Application Development for Mobile and Ubiquitous Computing


Dr. Ing. Thomas Springer
Technische Universität Dresden
Chair of Computer Networks
Structure

- Android
- .Net Compact Framework
- JXTA
  - basic concepts of Peer-to-Peer
  - JXTA platform
Google’s initiative to create an open platform for mobile devices

Open Handset Alliance
- Established November 2007
- Business alliance of > 45 companies
  - Device manufacturers: Sony Ericsson, Motorola, HTC, ...
  - Mobile operators: Vodafone, T-Mobile, China Mobile, ...
  - Others like Software companies, semiconductors, etc.
- Aim: “Revolutionize the mobile industry“
- all based on „open“ solutions

Open character
- Open Source: Apache 2.0, but not everything is released (e.g. no Google Maps)
- bundled applications have same priority and rights as third-party apps (even the homescreen can be replaced)
What is Android?

- Software stack for mobile devices
  - an Operating System
  - a Middleware
  - a set of basic applications
- Android SDK
  - Developer Tools
  - Emulator
  - Sample Code
  - Android Library
- Developing Language
  - Java (managed code)
- Virtual Machine
  - Dalvik (GNU/Linux kernel)
Android Architecture

- Application Framework
  - allows reuse and exchange of components
- Programming in Java, but with special VM implementation (Dalvik VM)
- Complete development environment

- Media Libraries - based on PacketVideo's OpenCORE; playback and recording of many popular audio, video and image formats, (MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG)
- SQLite - lightweight relational database engine
- Google Maps support
- Integrated Browser - based on WebKit (open source)
- Optimized graphics libraries - 2D library, 3D library based on OpenGL 1.0
- Version 2.6
- As an abstraction layer between hard- and software
- Core system services
  - threading
  - low-level memory management
  - hardware drivers
  - power management
Dalvik Virtual Machine

- alternative Java implementation
  - no Sun certification
  - basically just the syntax of the progr. language is the same
  - Dalvik byte code
    o must be compiled for Dalvik VM
  - no full Java ME, no full Java SE
    o four major libraries 'lang', 'util', 'io', 'net' fully available
- Optimized for mobile computers
  - memory management
  - every application runs in its own process
  - optimized for many parallel VMs
Anatomy of an Android application

- Four building blocks
  - Activity
  - Broadcast Intent Receiver
  - Services
  - Content Providers
- used components have to be declared in the Android Manifest file
• **Activity:**
  - a single screen of the application
  - extends the Activity class
  - consists of user interface elements (views) that respond to events
  - may return a value to another activity
  - When a new screen opens, the previous is put onto a history stack.
  - Methods of activity reflect lifecycle
**Service:**
- background thread working independent from the UI
- Local and Remote Services
- Activities can connect through `bindService()`
- When connected, communication is done by an interface exposed by the service; the interface is based on the AIDL (Android Interface Definition Language).
Intent:
- Events/messages exchanged between Activities at application level
- "message objects" used to move from one activity to another
- Consists of an Action string, URI and payload
- Common action values are MAIN (to open an app), VIEW, PICK and EDIT
- Example: "VIEW" with web address opens the browser, "CALL" calls a provided number
- Intent Filter express ability of component to handle particular intent types

BroadcastReceiver:
- Broadcast intents represent events propagated by the system (e.g. battery low, screen off, boot completed)
- BroadcastReceiver is special intent filter for system messages
**ContentProvider:**
- Sharing of data between applications
  - Database or Streaming of binary data
- implements a standard set of methods for allowing other applications to store and retrieve data.

**Resources:**
- external files (that is, non-code files); used in code and compiled into application at build time
- number of different kinds of resource files, including XML, PNG, and JPEG files
Android Manifest

- AndroidManifest.xml necessary for every application
- Describes the application's elements and when they should be initialized or activated
- Includes a list of permissions the application is offering or needing (e.g. for access to network or contacts data); so on installation, the user can grant or deny these.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.my_domain.app.helloactivity">
    <application android:label="@string/app_name">
        <activity android:name=".HelloActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```
Android Development Environment – Eclipse Plugin

Dr. Thomas Springer

Android Summary

- Java-based application development
  - Optimized Virtual Machine
  - Support of Java SE 1.6
  - Cross-compilation of bytecode
- Innovative concept for application programming
  - Activities, services and intents
- Declaration of UI
- Lifecycle of processes controlled by runtime system
  - Applications can be stopped at any time
.NET COMPACT FRAMEWORK
Provided by Microsoft

