Freie Universität Berlin

Diploma Thesis
Collaborative Data Processing on Mobile Handsets
Jan Kettner

Examiner: Prof. Dr. Mesut Günes
Tutor: Georg Wittenburg, M. Sc.

04.10.2010

Overview

- Motivation
- Smartphone OS (Android)
- Peer-to-Peer Networks (Pastry)
- MobP2P Base System
- MobP2P Example Application
- Software Demonstration
- Future Work

Motivation

Thesis is combination of 2 developments: “Peer-to-Peer” and “mobile” based on patent application by Georg Wittenburg

Modern smartphones have capable processing units (Nexus One: 1GHZ CPU) and provide high speed internet access (HSDPA)

Mobile operators offer affordable mobile data plans → Devices can be always connected to the internet

Mobile devices can be used to create and store data (Music, Images).

Sharing that data is next logical step (Flickr)

But: Existing applications for mobile devices are built on client- server architecture. Peer-to-Peer technologies offer an efficient alternative

Goal: *Develop a software for a mobile platform that uses Peer-to-Peer technology for shared data processing*

Smartphone OS

Gartner predicts by 2013 more smartphones than pcs (1.82 billion)

Smartphones available since mid 90s (Nokia Communicator Series)

Smartphone market changed in 2007:

Introduction of the Apple iPhone (capacitive touchscreen, multitouch, used with fingers instead of stylus) and the Android platform

Today wide selection of smartphone operating systems available (iPhone OS, Windows Phone 7, Web OS, Meego, Bada, Symbian)

Most smarphone operating systems are based on the Linux kernel

User Interfaces controlled by touch gestures

In device distribution of applications

SDK for third party developers available

Programming contests (ADC)

Android

Motivation: radical increase in mobile search queries in 2007

Open Handset Alliance (Google, Intel, ARM, Samsung, HTC, T-Mobile etc. Currently 71 members) released Android platform in October 2008

Android is open source, based on the Linux 2.6 kernel using the Java based Dalvik VM. Dalvik VM is register based and optimized for low memory scenarios

No licensing fees for Android

Today 17.7 % marketshare of smartphone os worldwide

Android SDK, freely available for Windows, Mac and Linux OS

Android Developer Challenge (ADC 1, 2008, 1778 entries, 5M $)

Android Market: Currently 120.000 applications

Android Tools

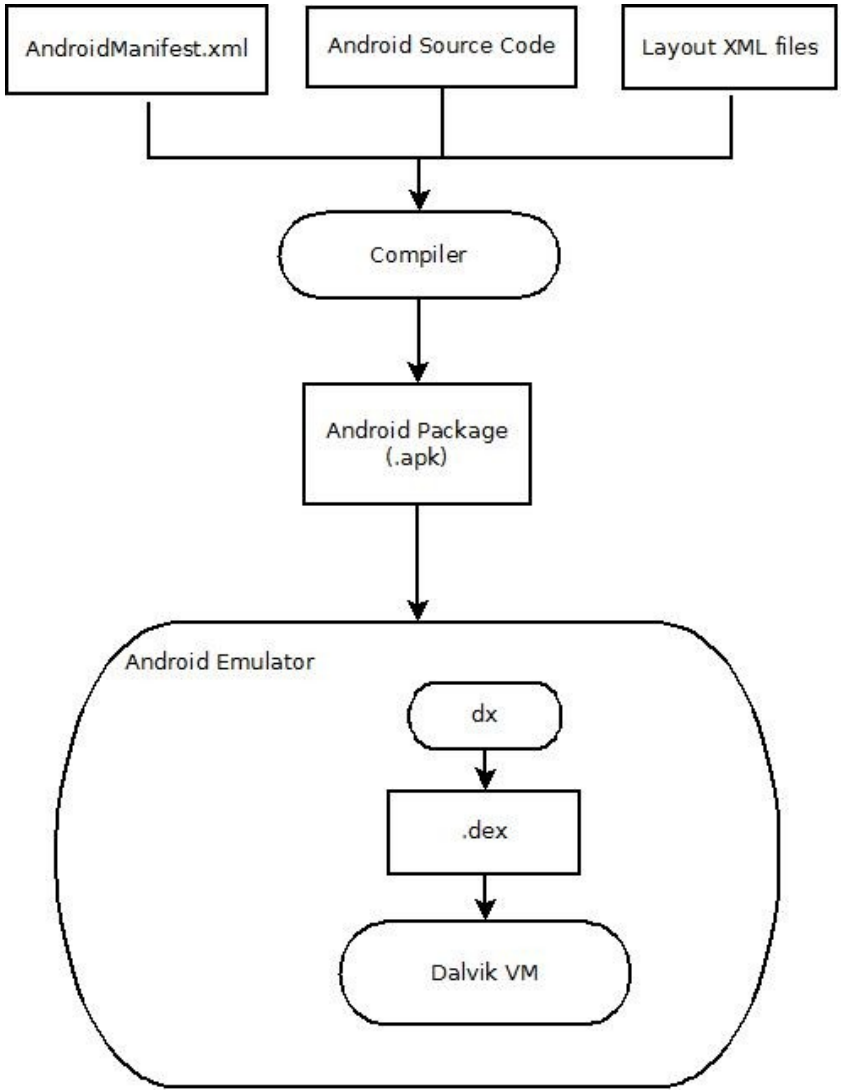
ADT – Eclipse plug-in for Android application development

