JXTA Overview

Bogdan PASCA
ENS-Lyon
2009

Acknowledgments

Ack

Gabriel Antoniu
IRISA/INRIA Rennes
Projet Paris

Ack

Bernard Traversat
Project JXTA
Sun Microsystems
Santa Clara, CA

Sun's Interest in Peer-to-Peer

- P2P is an instance of our vision — "The Network Is the Computer™"
- Advance Sun products' readiness for P2P

The Network is the computer - Sun Video

Growing interest in P2P

Innovative technology

- Distributed computing applications
  - **SETI@home** (search for ET)
    - **5.2 million participants worldwide,**
    - **distributed computing project with the most participants to date**
  - **Napster** (music exchange)
    - **2001 >25 millions registered users**
  - **Gnutela** (file sharing)
    - **2005, 1.81 million computers**
  - **Kazza** (file sharing)
  - **BitTorrent** (file sharing)
  - Instant messaging services
  - **Skype**
  - **SopCast (P2P Television)**

JXTA

A Generic Framework for P2P Computing

- **Open platform** for P2P programming
- **Common functionalities** for P2P solutions
- Language, OS, network agnostic
  - Java: full implementation available
  - C, Objective C, Perl Ruby, Python

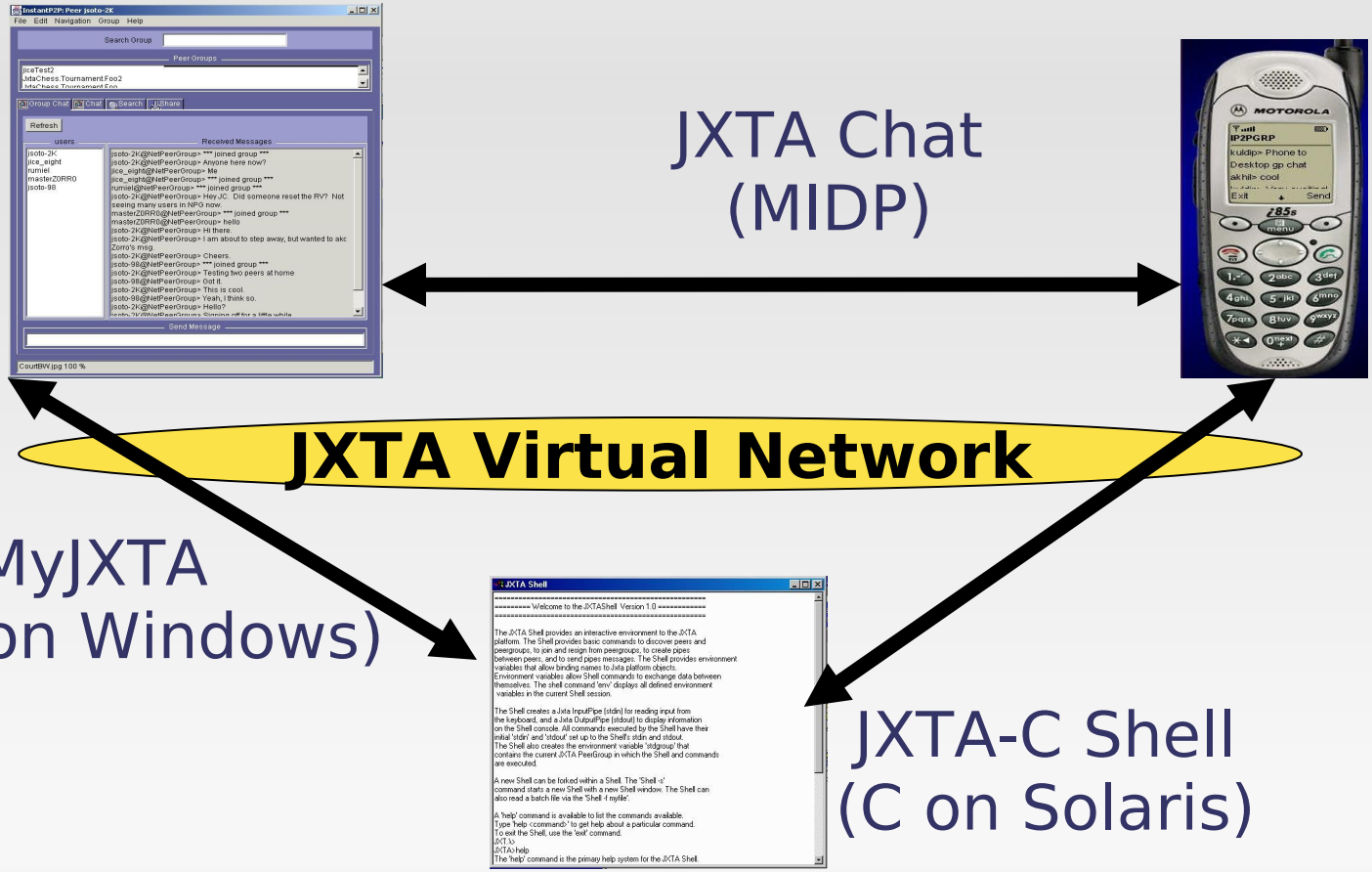
(work in progress)

  - TCP-IP, Bluetooth
- **Set of interoperable protocols** (XML)
- Open source project:

  https://jxta.dev.java.net/

JXTA Interoperability

Any Platform, Any Network



MIDP = Mobile Information Device Profile (part of JAVA Micro Edition)

Project JXTA

Technical Goal

- Build a **small**, **lightweight platform** as the foundation for peer to-peer massively scalable network computing

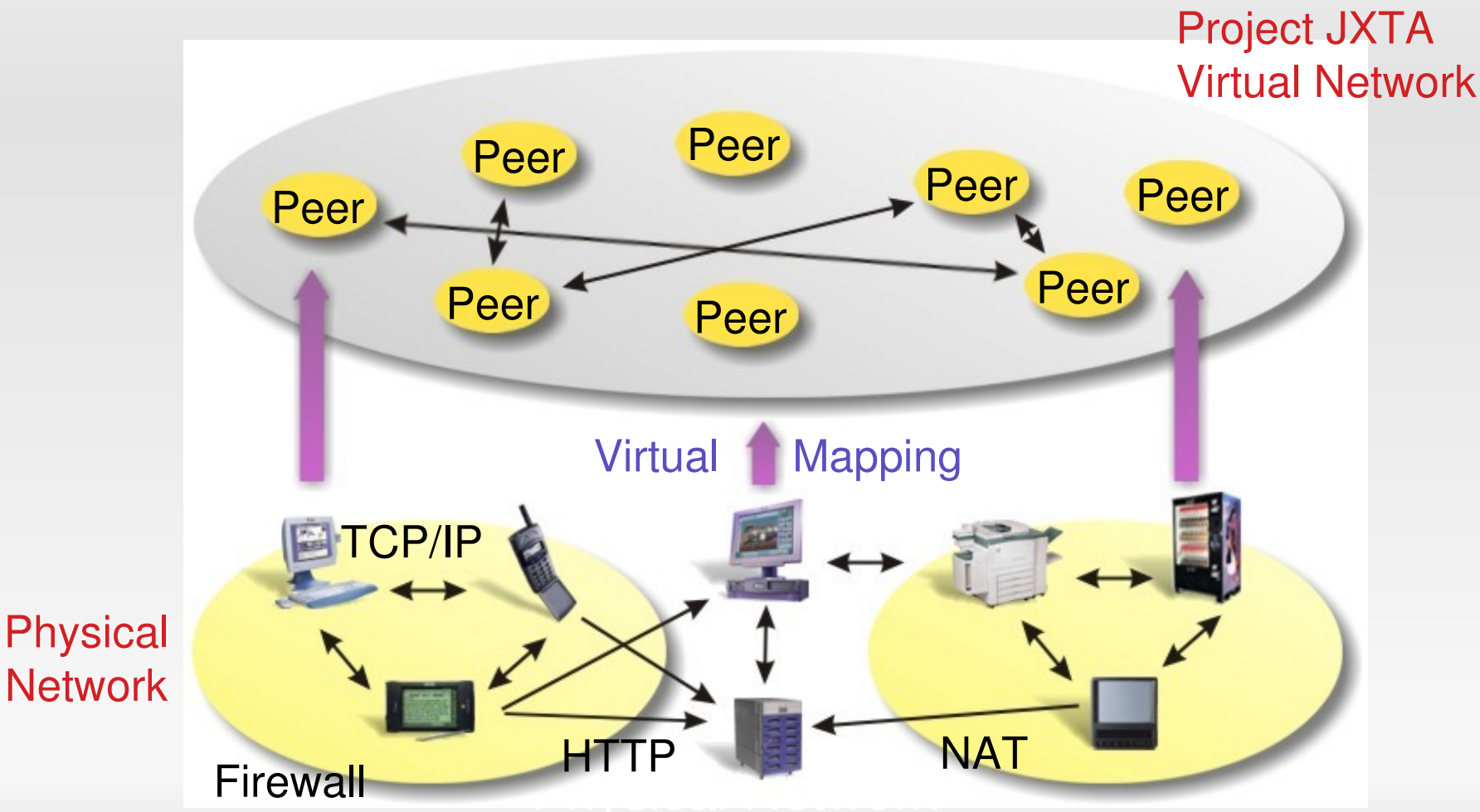
- **jux·ta·pose - v. tr.** *To place side by side, especially for comparison or contrast.*
  - recognition that P2P is juxtaposed to client-server or Web-based computing (today's traditional distributed computing model)

