sip2peer Tutorial
Android Example

Marco Picone

Source Code:
Android-Sip2Peer-1.0.zip

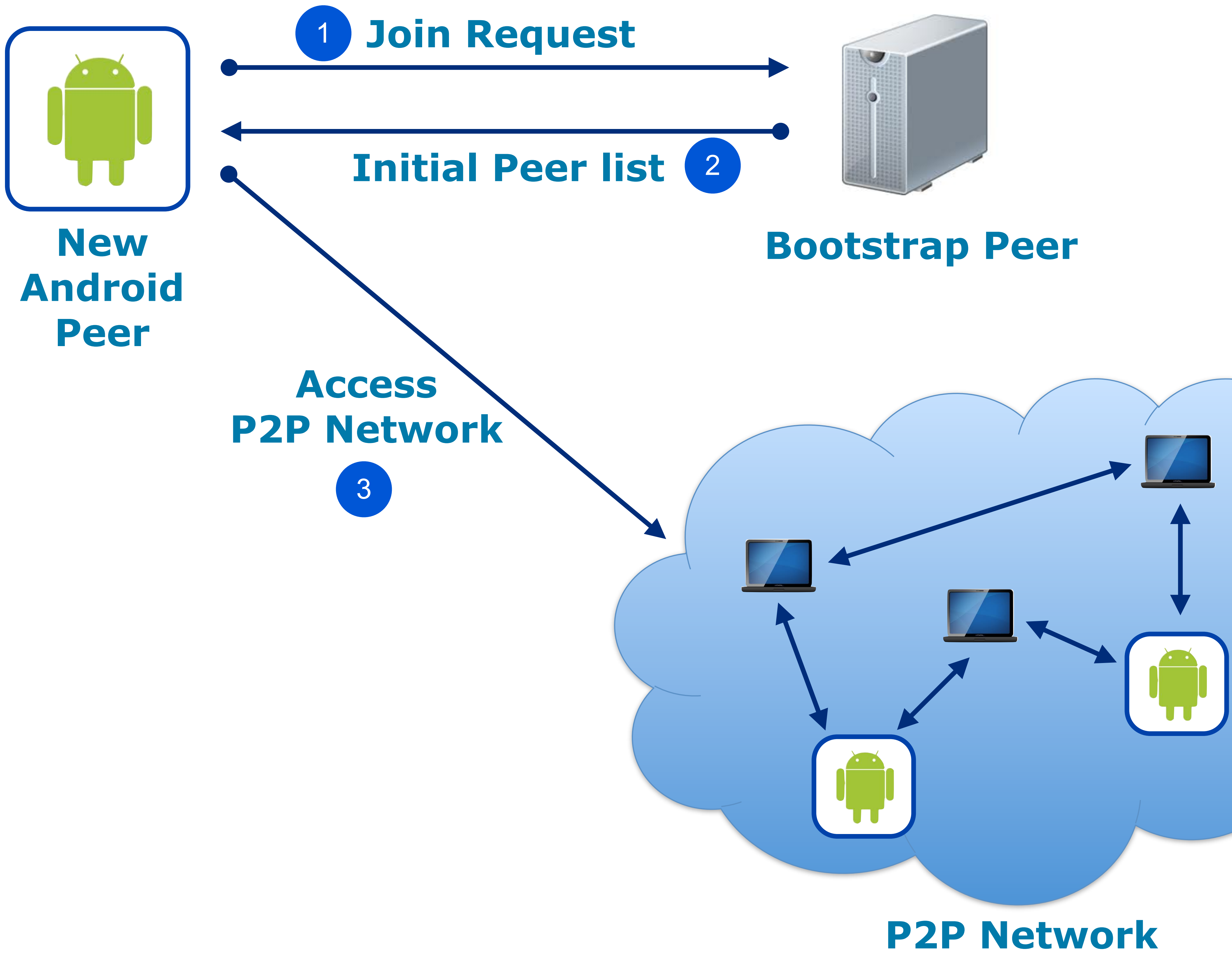
Università degli Studi di Parma
Parma, Italy

Outline

- Introduction
- Application Structure
- sip2peer on Android

Introduction

This tutorial shows how to implement an Android application based on sip2peer that can interact with a bootstrap server to receive a list of active peers and exchange PING messages with them. The role of the node is the same illustrated in the tutorial "*s2p-Tutorial-Bootstrap-FullPeer-SBC.pdf*" available on the project web site. The first part of the tutorial illustrates the structure of the provided application, while the second one is dedicated to the implementation and how to use sip2peer on the Android platform.
The application is just an example showing how is easy with sip2peer to build and heterogeneous P2P system made by mobile and PC nodes implementing the same behavior.