Mobile version of .Net Framework
- tailored to resource limitations of mobile devices
- same programming model

Specific runtime environment
- adapted to limited memory, processing power and portability

Based on Common Language Infrastructure (CLI)
- standard for programming language and platform-independent applications
- binary compatible with standard .Net framework code
- supported languages: C# and Visual Basic .Net
- open to further languages which can be compiled to CLI-code

Particular layer for device platform and operating system independence
2 major building blocks

- Compact Framework Class Libraries (Common Language Intermediary)
  - object-oriented libraries organized in hierarchical name space
  - provides basic functionality and interfaces for XML processing, Web services, and for developing web-based applications
  - language independent, factored into a series of DLL files
  - only used libraries are integrated

- Common Language Runtime (CLR)
  - comparable with Java VM – runtime executes intermediary byte code
  - memory management, thread management, security model, exception handling
  - Common Type System
    - language-independent type system
    - enables interoperability of different programming languages and platforms
Microsoft .Net Compact Framework - Architecture

- App 1
- App 2
- App 3

- .Net Compact Framework
  - User Interface
  - Web Services
  - XML
  - Data
  - Base Classes
  - Execution Engine
  - Platform Adaptation Layer
  - Host Operating System

- Compact Framework Class Libraries
- Common Language Runtime (CLR)

- Managed Code
- Native Code
Microsoft .Net Compact Framework

- **Native Code**
  - provided in target processor native code
  - Execution Environment
    - Just-in-Time Compiler transforms CLI-code to processor specific code
  - Platform Adaptation Layer
    - enables independence from particular OS
    - maps OS APIs to .Net Compact Framework APIs
  - Operating System
    - .Net Compact Framework independent from Windows

- **Managed Code**
  - provided as language independent byte code
  - transformed to native code by Just-in-time compiler at runtime
  - Applications
  - Extension Libraries
  - Class Libraries
Common Language Runtime
- executed with all applications in one process – usage of shared memory space
- managed code isolated in application domains
- JIT compiler
  - transfers code to memory
  - transformation to native code – at granularity of methods and types
  - only used code blocks are transformed and cached
- Two compiler versions
  - IJIT – fast, generated generic code
  - SJIT – optimized for ARM processors
- garbage collector
  - based on Mark-and-Sweep algorithm
  - Defragmentation
Base Class Libraries

- comprise
  - core types, file I/O, sockets networking
  - reflection
  - globalization – localization supported
  - System.*, System.Collections.*
  - System.ComponentModel.*
  - System.IO.*
  - System.Security.*
  - System.Threading.*
  - ...

Dr. Thomas Springer
Two types of graphics libraries (Similar to Java ME)

- **high-level libraries**
  - subpart of standard .Net framework elements
    - Button, TextBox, Label, RadioButton, ComboBox, ListBox, TreeView
  - other implementation
  - additional libraries can be integrated
  - System.Windows.*

- **low-level libraries**
  - subpart of standard .Net framework functions
  - 2D graphics operations
  - Image, Icon, Bitmap and Graphics
  - System.Drawing.*
Multiple Protocol support
- System.Net.Sockets class abstracts from transport protocols
- TCP, UDP and HTTP supported
- Standard mechanisms for encryption and authentication
- Handling of IP-Addresses

Web Services well supported
- based on Visual Studio .Net features
  - parses WSDL documents
  - generates easy-to-use client proxy classes
  - System.Net.*

XML processing
- simple XML processing – XmlReader, XmlWriter
- maximum performance, noncached, forward only XML reading and writing
- XML DOM – XmlDocument class
- in-memory tree for more complex operations on document
- System.Xml.*
Persistent Memory

- File Access using System.IO.* classes and operations

- Active Data Objects (ADO.Net)
  - classes for management of relational data sets
  - DataSets in memory can be manipulated
  - DataAdapters allow access two different types of data sources

- Microsoft SQL Server CE
  - lightweight version of SQL Server for mobile devices
  - uses 1MB up to 3MB memory
  - SQL Server in backend as master DB
  - Data replicated on lightweight DB
  - ActiveSync component coordinates synchronization
Development Environment

- Visual Studio .Net
- Different programming languages
- C#, Visual Basic .Net
PEER-TO-PEER AND JXTA
Peer-to-Peer (P2P):

- A Peer-to-Peer system is a self-organizing system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services. [Steinmetz, Wehrle: Peer-to-Peer Systems and Applications]

  - Shift in paradigm from coordination to cooperation
  - Client/Server: asymmetric functionality
  - Peers: symmetric functionality (equal peers)
Why Peer-to-Peer for Mobile Computing?