JXTA

Services and Applications

- Distributed storage and data sharing
  - Search, indexing and file sharing
- Large scale distributed computing
- P2P messaging and collaboration tools

JXTA Virtual Network



Peers

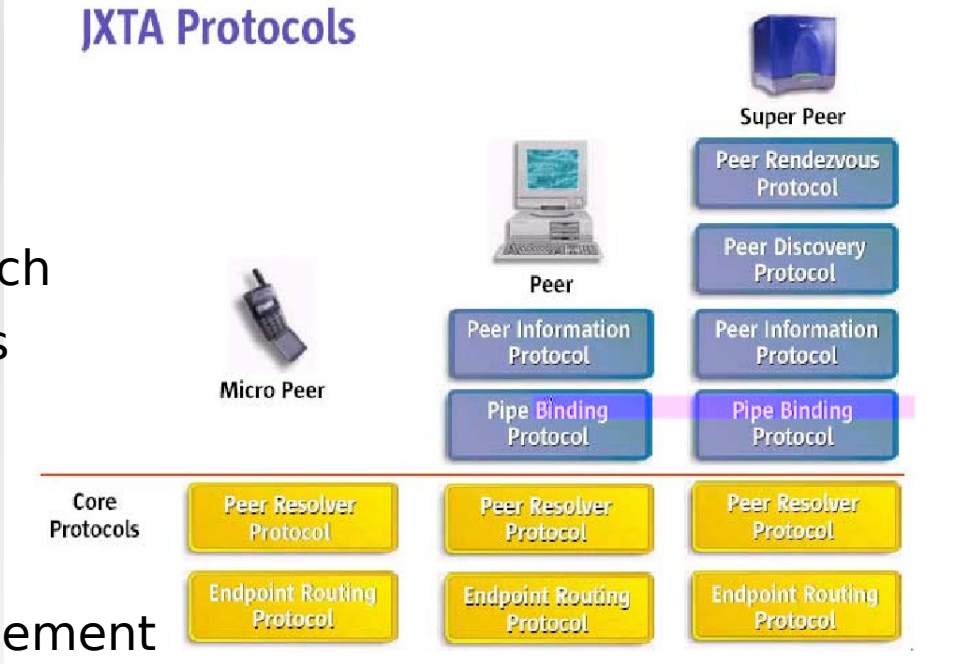
What are?

- Any network device implementing a JXTA Protocol
  - Sensors, phones, PDAs, PCs, Supercomputers
- A peer
  - Unique identifier (128bit UUID)
  - Addressable independently of its location
    - firewalls, NAT
  - Multiple Peer “endpoint” address
    - TCP, HTTP, etc.

Peers

Types

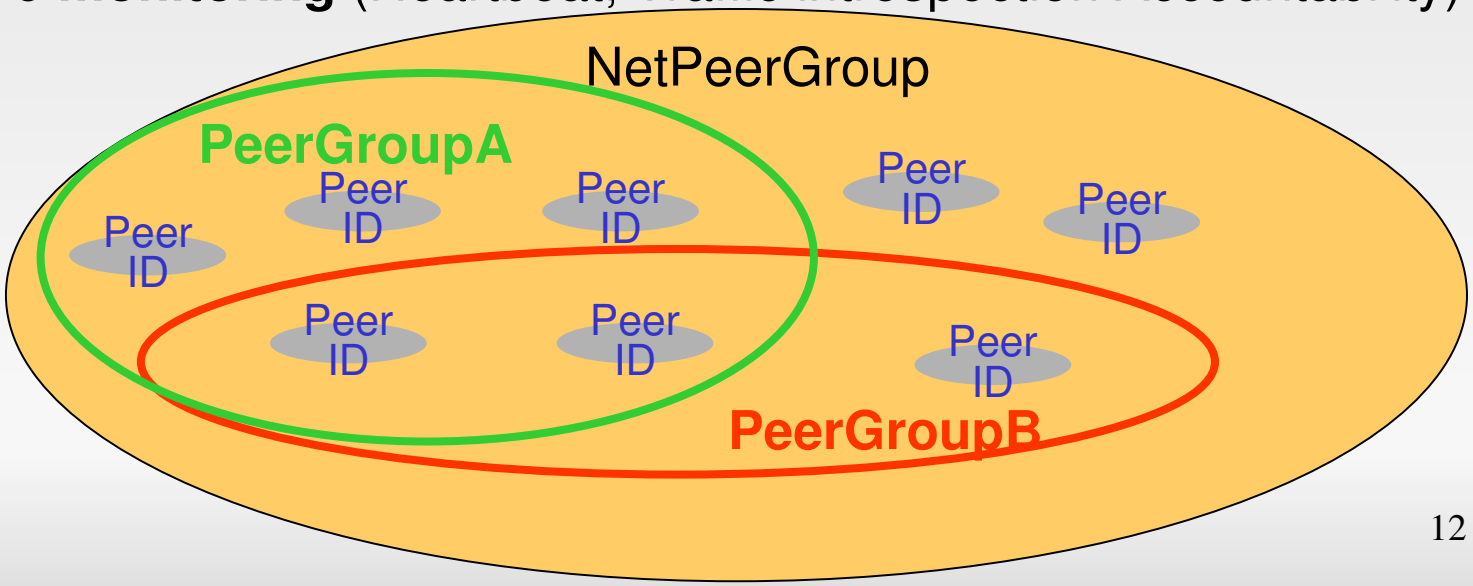
- **Minimal edge**
  - Implements only core JXTA services
  - send/receive
  - Ex: Sensor devices
- **Full edge**:
  - + cache advertisement search
  - Include phones, PCs, Servers
- **Super-Peer:**
  - Rendezvous :
    - + fwd requests
    - Maintains global advertisement indexes
  - Relay :
    - +routing cache +firewall support
  - Proxy
    - Support for minimal edge peers



Peer Groups

Why Peer Groups?

- Provide a “group” identity (common interest)
  - File sharing group
  - CPU Sharing group
- Create secure & protected domains
- Scope peer operations (discovery, search, communications)
- Enable **monitoring** (Heartbeat, Traffic introspection Accountability)



Advertisements

- Every resource is represented by an advertisement
  - Peer
  - Peer group
  - Pipe
  - Service
  - Content
  - Peer status