1 Join Request
Initial Peer list 2
New Android Peer
Bootstrap Peer
Access P2P Network 3
P2P Network

Source Structure

• **SimplePeer:** Extends Peer base class to implement the node behavior (as described in the first sip2peer tutorial [link]).

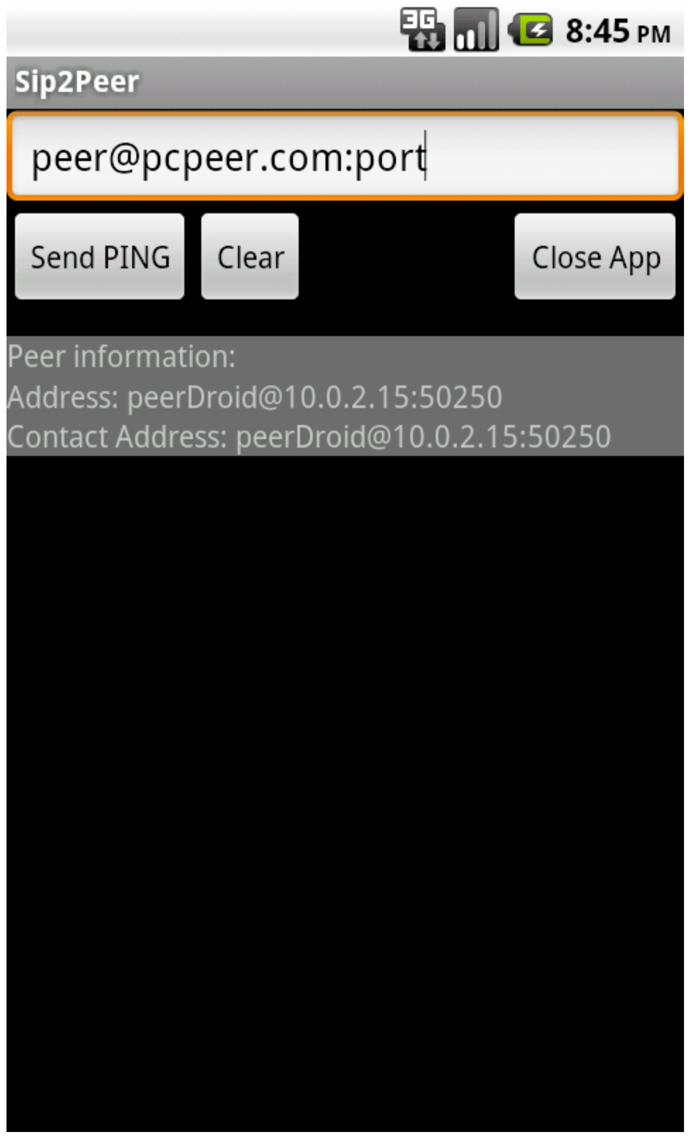
• **PeerActivity:** Main application activity, used to show information about the peer, send PING messages and access other available sections.

• **ConfigPeerActivity:** Activity dedicated to the peer configuration. It allows to set the bootstrap and SBC address and if necessary to activate the reachability check for peers in the same network.

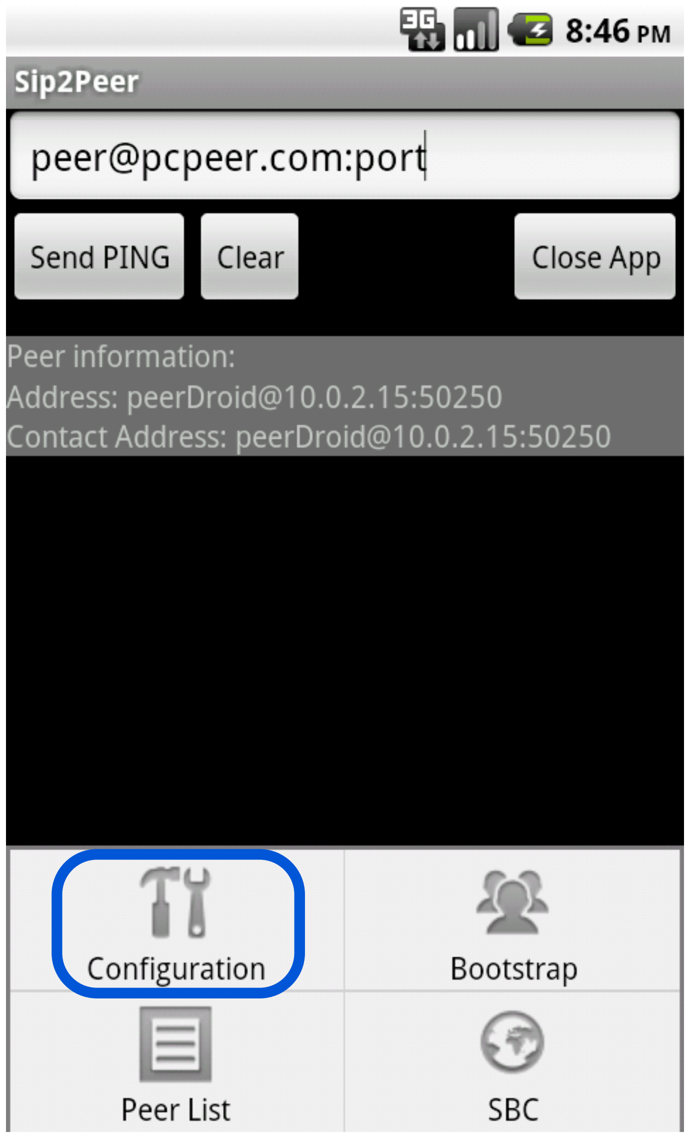
• **PeerListActivity:** Keeps updated a view with the list of known peers initially received from the bootstrap node.