- **Decentralized approach**
  - supports ad hoc infrastructures
  - dynamic discovery of peers

- **Scalability**
  - supports potential high number of devices and users typical for mobile computing environments
  - supports sharing of limited device resources
  - supports balancing of bandwidth usage (e.g. for content sharing)

- **Self-managing**
  - supports arrivals, departures and failures of mobile devices
Mobile Peer-to-Peer Social Networking

- **Peerhood**
  - Middleware for P2P communication based on different wireless comm. Technologies (Bluetooth, WLAN)
  - Dynamic device/user detection

- **P2P - Social Networking application**
  - Grouping of users based on similar profiles
  - Chatting, content-sharing within groups
- P2P sharing of audio information and music
Probably the most important aspect of P2P networks
Where to store/manage, and how to find a certain data item/resource in a distributed system without any centralized control or coordination?

The Lookup Problem

Dr. Thomas Springer
Classification of Peer-to-Peer Networks (P2P):

- **unstructured P2P**
  - centralized P2P
    - server for coordination and search
    - example: Napster
  - pure P2P
    - no centralized coordination
    - example: Gnutella, Freenet
  - hybrid P2P
    - dynamic central entities, some peers act as coordinator
    - examples: Gnutella2, BitTorrent
- **structured P2P**
  - Distributed Hash-Table
    - "fixed" connections in overlay
    - examples: Chord, CAN, Pastry, Tapestry

Centralized P2P – Example Napster

- Connection to Napster Server: publish IP address and files, which can be shared
- Request server for wanted object. Server returns a list of peers, which has object.
- Selection of a peer (on the basis of estimated download time, available bandwidth resp. after pinging P2P-connection to peer) and download
Problems of a Central Directory

- **Single-Point-of-Failure**
  - If the central directory server crashes, the entire P2P application crashes.

- **Performance Bottleneck**
  - In large P2P systems with hundreds of thousands connected users, the central directory server has to cope huge amounts of data and thousands of queries per second.

- **Centralized P2P application is only partially decentralized.**
  - Traffic decentralized, management still centralized
Pure P2P – Example Gnutella

- No central entities at all
- Content-location directory distributed over the peers themselves
- Hand-shake with an already known member (host cache) of the Gnutella network (preconfigured list in client software)

Advantages:
- No single-point-of-failure
- No performance bottleneck
- No censorship possible

Dr. Thomas Springer
Query Flooding

- Discovering new peers in Gnutella
  - Send a broadcast ping
  - Active peers answer with a pong

- Locating specific content in Gnutella
  - Iterative search
  - Send a Query msg to all neighbours
    IF neighbour N1 owns content
      THEN answer with QueryHit msg
    ELSE pass on Query msg to next peers
  - Query msg contain TTL (max. 10 hops)
  - Swarm download if more than one peer owns the requested content
The Problem with Query Flooding

- Big overhead, few results
- \( \rightarrow \) Scalability

Kelsey Anderson: *Analysis of the Traffic on the Gnutella Network*. University of California, San Diego, CSE222 Final Project, March 2001
Challenge: How to combine
- Efficiency of centralized approach with
- Robustness of decentralized approach

Solution:
- Transparent separation between
  - Super Nodes (SN)
    - build a “high-speed backbone” for the P2P network
    - Earn or loose their privileges due to their system resources
    - Keep track of all content offered by related ordinary nodes
  - Ordinary Nodes (ON)

Content Hash = Improved identifier for content
- Seamless retrieval from different peers
Supernodes know, and communicate with each other

Each supernode is related to approx. 200-500 ordinary one

All peers are equal, but some peers are more equal than others
Managing the KaZaA Overlay Network

- **Joining the P2P network**
  - User gets list of super nodes when downloading the software
  - Searching the list for operating super node, connection establishment
  - Receiving an actual list of super nodes, ping to 5 of this nodes, choose super node with lowest RTT as superior node
  - When super nodes leaves, peer gets new list and chooses new one

- **Locating and retrieving specific content**
  - Peer sends search request to super node
    - Returns list of results
    - else send up request to neighbouring super nodes
  - Each query is only directed to a subset of all super nodes
  - Parallel downloads are possible due to unambiguous content hashes
Unstructured Peer-to-Peer:

- Two phases: service search and service usage
- Flooding in pure P2P networks

[Diagrams showing different architectures: centralized, hybrid, pure]