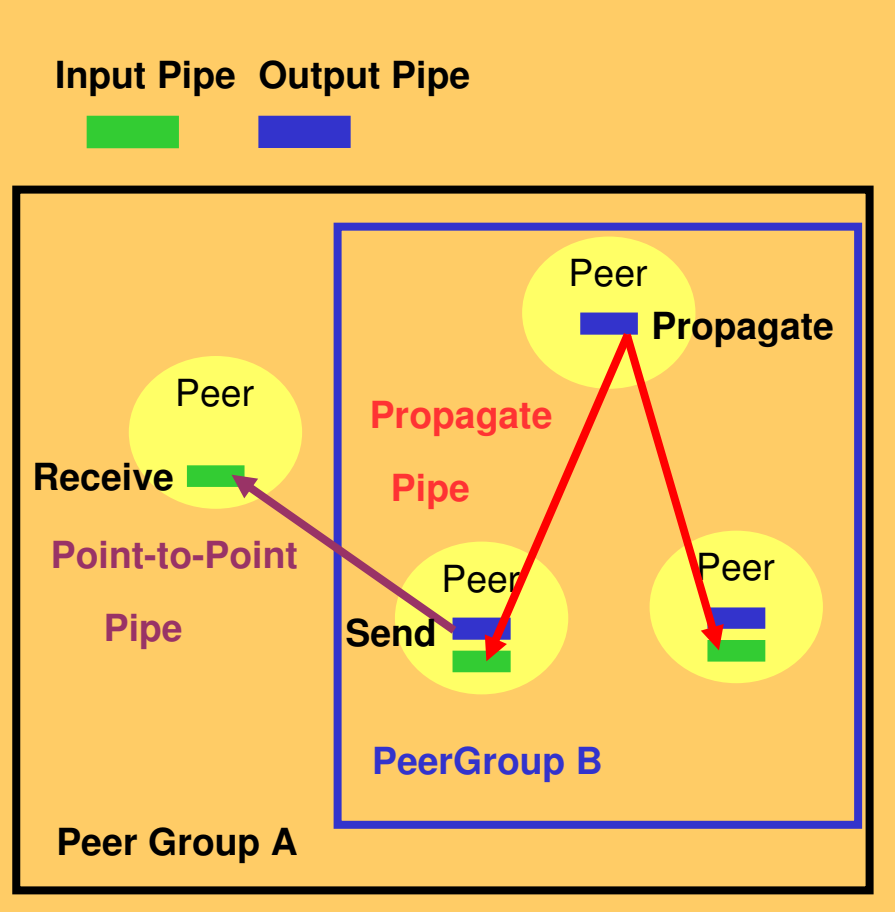
PeerGroup Advertisement:

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA>
    <GID>
        urn:jxta: uuid-
    BCBCDEABDBBBABEABBBABA000000
    </GID>
    <MSID>
        urn:jxta:uuid-
    BFEFDEDFBABAFRUDBACE00000001
    </MSID>
    <Name>
        My Group
    </Name>
    <Desc>
        This group is to be used for my own testing
    </Desc>
</jxta:PGA>
```

Pipe

Virtual Communication Channel

- Non-localized communication channel between two or more peers
  - Uni-directional
  - Asynchronous
  - Unreliable



Pipe

Communication Model

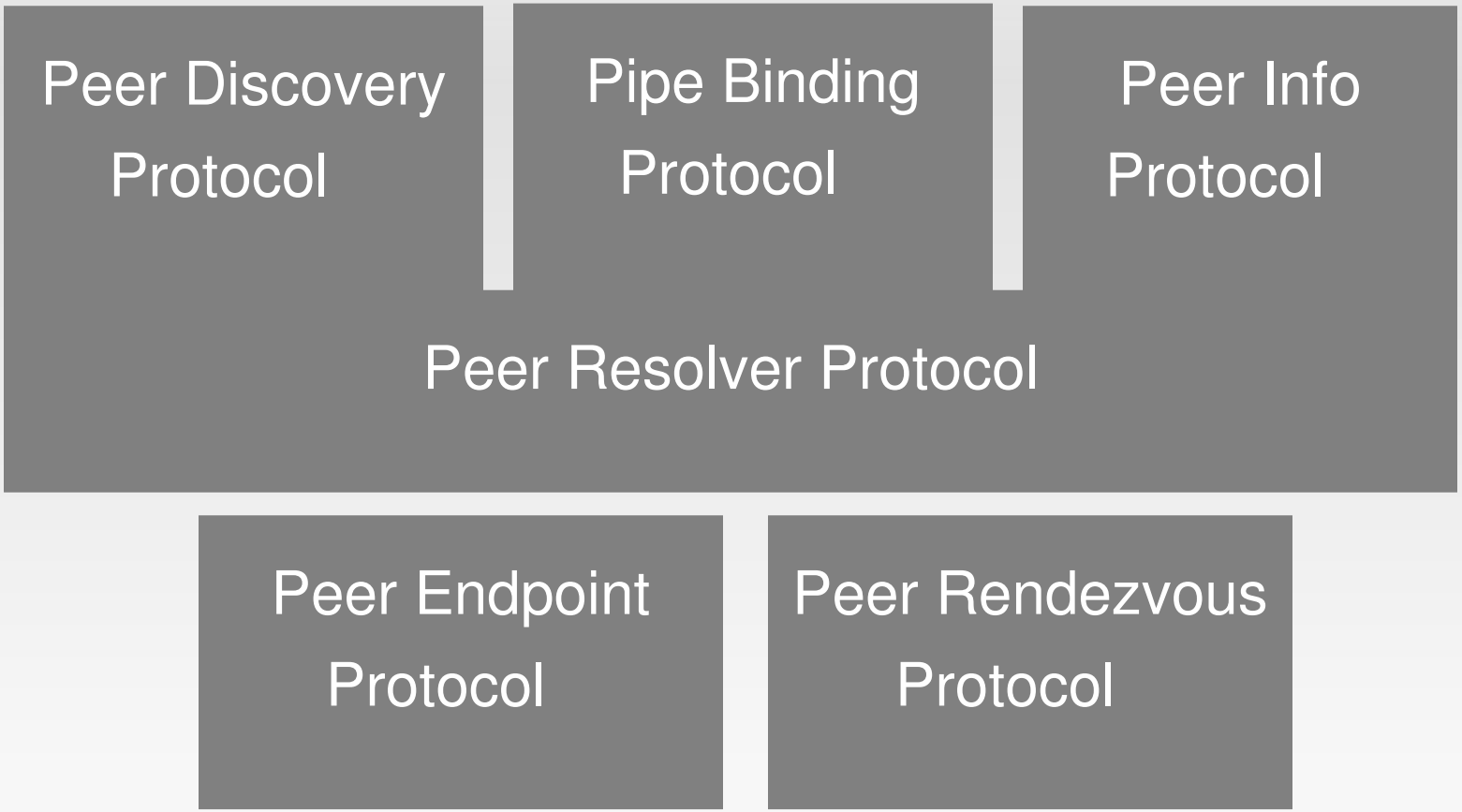
- Connect to services independently of their peer locations
- Dynamic binding
  - At pipe creation or for every message sent
- Build highly-available services
  - Transparent fail-over by reconnecting pipe endpoints
- Pipeline multiple services to form complex service

Network Services

- Any kind of service available on the network
- Split in two categories:
  - Peer Services
    - Instantiated only on the pear that publishes the service
  - PeerGroup Services
    - Collection of service instances running on multiple members of a peer group
    - Potentially on all peer members
- Can be dynamically loaded

JXTA

Protocol Stack



JXTA

Peer Discovery Protocol

- Purpose: discover a JXTA resource
  - Described by advertisement
  - Service, pipes, peers, peer-groups etc.
- Discovery
  - decentralized, centralized or hybrid
- Two levels of discovery
  - Joining a JXTA network
    - Multicast
    - Unicast (peer knows the rendezvous node location)
  - Discovering a JXTA resource within a JXTA network
    - Cascaded discovery (the horizon of the second peer)
    - Via rendezvous peers

JXTA