Android applications consist of source code and XML data and are signed

Android Manifest declares application components & determines its rights

Compiler creates an Android Package

Deployed application packages are converted (optimization) into Dalvik executable files (.dex)



Android Application Components

Activity – User interface component. Applications may have multiple independent Activities drawn in their own windows. Activities have multiple entry points (no *main()* method)

Intent – Activates other application components. Directed and undirected Intents. Intent may transport data bundles.

Services – Services run indefinitely and independently in the background (e.g. waiting for a network event) and may activate Activities with Intents

Broadcast Receivers – Broadcast Receivers respond to broadcast announcements mostly originating in system code

Content Providers – Content Providers can be used to make application data available to other applications.

Peer-to-Peer

P2P technologies responsible for 45% to 82% of worldwide internet traffic. P2P mostly known for its use in file sharing application but there are many other possible applications (Diaspora, Skype, IM, IPTV)

Peer-to-Peer Systems are defined as a system of self-organizing equal and autonomous units, peers, that without using centralized services operate on an existing network with the goal of sharing resources

First generation (unstructured) Peer-to-Peer networks (Gnutella, Kazaa) made no implications on network structure. Nodes connected randomly. To send messages → Flooding. Solution were hybrid systems (Napster).

Second generation networks (Chord, CAN, Pastry) introduced DHT that partitioned the namespace of the network. Messages are forwarded to that entry in DHT corresponding to the target Node ID

Pastry

Pastry is a second generation Peer-to-Peer protocol. Basic Idea: Forward messages to that node sharing a longer prefix with the target node Id than the present node

To each node a 128 bit identifier is assigned (Node Id). Nodes maintain 3 types of state tables (Leaf Set, Routing Table, Neighborhood Set)

The Leaf Set holds those node Ids that are numerically closest. Neighborhood Set holds those node Ids that are closest according to the proximity metric (Authors propose GPS data or internet hop count).