• **JoinPeerActivity:** Activity used to send the first Join message to the bootstrap peer in order to receive the initial list of available peers in the network.

Main UI & Menu


8:45 PM
Sip2Peer
peer@pcpeer.com:port

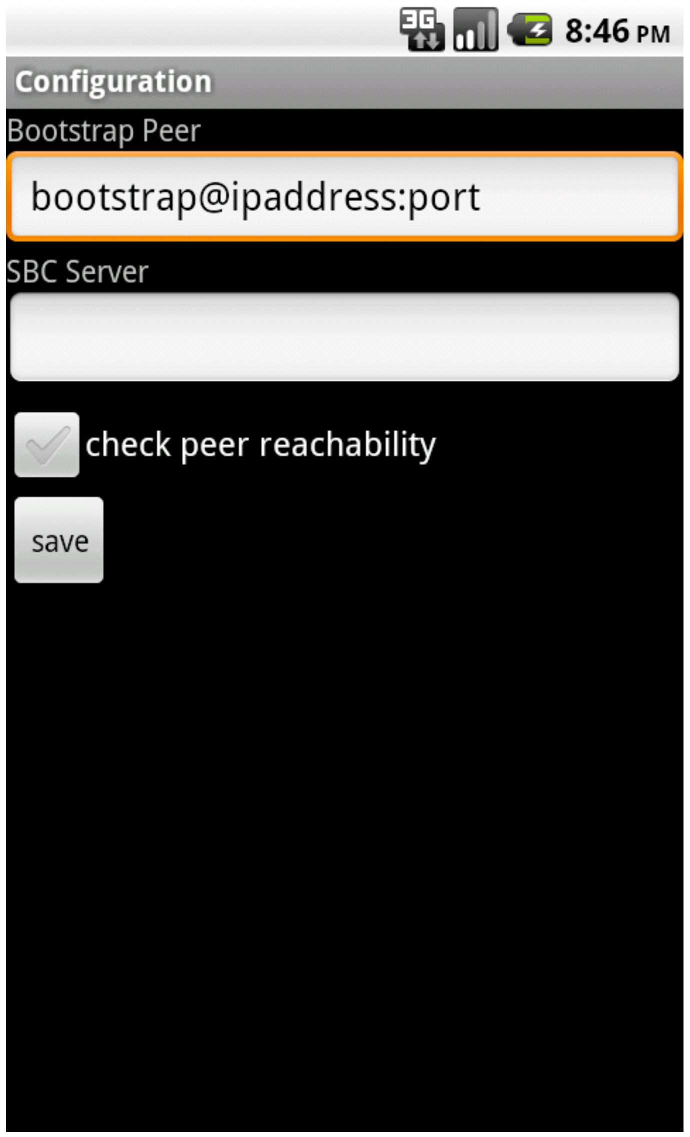
Peer information:
Address: peerDroid@10.0.2.15:50250
Contact Address: peerDroid@10.0.2.15:50250


8:46 PM
Sip2Peer
peer@pcpeer.com:port

Peer information:
Address: peerDroid@10.0.2.15:50250
Contact Address: peerDroid@10.0.2.15:50250

On the left side, the main screen of the application. It shows information about the running node, such as IP address, name and contact address.
Before starting the join procedure and the communication with other connected peers, it is necessary to configure the node though the dedicated activity available by clicking on the Configuration button in the main menu.