Pipe Binding Protocol

- Allows establishing a virtual communication between one or more peers
- Peer binds pipe advertisement to pipe endpoint
- Pipe ends maintain the virtual link, re-establish it if necessary

JXTA

Peer Information Protocol

- Allows peers to find-out about capabilities and status of other peers:
  - Uptime, traffic load, state
- Examples:
  - ping to see if alive
  - Query a peers properties

JXTA

Peer Resolver Protocol

- Enables a peer to implement high-level search capabilities
- Allows send/receive of generic queries:
  - Find or search: peers, peer groups, pipes, other information

JXTA

Endpoint Routing Protocol

- Find destination peer route information
- Peers often not directly connected to each other
- Example:
  - A → B but no direct route
  - A needs intermediary peers to route the message
- Peers implementing the protocol respond:
  - List of gateways along the route

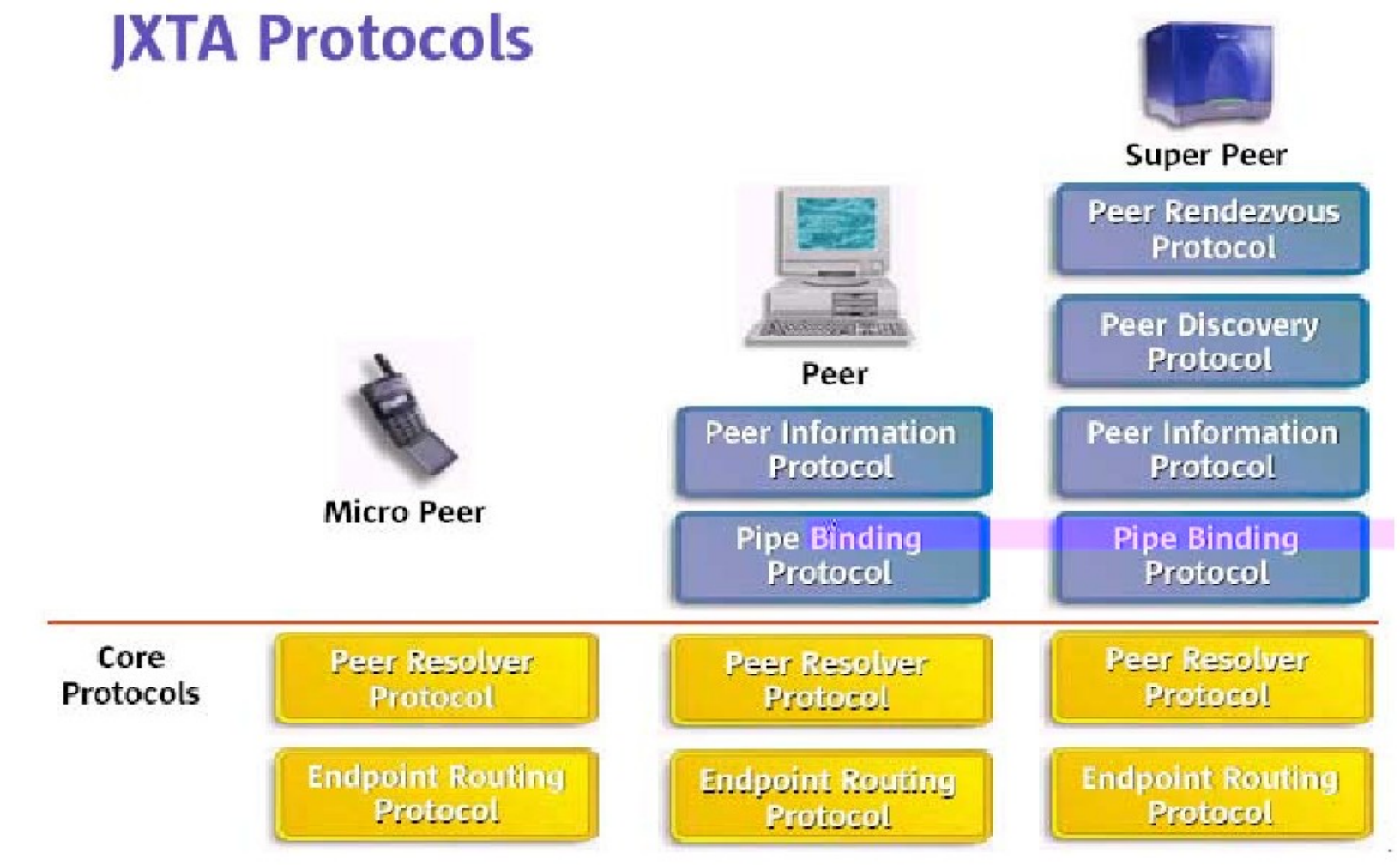
JXTA

Rendezvous Protocol

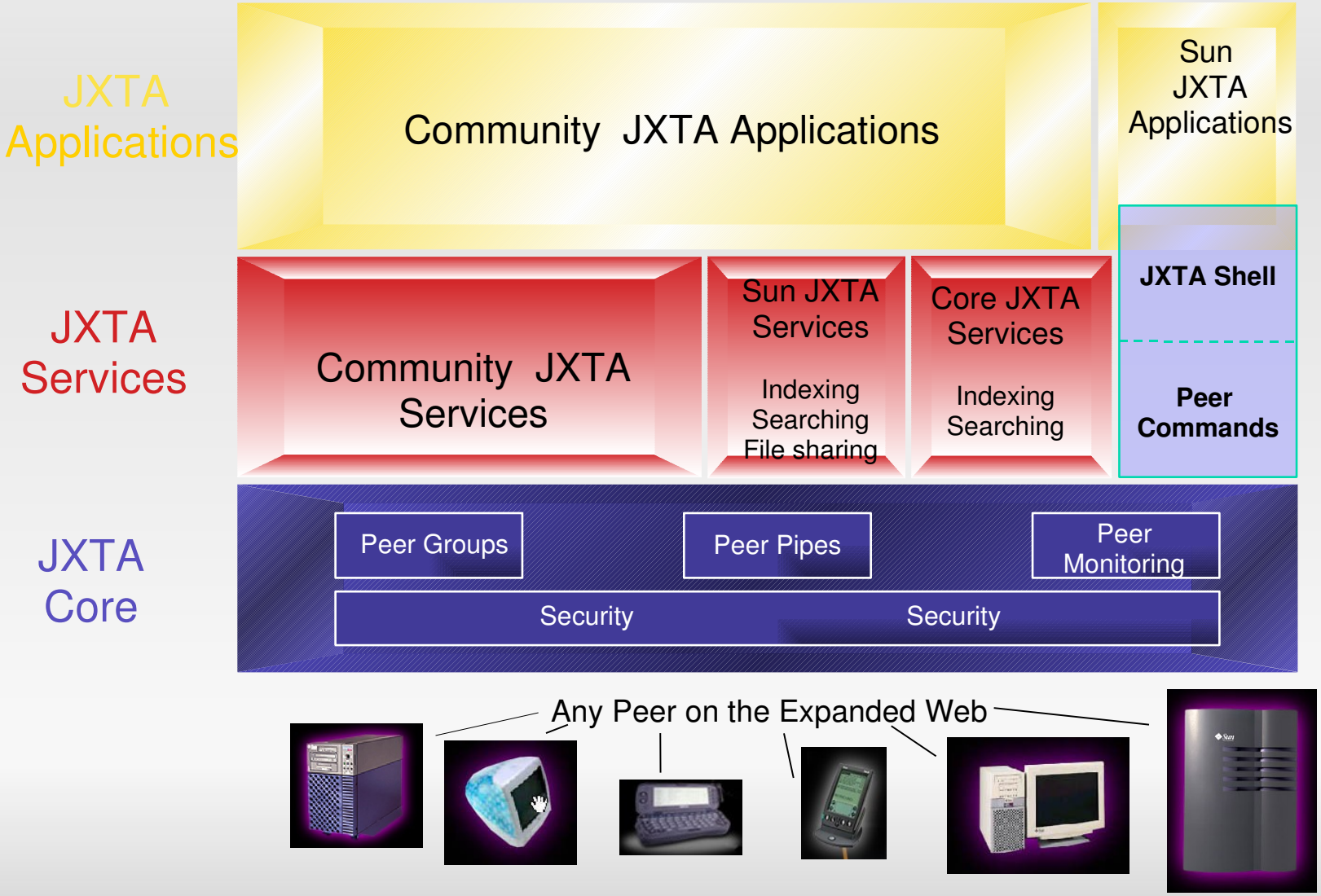
- Can send messages to all service listeners
- Used to propagate messages
  - Peer resolver protocol
  - Pipe binding protocol

JXTA

Protocols



P2P Software Architecture



JXTA

Core Services (1)

- Discovery Services:
  - Way peers can discover each other, the existence of other peer groups, pipes, services, and the like.
- PeerInfo Service
- Pipe Services
  - Main means of communications between peers; provides an abstraction for a one-way, asynchronous conduit for information transfer

JXTA

Core Services (2)

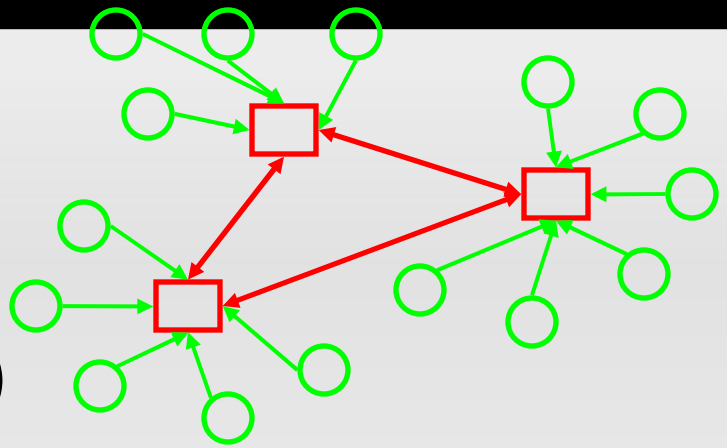
- Resolver Services:
  - Allows peers to refer to each other, peer groups, pipes, or services indirectly through a reference (called an advertisement in JXTA lingo); the resolver binds the reference to an implementation at run time.
- Membership Services:
  - Determines which peers belong to a peer group; handles arrival and departure of peers within a peer group
- Access Services:
  - Security service for controlling access to services and resources within a peer group; a sort of security manager for the peergroup

Lookup mechanisms in P2P systems

- Peers have high churn rate
- Based on DHTs (Chord, Pastry, Tapestry, etc)
  - Efficient
  - But costly to maintain a consistent distributed index
- Based on crawling (Gnutella, KaZaA, etc)
  - Expensive
  - But do not have any maintenance cost

⇨ Hybrid approach: loosely-consistent DHT

Organization of a JXTA overlay



- Type of peer involved
  - Edge peer
  - Rendezvous peer
- Concept of peer view (PV)
  - Ordered list of known rendezvous peers
  - Peer view protocol for the discovery of other rendezvous peers
- Structured overlay
  - Loosely-Consistent DHT between rendezvous peers
  - Mechanism for ensuring convergence of the PV across rendezvous peers
    - Key point for the efficient of the loosely-consistent DHT
    - Fallback mechanism: limited-range walker

Time for the PV to converge across rendezvous peers? (1/4)

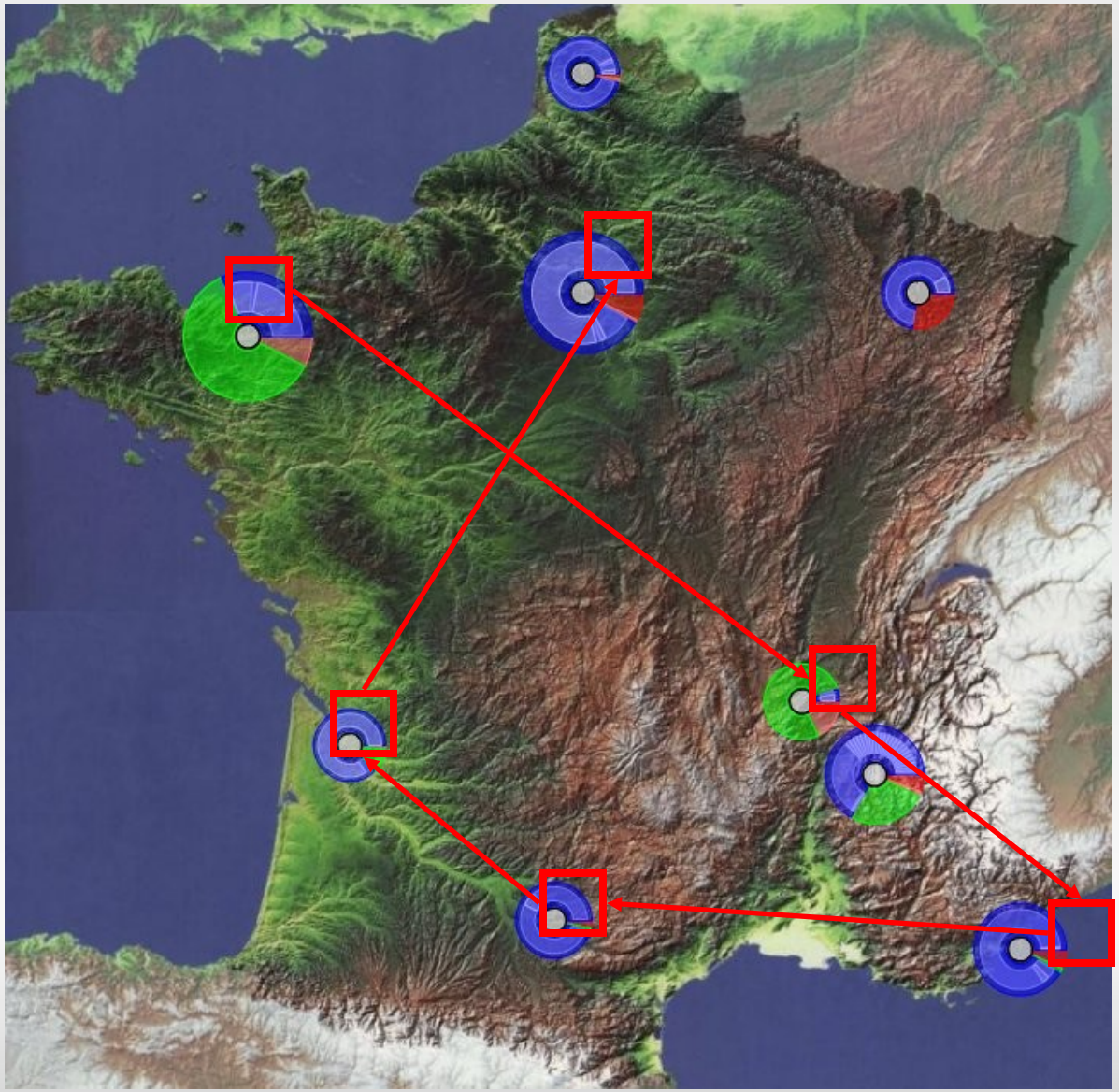
Testbed: up to 580 nodes

- Rennes (161), Bordeaux (44), Orsay (185), Toulouse (50), Sophia (100) and Lyon (40)

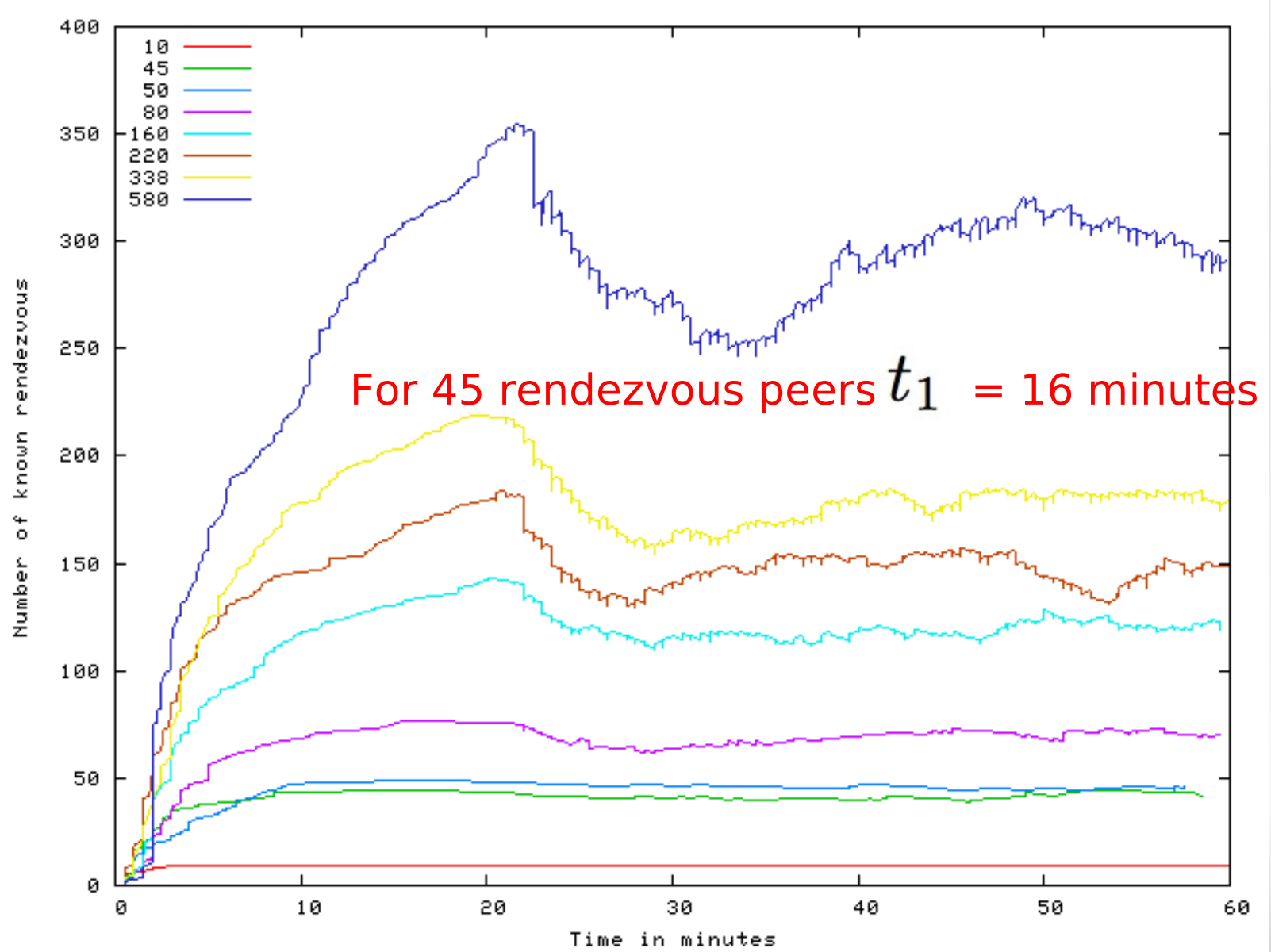
- Logical overlay
  - n rendezvous peers (n = 2, 10, 80, 160, 580)
  - No delay between startup
  - Initial topology: simple link between rendezvous peers
- Duration: up to 2 hours

Grid'5000