Row n of Routing Table holds those Node Ids that share an n-digit prefix. Column m of RT holds those Node Ids where the n+1 digit is m

Pastry can route a message to the numerically closest node in $\lceil \log_{2^b} N \rceil$ Steps. Delivery is guaranteed unless $\lfloor |L|/2 \rfloor$ adjacent nodes fail simultaneously

Pastry – Node Join and Departure

Pastry JOIN:

New node sends JOIN message to known Pastry node
The known Pastry node routes the JOIN message
The message is routed to the numerically closest NodeId
Each node the message passes sends state table info to new node

Pastry Node Departure:

No hand-off procedure. Node failure must be detected by neighbors
Nodes must periodically contact nodes in Neighborhood Set
When failure detected node X contacts Neighborhood Set nodes
Neighborhood Set nodes send state tables to X
Node X repairs state table with new information

Pastry Routing Algorithm

1) Target NodeID ⊆ LeafSet → Forward to Closest NodeID

else

2) Check if appropriate entry ex. in Routing Table → Forward To

else

3) Forward to numerically closest node in all tables

BSP: 10233020, 10232100, 10233301

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

MobP2P Overview

Goal: *Develop a software for the Android platform based on Pastry for shared data processing*

Function: Users can create data items that are automatically synchronized on authorized remote devices

Design Goals

Reusability (Extensibility):

System consists of two parts: Base System and Example Application

Portability: fixed set of message types and structure to enable porting to multiple platforms

Redundancy: Data items are stored on external nodes, from where they can be retrieved by authorized nodes