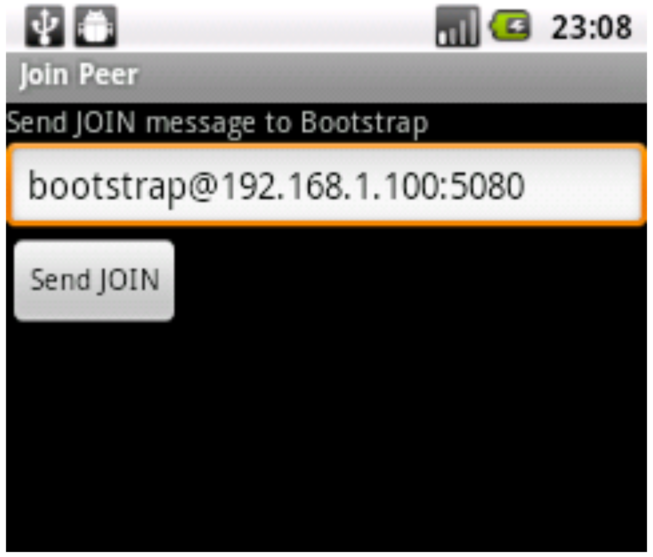
Configuration


8:46 PM
Configuration
Bootstrap Peer
bootstrap@ipaddress:port
SBC Server
check peer reachability
save

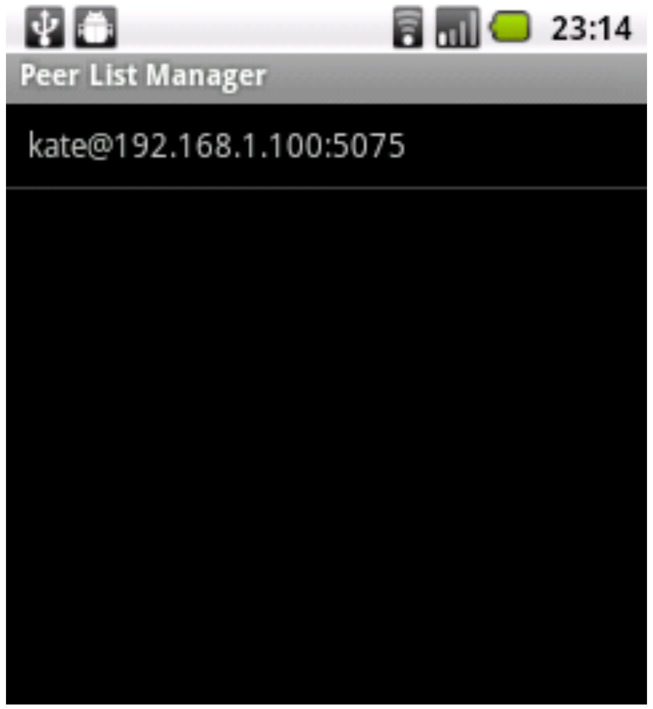
Configuration Activity allows to set:

- **Bootstrap Peer Address:** IP address and listening port of the bootstrap peer used to join the peer-to-peer network and retrieve the list of already connected nodes. It is mandatory for this example scenario.
- **SBC Server Address:** IP address and listening port of the SBC server. SBC is used to manage communication with peers behind NAT (read the related tutorial [link-1] [link-2]). Is not mandatory and should be used only if necessary.
- **Check peer reachability:** If checked, the node verifies if known nodes are in the same network and if it is possible to communicate using their local IP addresses.