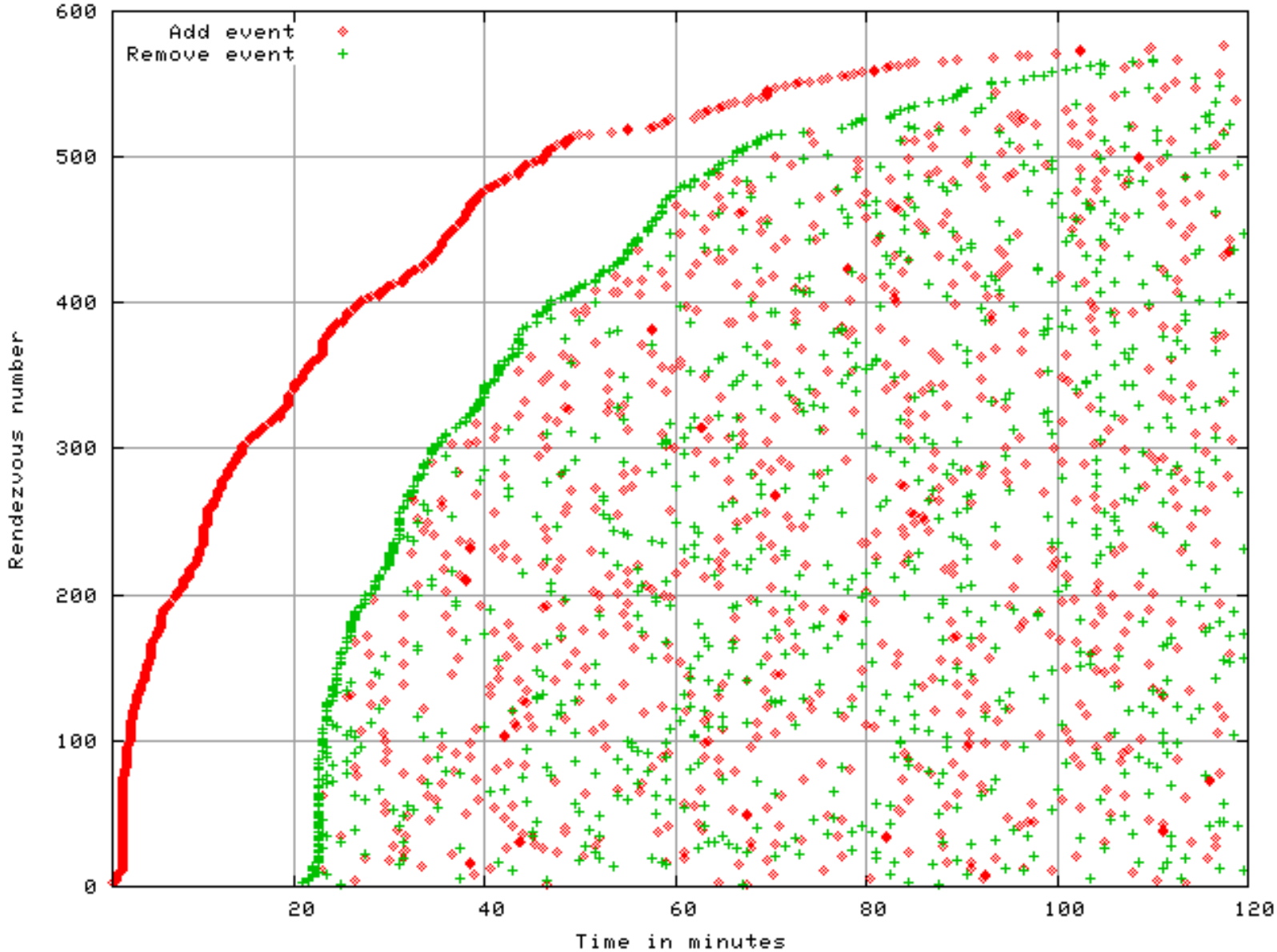
Time for the PV to converge across rendezvous peers? (2/4)



Time for the PV to converge across rendezvous peers? (3/4)

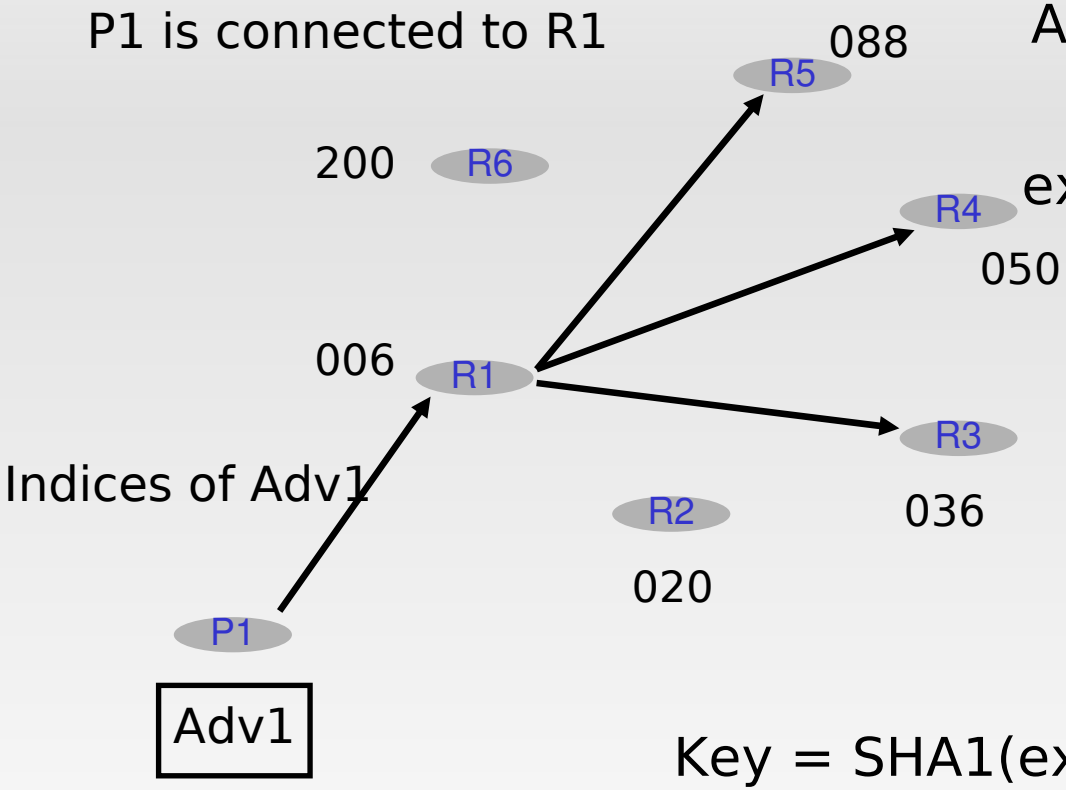


Time for the PV to converge across rendezvous peers? N = 580 (4/4)



How are advertisements published?

P1 is connected to R1



Adv of type Peer, with attribute
Name = JuxMem

expression = PeerNameJuxMem

key = 116

rank(3) = R4

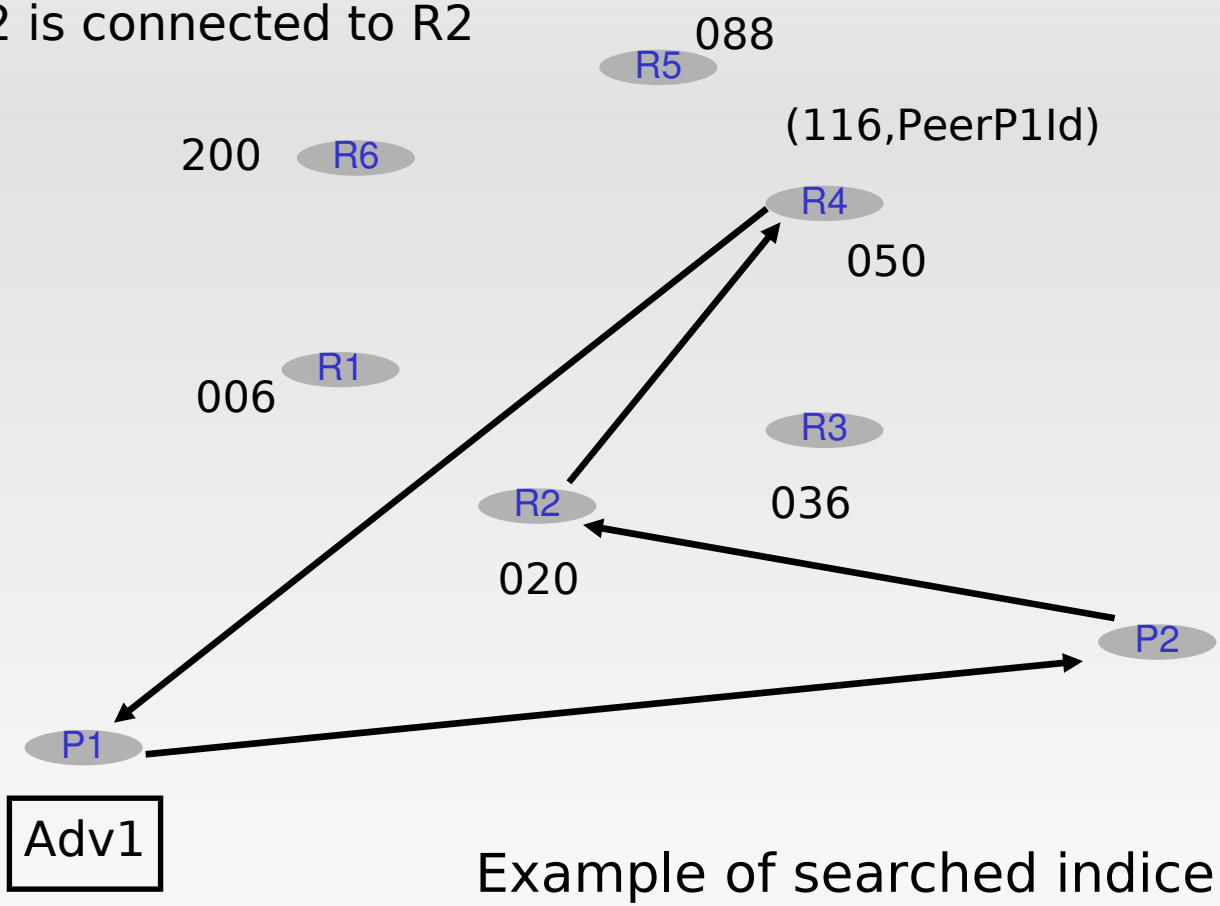
Key = SHA1(expression)

hash_space = max value returned by hash function

pos = floor(key * size_perview / hash_space)

How advertisements are searched?

P2 is connected to R2



PeerID
006
020
036
050
088
200

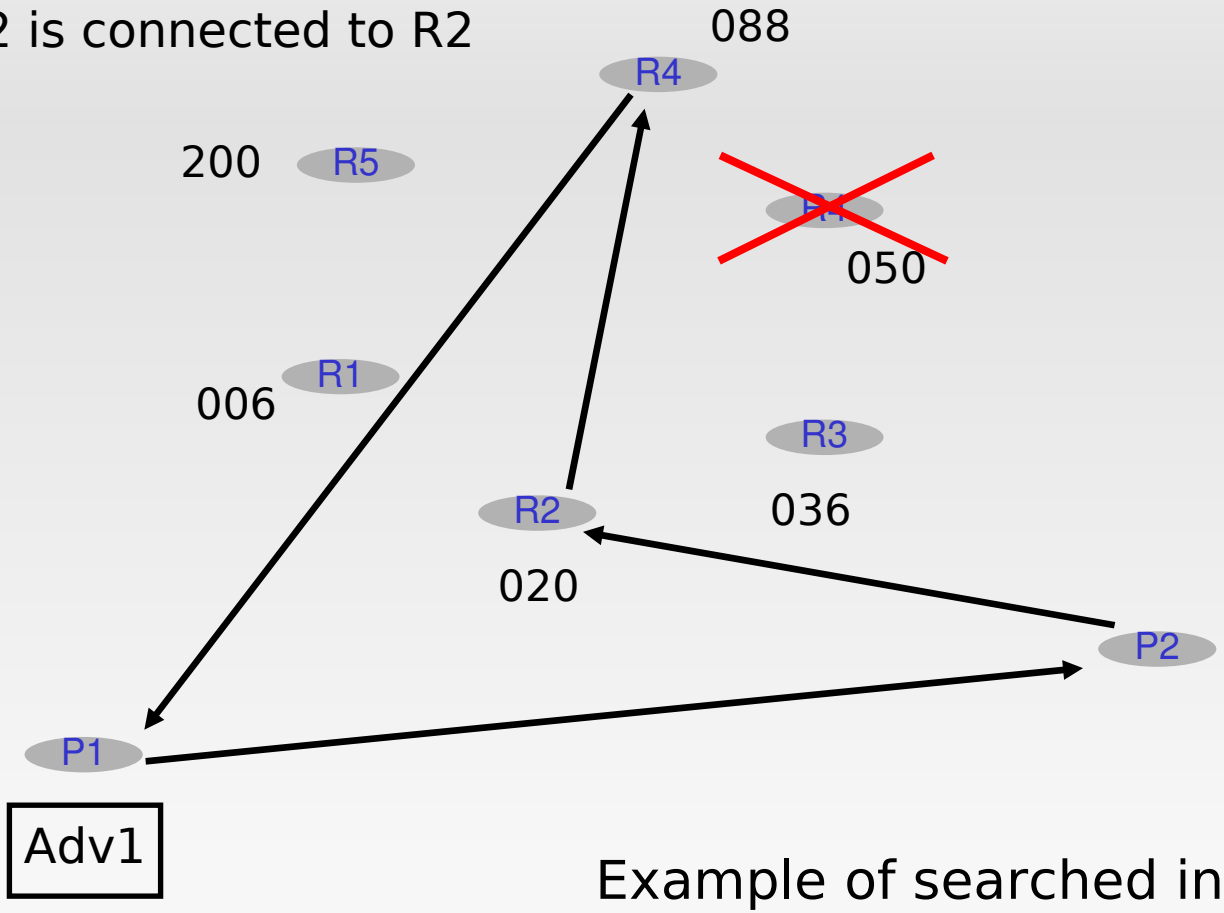
RPV of R2

Example of searched indice:

rank(replicaPeer(PeerNameJuxMem)) = R4

Inconsistent view

P2 is connected to R2



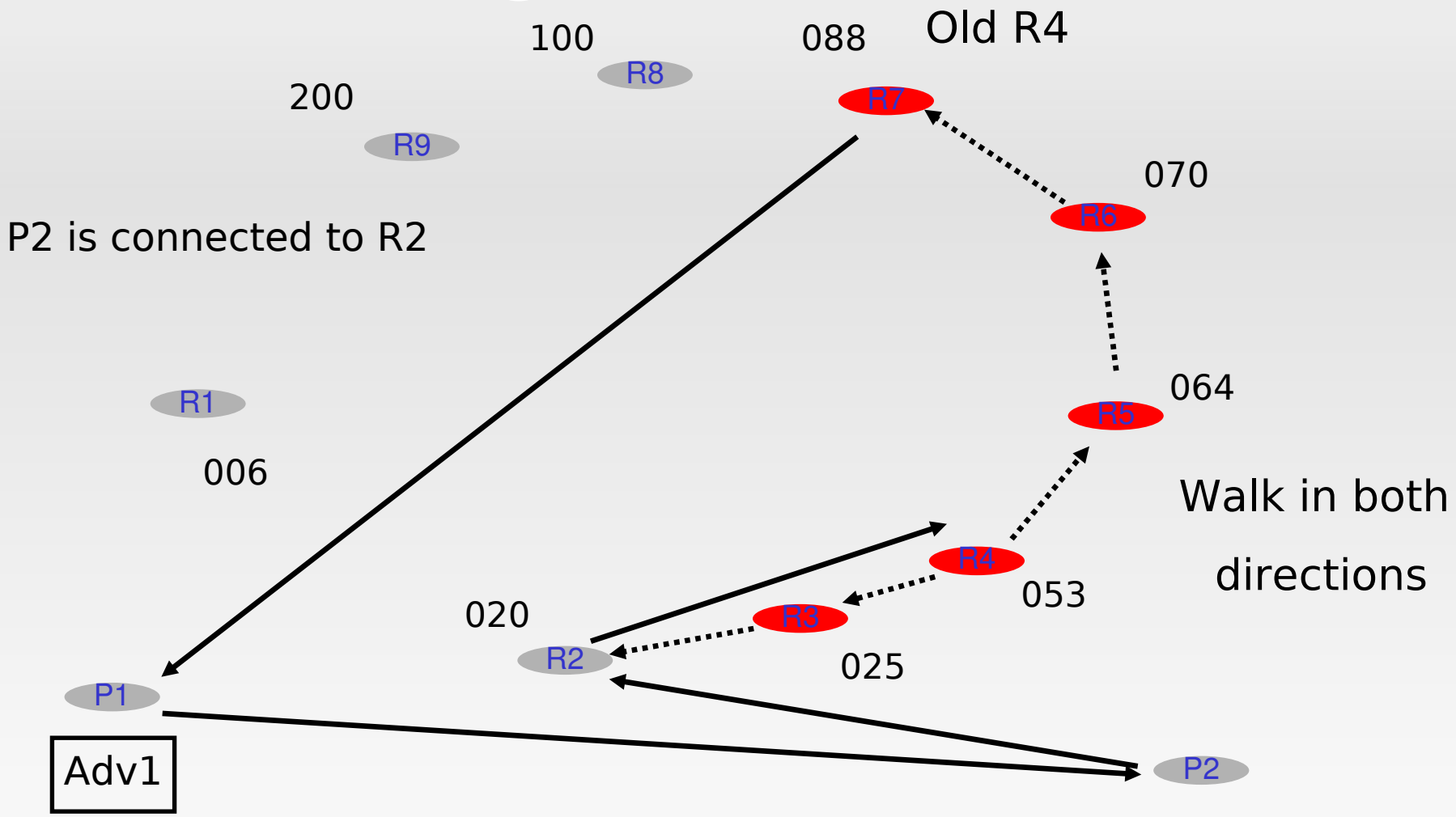
PeerID
006
020
036
088
200

RPV of R2

Example of searched indice:

rank(replicaPeer(PeerNameJuxMem))
= R4

Limited-range walker



rank(Peer+Name+JuxMem) = R4

Finding Rendezvous Peers

- Edge peers maintain lists of rendezvous peers
- Dynamic fail-over when connection fails
- Edge peers discover and cache Rdv advertisements
- Seeding Rdvs are used to bootstrap
- Auto-promotion to Rdv if none can be found in PeerGroup

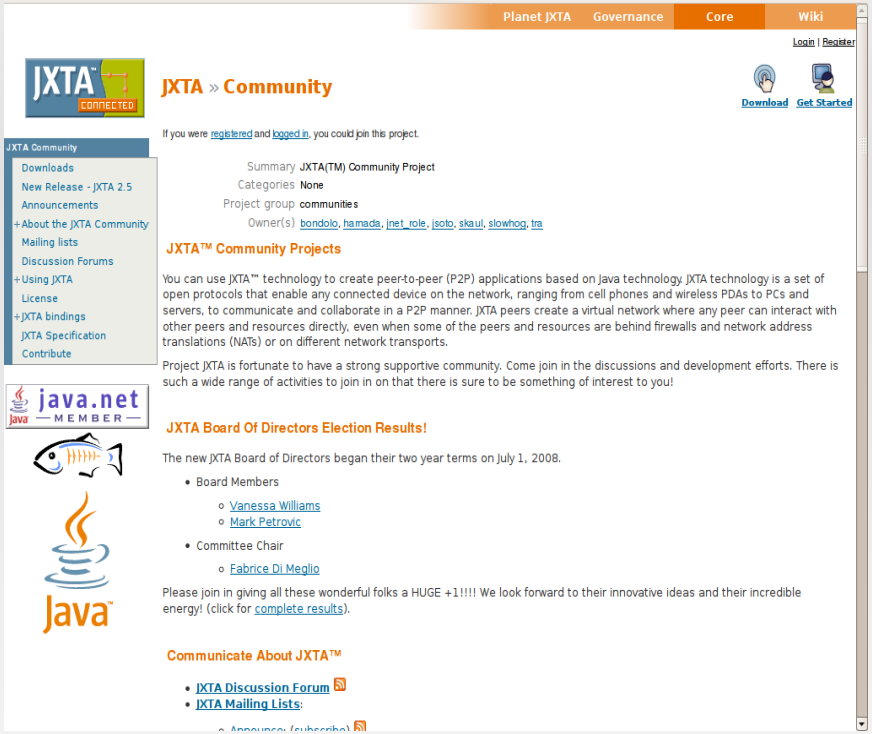
Conclusion:

JXTA technology today

- Enhanced network and J2SE platform
  - Mature, stable, secure, scalable platform.
  - 2.0 spec, code, demos, docs, and tutorials on-line
  - Public virtual network in place
  - Release 2.5
- JXTA-C and JXME implementations available
  - JXTA-C better on some points than JXTA-J2SE