MobP2P Base System Architecture

Base System

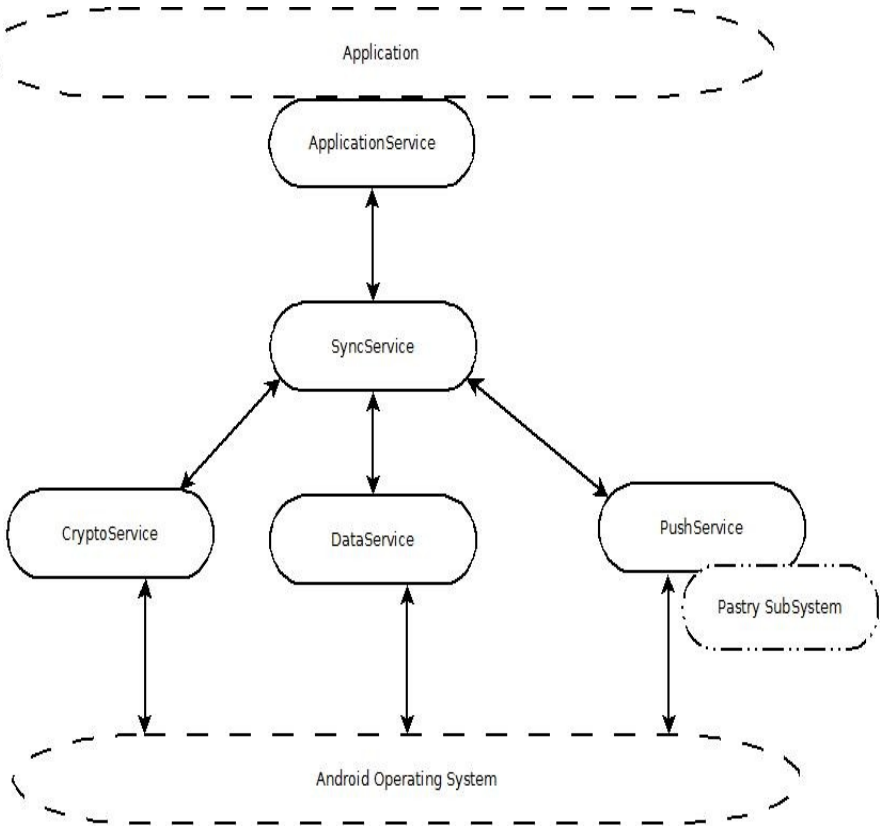
SyncService: Data item management

PushService: Creation and processing of Pastry and Application messages

Pastry SubSystem: Pastry implementation

DataService: Data item and Pastry data storage

CryptoService: Encryption and decryption of data items



Example Application

ApplicationService: Interface that enables access to the Base System

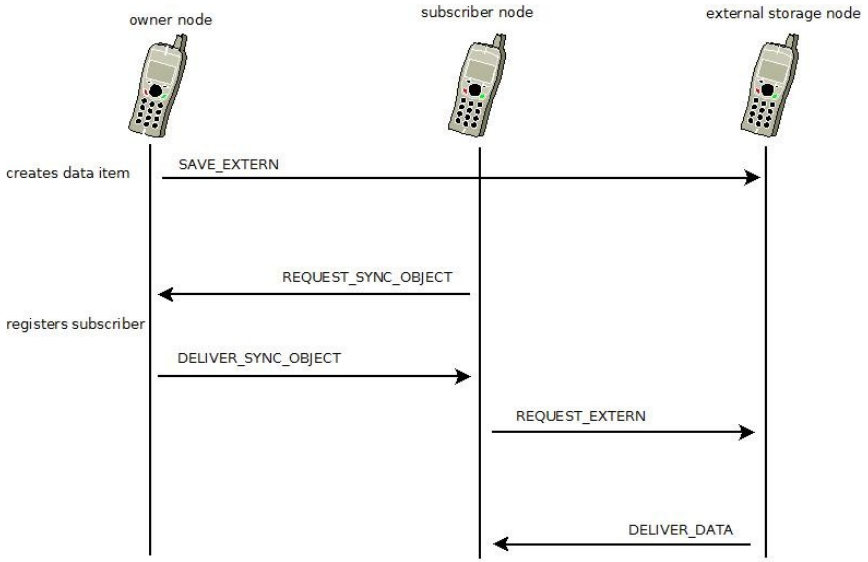
MobP2P Three Perspectives of Data Items

A data item can be seen from three perspectives:

SyncObject: Perspective of owner, read and write access, creates and distributes key

Remote Object: Perspective of subscriber, read access, holds key for decryption

ExternObject: Perspective of external storag node, cannot access encrypted data item,



MobP2P Message Format

Distinction between Pastry and application level messages. Pastry Level messages are handled by PushService.

Application level messages are wrapped in a Pastry level message. If a Pastry level message contains an application level message the message is passed on to the SyncService for further processing.

Pastry level messages:

JOIN, JOIN_RESPONSE, SEND_TABLES, APP_DATA

Application Level Messages:

REQUEST_SYNC_OBJECT, DELIVER_SYNC_OBJECT, DELIVER_DATA, SAVE_EXTERN, REQUEST_EXTERN, DELIVER_EXTERN

MobP2P Pastry Implementation

Testing on Android emulator caused extreme low node numbers

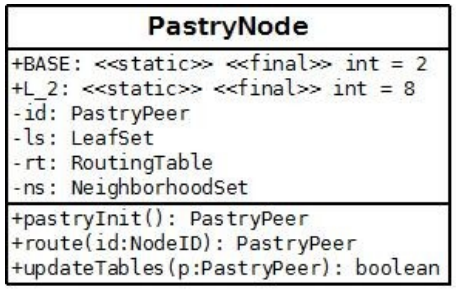
Pastry configuration values:

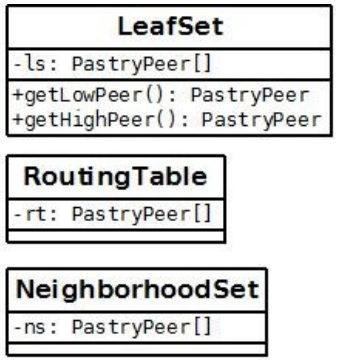
B(Base) = 2

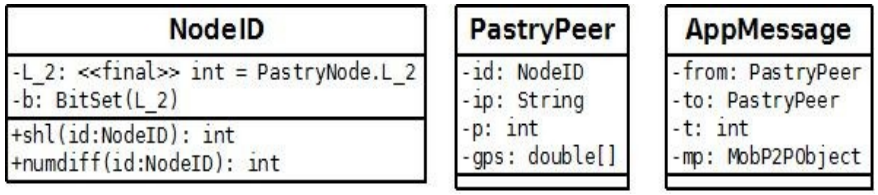
L/2 = 8

Numerical difference used always instead of proximity metric

To send a message an AppMessage instance is passed to the PushService.push() method