Bootstrap Join & PeerList

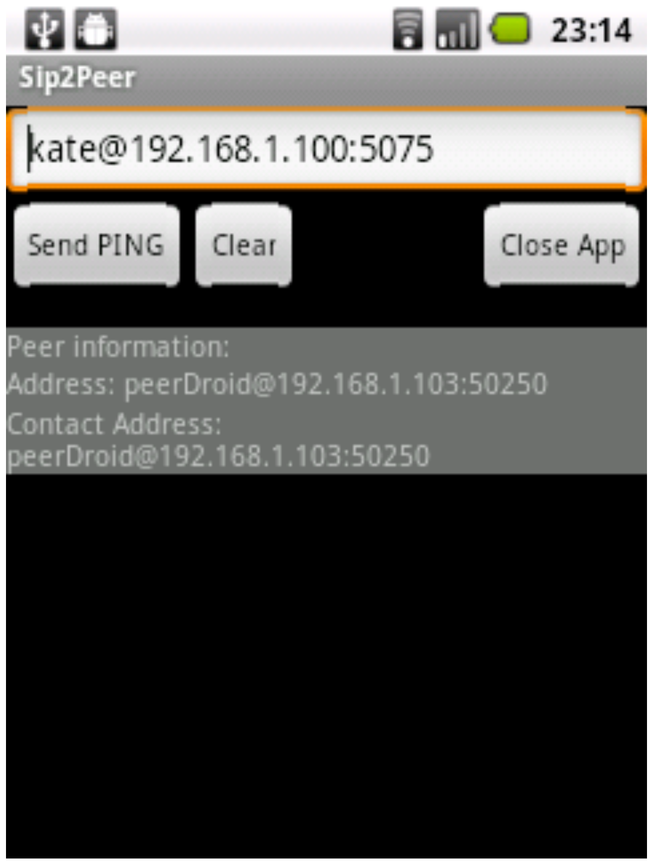

23:08
Join Peer
Send JOIN message to Bootstrap
bootstrap@192.168.1.100:5080

When node configuration is completed, by clicking on the "Bootstrap" button in the main menu it is possible to send the Join Message to the Bootstrap peer in order to complete the access procedure to the network and receive the initial list of available peers.


23:14
Peer List Manager
kate@192.168.1.100:5075

When the list has been correctly received, it is shown in the associated view in the main menu though the button "PeerList". Information available for each peer can be used to send a PING message from the main view of the application.

Ping Message



kate@192.168.1.100:5075

Peer information:
Address: peerDroid@192.168.1.103:50250
Contact Address:
peerDroid@192.168.1.103:50250

By using the address provided in the peer list or by clicking on one of listed rows, it is possible to send a PING message to a target node. Incoming messages will appear as popup messages in the destination node.

sip2peer on Android

```
public class SimplePeer extends Peer
{
    ...

    protected void onReceivedJSONMsg(JSONObject jsonMsg, Address sender){...}

    protected void onDeliveryMsgFailure(String peerMsgSended, Address receiver,String contentType)
    {...}

    protected void onDeliveryMsgSuccess(String peerMsgSended, Address receiver,String contentType)
    {...}

    ...
}
```

As illustrated in the previous tutorial, by extending the s2p **Peer** class and implementing message managements methods it is possible to have full control over all needed communication aspects.

SimplePeer.java located in the package **it.unipr.ce.dsg.s2p.example.peerdroid** represents a first and easy example about how to use sip2peer library also on the Android platform.

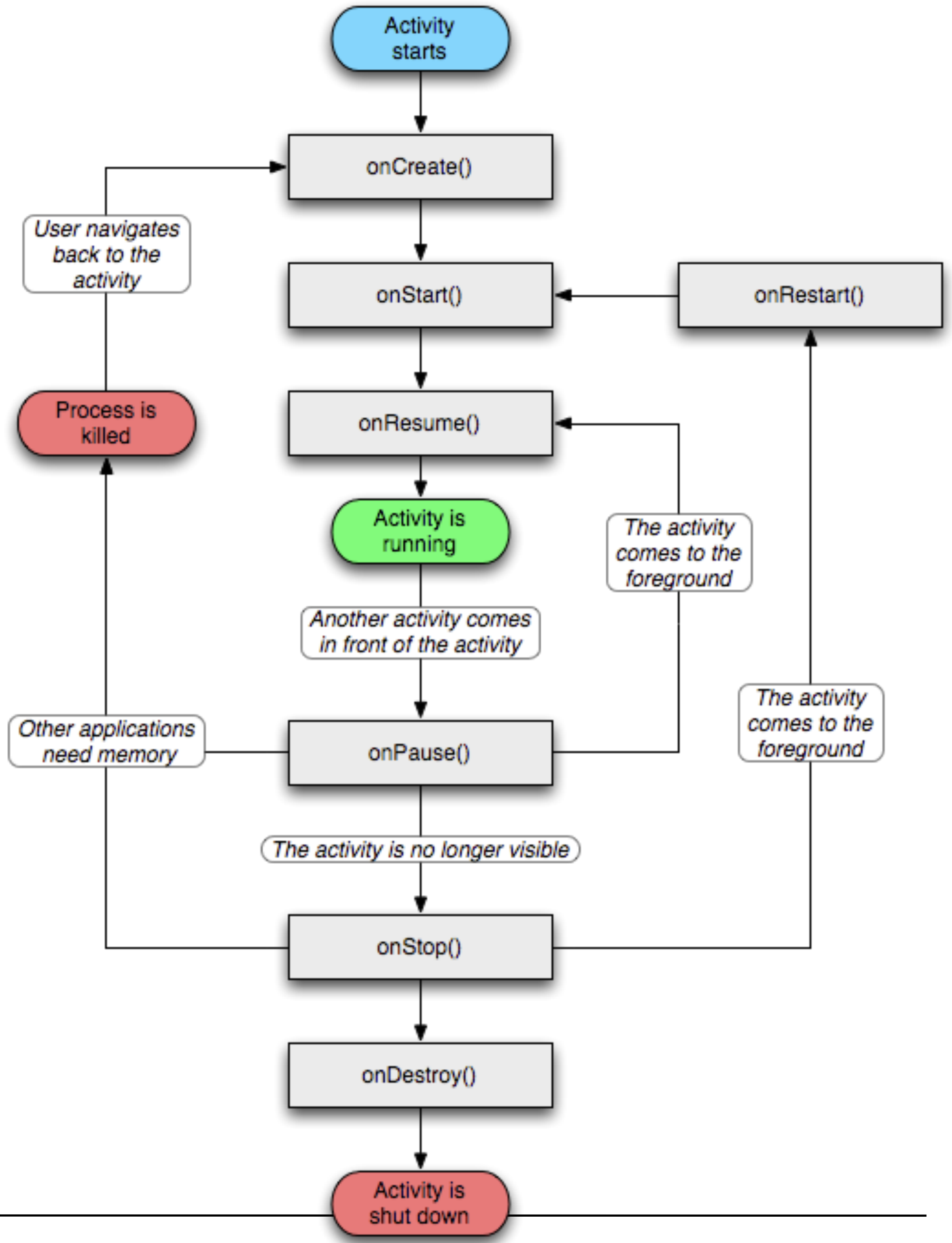
Furthermore since Android is Java-based and the imported jar files are exactly the same used for the Java standard edition, the majority of your code could be the same for both implementation.

