di autostart( ) può ritornare automaticamente alla condizione di edge peer
- di solito JXTA mantiene al massimo 5 rendezvous per ogni gruppo, anche nel caso in cui tutti i partecipanti al gruppo siano rendezvous
- se esistono più di 5 rendezvous nella rendezvous peer view del peer (la visione locale che un peer ha della rete di rendes vouz), il peer può decidere di tornare allo stato di edge peer. La decisione è influenzata anche dal numero di edge peers connessi
- i peers serviti da quel rendezvous devono trovare un altro rendezvous a cui connettersi
- per connettersi ad altri rendez vous, i peers utilizzano i rendezvous advertisements reperiti mediante il rendezvous a cui erano precedentemente connessi

RENDEZVOUS SUPERPEERS

- Quando un peer diventa Rendez-Vous, pubblica un proprio advertisement, al cui interno sono contenute le informazioni necessarie agli altri peer al fine di stabilire una connessione con esso.
- Formato di un advertisement

```
<jxta:RdvAdvertisement xmlns:jxta="http://jxta.org">
    <RdvGroupId> . . . </RdvGroupId>
    <RdvPeerId> . . . </RdvPeerId>
    <Name> . . . </Name>
    <RdvServiceName> . . . </RdvServiceName>
    <RdvRoute> . . . </RdvRoute>
</jxta:RdvAdvertisement>
```

- Un edge peer che desidera collegarsi ad un rendezvous peer, effettua una ricerca di questi advertisements
- La ricerca avviene a partire dai rendezvous peer già esistenti (all'inizio a partire dai seeding rendezvous)

CONNESSIONE EDGE/RENDEZVOUS PEER

- un edge peer è connesso in ogni istante ad un unico rendesvouz peer
- un edge peer ricerca continuamente (in background) ulteriori rendes vouz peer advertisements e li memorizza nella propria cache locale
- i rendezvous advertisment sono mantenuti nella cache locale del peer al momento della sua disconnessione dalla rete e possono quindi essere ritrovati in esecuzioni successive
- Un edge peer, per connettersi ad un rendezvous peer esegue, nell'ordine, i seguenti passi
  - ricerca nella propria cache locale
  - invia richieste di advertisement sulla rete
  - prova a connettersi ai seeding rendezvous
  - diventa esso stesso un rendezvous peer

CONNESSIONE EDGE/RENDEZVOUS

- La ricerca degli advertisements dei rendezvous peer avviene mediante il discovery service

- Per connettersi ad un rendezvous
  - connectToRendezVous (PeerAdvertisement adv) il parametro indica l'advertisement di un rendez vous scoperto mediante il Discovery Service
  - connectToRendezVous (EndpointAddress addr). Il parametro indica l'indirizzo del rendez vous

- L'invocazione di questi metodi genera un lease request message.

- Lease request message: viene spedito da un peer ad un rendez-vous per chiedere il permesso a collegarsi ad esso, di poterlo usare per propagare i propri messaggi e di essere inseriti nella lista dei peer in grado di ricevere messaggi propagati da altri.

GESTIONE CONNESSIONI AI RENDEZVOUS

- Per disconnettersi da un rendez vous

  void disconnectFromRendezVous(ID peerID)

- L'invocazione del metodo genera un lease cancel message
- Per la gestione dei messaggi di connessione/disconnessione è possibile utilizzare un listener, il RendezvousListener.
- Il RendezvousListener definisce un unico metodo

  rendezvousEvent(RendezvousEvent event)

che deve essere implementato e poi registrato presso il rendez-vous service mediante il metodo addListener