  - JXTA-C 2.5 & JXME 2.5
- Community projects & industrial choice

JXTA Community

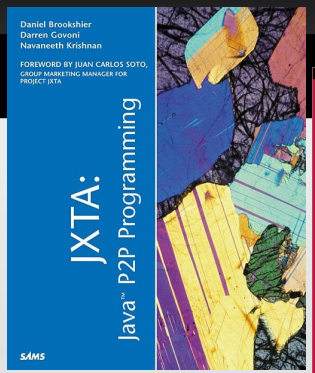
- Over 2,700,000 downloads
- 120+ active projects
- 18,000+ members
- Active discussion groups
- Community actively contributing and integrating technology

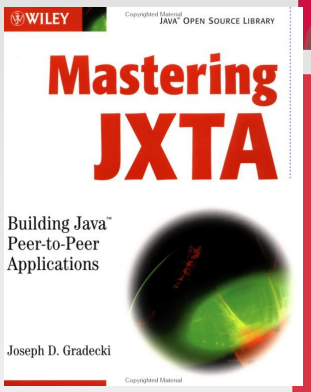


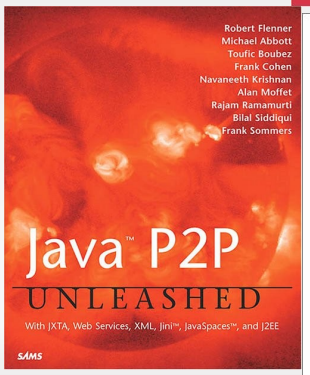
https://jxta.dev.java.net/

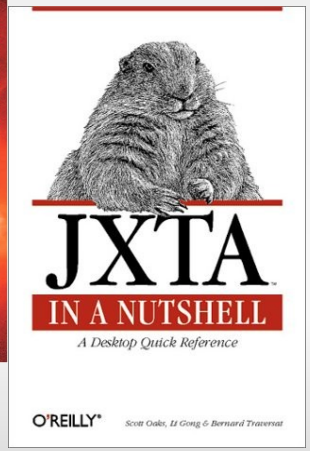
JXTA Bookshelf

- JXTA: P2P Computing with Java, Sing Li, 2002
- JXTA, Brendon J. Wilson, 2002
- JXTA: Java P2P Programming, Daniel Brookshire et al, 2002
- Mastering JXTA Development, Joe Gradecki, 2002
- Java P2P Unleashed, Robert Flenner et al, 2002
- JXTA in a Nutshell, Scott Oaks et al, 2002









Let's get started !

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

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();
     }
}
```

Creating and Publishing a Service

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

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);
```

Create a Module Spec Advertisement

```
ModuleSpecAdvertisement mdadv = (ModuleSpecAdvertisement)

    AdvertisementFactory.newAdvertisement(ModuleSpecAdvertisement.getAdvertisem
    entType());

   // 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();
   }
```

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);
```

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();
  }
}
```

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

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 );
```

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.getOutputPipe();
        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" );
}
```