On the Android platform you should only take care of a different initialization method without the configuration file path for the Peer and the interaction between the node instance and the user interface.

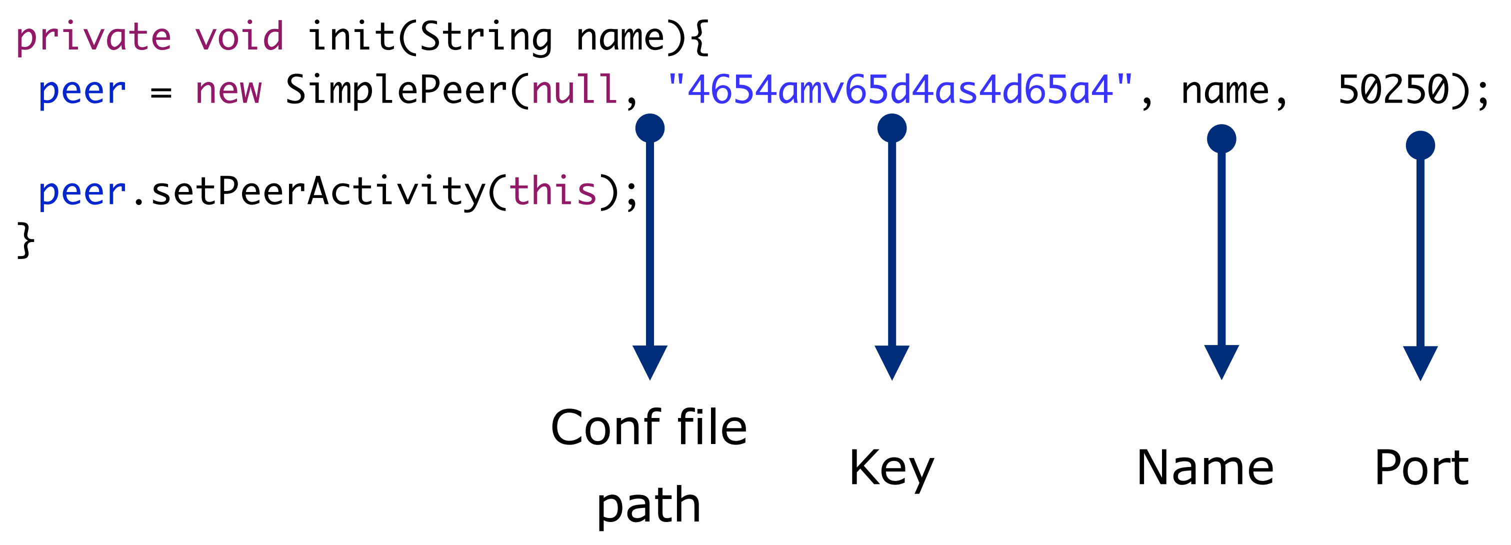
sip2peer on Android

```
public static SimplePeer peer = null;

....

@Override
protected void onStart() {
     super.onStart();
     if(peer==null){

          init("peerDroid");

          addressPeer.setText("Address: "+peer.getAddressPeer());
     }
}
```