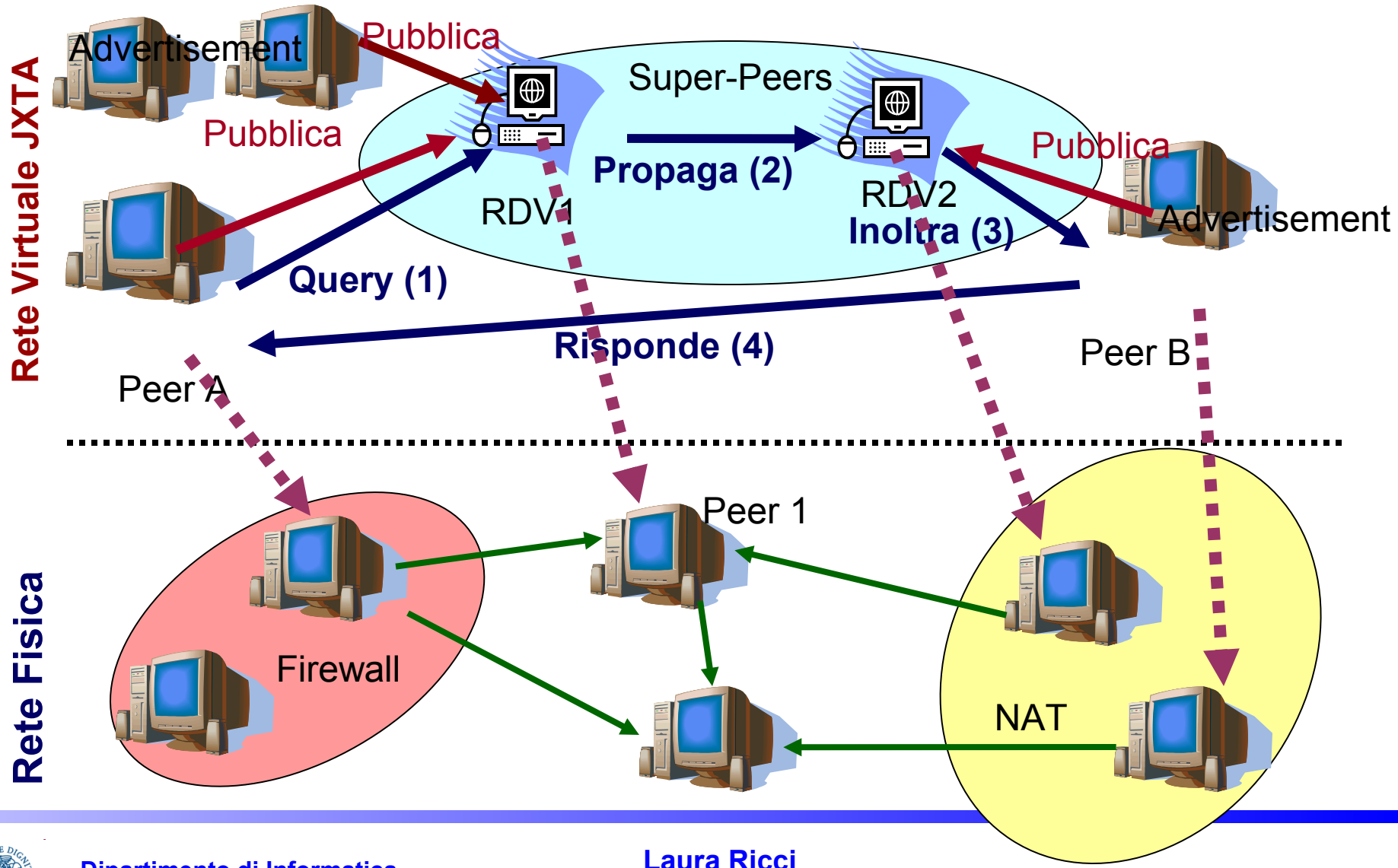
- ogni messaggio di connessione/disconnessione ricevuto dal rendez-vous service provocherà il risveglio del listener, con la notificazione di uno specifico evento.
- previsti diversi possibili eventi: RDVCONNECT, RDVDISCONNECT , RDVFAILED : BECAMERDVOUS, BECAMEEDGE,...

RENDEZ VOUS SUPERPEERS

Funzioni di un Rendez Vouz peer

- memorizza nella propria cache un indice che contiene un insieme di chiavi di advertisements pubblicati in edge peers appartenenti al gruppo
- gli advertisments sono memorizzati unicamente negli edge peers, non nei rendezvous peers
- gli edge peers inviano al rendez-vouz peer le chiavi degli advertisment pubblicati

UN ESEMPIO


Rete Virtuale JXTA
Rete Fisica
Advertisement
Pubblica
Pubblica
Super-Peers
Propaga (2)
RDV1
RDV2
Inoltra (3)
Pubblica
Advertisement
Query (1)
Risponde (4)
Peer A
Peer B
Peer 1
Firewall
NAT