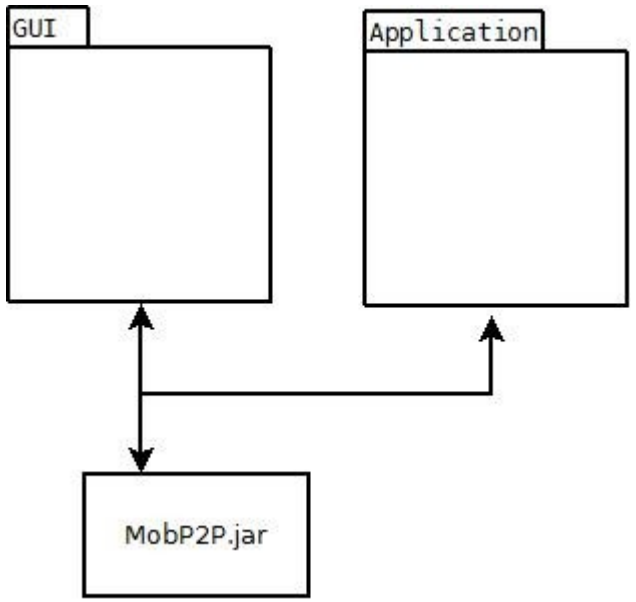
MobP2P Example application

PeerStrings Application designed to test the MobP2P Base System functionality

Main functionality is synchronization of String data

GUI package holds Activities

Application package holds Implementation of the Application Service Interface



MobP2P Example Application cont'd

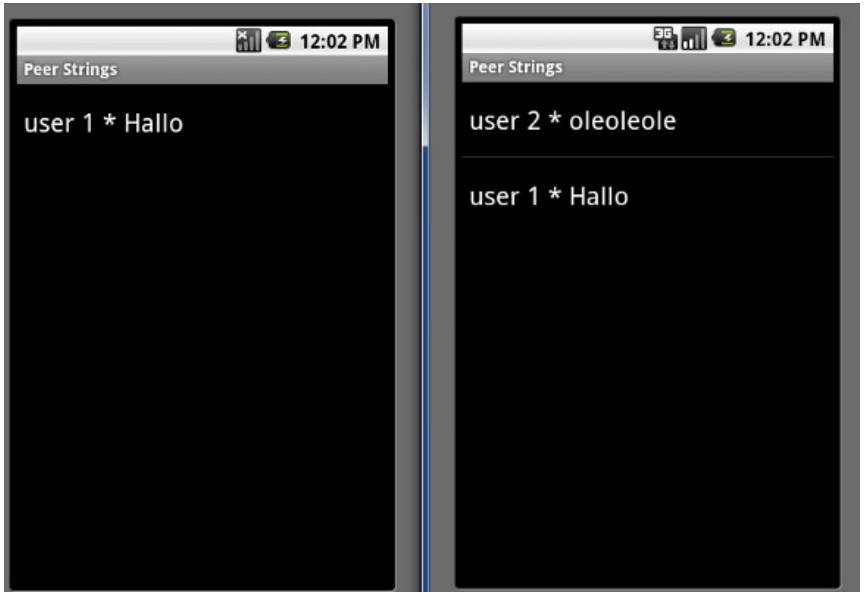
DisplayStrings is main Activity of the example application

Displays a list of String data items from user and those data items the user has subscribed to

Other Activities (EnterNode, EnterPastryPeer, EnterContact, EnterFriend) used to enter data

Possible future applications:

Shared Shopping List, Calendar Synchronization, Shared Notepad, etc.



Future Work

Implementation of proper proximity metric

Implementation of Neighborhood Service

Object Serialization

Implementation for various smartphone platforms