Main application activity (PeerActivity.java) maintains an instance of the SimplePeer class. It is initialized in the init function invoked in the onCreate(). It is a default method of each activity called at the beginning when it becomes visible to the user. Peer instance is used for the communication with other nodes and interacts with the user interface in order to show the information of incoming messages and to read user inputs related to the type of message that has to be sent.


Activity starts
onCreate()
onStart()
onRestart()
User navigates back to the activity
onResume()
Process is killed
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

sip2peer on Android



```java
private void init(String name){
 peer = new SimplePeer(null, "4654amv65d4as4d65a4", name,  50250);

 peer.setPeerActivity(this);
}
```

Init function uses the extended Peer constructor allowing to specify fundamental parameters without the configuration file that has been used in the other tutorials. In details, used parameters are:

- **Configuration file:** In this case it is null because we specify the parameters through the constructor.
- **Key:** Unique identifier of the peer in the network.
- **Name:** Name of the peer, could coincide with the key.
- **Port:** Listening port used by the node to receive incoming messages.

SimplePeer class in this case keeps (with a proper set method) the reference of the main activity in order to update the UI if necessary when an incoming message arrives.

sip2peer on Android

```
sendBut.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View arg0) {

  if(peer!=null) {
    if(addressEdit.getText().toString().contentEquals("peer@pcpeer.com:port")){
         ...
    }
    else{

        peer.pingToPeer(addressEdit.getText().toString());

        addressEdit.setText("");
    }
  }
  else
   showDialog(DIALOG_CONFIG);
}});
```

SimplePeer object in main activity can be used as in the example above to send a message using the provided method *pingToPeer* or the traditional send functions provided by the base class.

```
public void send(Address toAddress, BasicMessage message);

public void send(Address toAddress, Address toContactAddress, BasicMessage message)
```

sip2peer on Android

```
@Override
protected void onReceivedJSONMsg(JSONObject jsonMsg, Address sender) {

try {

 JSONObject params = jsonMsg.getJSONObject("payload").getJSONObject("params");

 [...]

 if(jsonMsg.get("type").equals(PingMessage.MSG_PEER_PING))
 {

  PeerActivity.handler.post(new Runnable() {
  public void run() {
     Toast toast = Toast.makeText(peerActivity.getBaseContext(),"Received: "+ PingMessage.MSG_PEER_PING ,Toast.LENGTH_LONG);
     toast.show();
  }});

  PeerDescriptor neighborPeerDesc = new PeerDescriptor(params.get("name").toString(), params.get("address").toString(), params.get("key").toString(),
params.get("contactAddress").toString());
  addNeighborPeer(neighborPeerDesc);

 }

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

Inside override onReceiveJSONMsg method of SimplePeer class it is possible to use (as in the above piece of code) a PeerActivity instance in order to update the application user interface. Using an handler object is you can for example to update a label or as showed to show a popup message about the incoming message.

License

http://www.gnu.org/licenses/gpl.html

Contact us for specific releases or additional information about the license.

Contacts

Designer(s)

- Marco Picone (picone@ce.unipr.it)
- Fabrizio Caramia (fabrizio.caramia@studenti.unipr.it)
- Michele Amoretti (michele.amoretti@unipr.it)

Developer(s)

- Fabrizio Caramia (fabrizio.caramia@studenti.unipr.it)
- Marco Picone (picone@ce.unipr.it)

http://dsg.ce.unipr.it

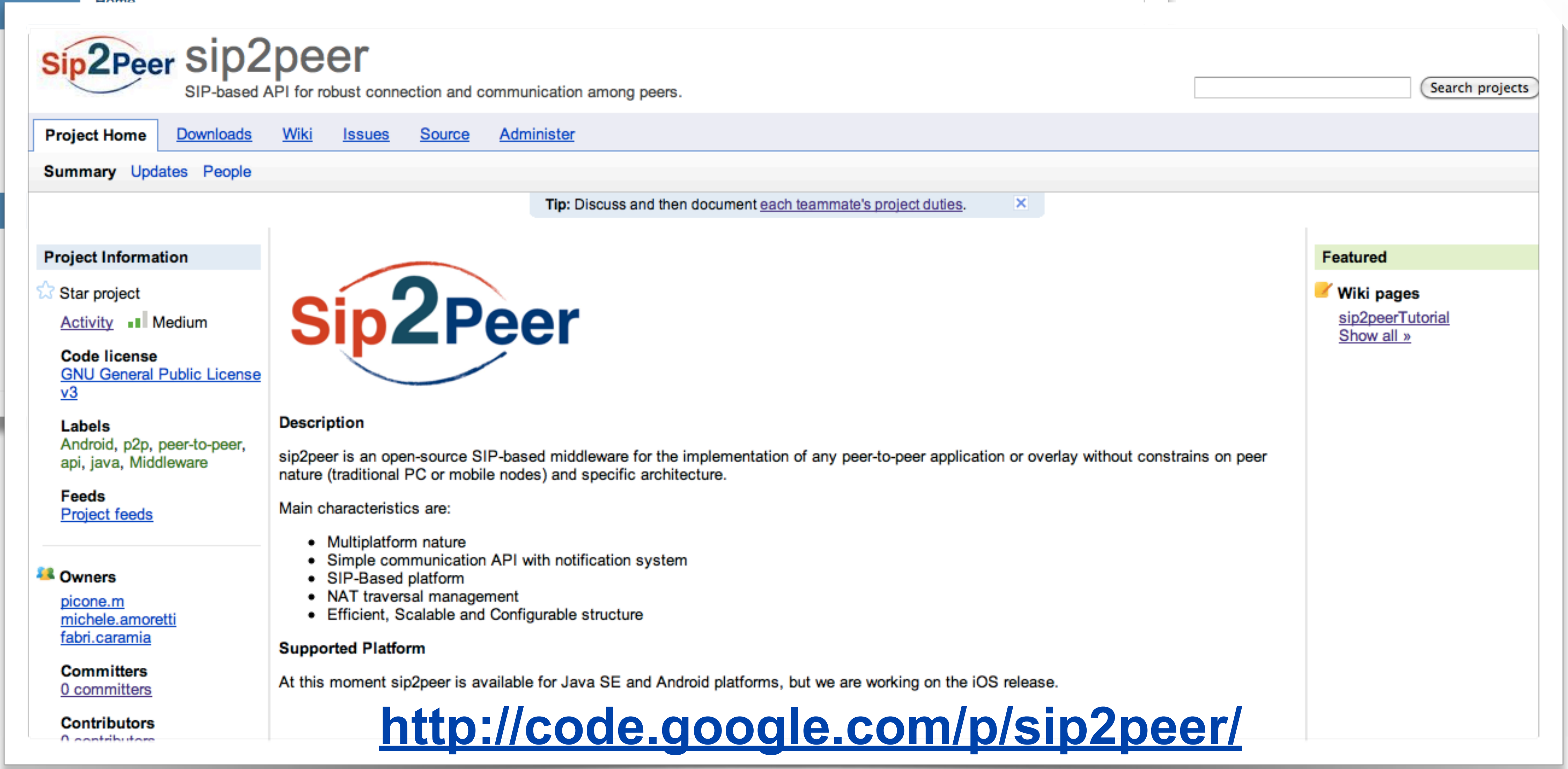
Contacts

http://dsg.ce.unipr.it/?q=node/41

Distributed Systems Group
Dip. di Ingegneria dell'Informazione
Università degli Studi di Parma

About us
- DSG Home
- People
- Research
- Publications
- Students

Projects
- DSG Projects
  - DEUS
  - DGT
  - P2P & Network Coding
  - PATROL
  - sip2peer



sip2peer
SIP-based API for robust connection and communication among peers.

Project Home | Downloads | Wiki | Issues | Source | Administer
Summary | Updates | People

Tip: Discuss and then document each teammate's project duties.

Project Information
Star project
Activity Medium
Code license
GNU General Public License v3
Labels
Android, p2p, peer-to-peer, api, java, Middleware
Feeds
Project feeds
Owners
picone.m
michele.amoretti
fabri.caramia
Committers
0 committers
Contributors

Description

sip2peer is an open-source SIP-based middleware for the implementation of any peer-to-peer application or overlay without constrains on peer nature (traditional PC or mobile nodes) and specific architecture.

Main characteristics are:

- Multiplatform nature
- Simple communication API with notification system
- SIP-Based platform
- NAT traversal management
- Efficient, Scalable and Configurable structure

Supported Platform

At this moment sip2peer is available for Java SE and Android platforms, but we are working on the iOS release.

Featured
Wiki pages
sip2peerTutorial
Show all »

http://code.google.com/p/sip2peer/