Dr. Thomas Springer
In unstructured P2P networks
- Content is duplicated randomly on peering nodes
- Centralized approach: Relocation of content easy but does not scale well
- Fully decentralized approach: Relocation of content easy but wasteful
- There is no guarantee for a result when searching since the lifetime of request messages is restricted to a limited number of hops
- Central servers suffer from a linear complexity for storage
- Flooding-based require costly breadth-first search which leads to scalability problems in terms of communication overhead

In structured P2P networks
- Content location follows specific patterns -> no flooding needed
- Distributed Hash Tables (DHT) = central mechanism for indexing and searching content
- Afford guaranties when searching for an object

Examples:
- Chord, Pastry, Kademlia (part of BitTorrent and eMule)
- mainly differ in routing
Main Approach - Distributed Hash Tables:

- logic structuring of P2P networks to overcome tremendous bandwidth consumption for flood-based algorithms
- requested characteristics
  - load balancing: equally distributed
  - scalable: minimum amount of messages
  - robust: adding and removing peers
  - no global view to network should be required

Distributed Application

Distributed Hash Table
(CAN, Chord, Pastry, Tapestry, ...)

Node 1  Node 2  Node 3  ....  Node N

P2P PLATFORM: JXTA
Overview

- JXTA (Juxtapose)
  - Open Source Project founded by Sun in 2001
  - open, standardized platform for the implementation of peer-to-peer applications
  - set of open, generalized peer-to-peer protocols
  - Implementations available for Java SE, Java ME, C, C++, and C#

- Goals
  - Platform independence (programming language, communication protocols, deployment platform)
  - Ubiquity (heterogeneous device types and hardware)
  - Interoperability
Architecture

- **Core**
  - essential primitives and building blocks for P2P applications
  - discovery, transport, peer creation, peer grouping, security primitives

- **Services**
  - includes network services
  - not necessary but desireable for P2P
  - searching and indexing, directory, storage system protocol translation, resource aggregation, authentication and public key infrastructure

- **Applications**
  - e.g. instant messaging, document and resource sharing, content management and delivery, etc.

JXTA Virtual Network
Core - Basic Abstractions

- **Peers**
  - Represents autonomous, independent unit
  - Any networked device that implements one or more JXTA protocols
  - Includes sensors, phones, PDAs, Laptops, PCs
  - Self-organized into **Peer Groups**
  - Offer services and resources based on **Advertisments**
  - Communicate based on **Pipes and Messages**

- **Peer endpoints**
  - Peer publishes one or more network interfaces
  - Used to establish connections between peers
Peer Groups

- Groups for boundaries for search and resource sharing
  - groups can form hierarchies
  - each group has single parent
    - search is performed within a group
    - advertisements are published in the group and the parent group
- Forming a secure environment
  - by enforcement of policies for membership
  - e.g. for limited access to peer resources
Communication between peers is based on pipes
- Asynchronous, unidirectional, non-reliable
- Can transport any type of message and data
- Pipe endpoints (input, output) can be dynamically bound to peer endpoints at runtime
Several types of pipes

- **Point-to-point**
  - connects two pipe endpoints (unicast)

- **Propagate**
  - connects one output pipe to multiple input pipes (multicast)
  - all output and input pipes have to belong to the same peer group

- **Secure Unicast Pipe**
  - special point-to-point pipe
  - provides a reliable and secure communication channel
Messages

- Basic unit of data exchanged between peers
- Can be send and received by Pipes or Endpoints
- Binary encoding used

- Consists of ordered sequence of elements
  - set of name/value pairs
  - value can be of arbitrary type
- Messages can be encrypted and digitally signed
- describe offerings of peer resources
- based on XML
- identified by Advertisement ID
  - identification of resources independent from physical address

```xml
<?xml version="1.0"?>

<!DOCTYPE jxta:PipeAdvertisement>

<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
  <Id>
    urn:jxta:uuid-
    59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504
  </Id>
  <Type>
    JxtaUnicast
  </Type>
  <Name>
    TestPipe.end1
  </Name>
</jxta:PipeAdvertisement>
```
Super Peers

- **Edge Peers**
  - Peer ID
  - Peer Endpoints

- **Rendezvous Super Peers (RP)**
  - functions can be provided by any peer
  - provide minimal infrastructure for resource discovery
  - peer advertisements are sent to rendezvous peer
  - rendezvous peers exchange advertisements
  - additionally replication of advertisements of neighbor RPs

  - Search based on RPs
  - If advertisement found, search is mediated to offering peer
  - if no offer found, request is sent to further known RPs
  - bootstrapping mechanism to discover RPs
JXTA Rendezvous Super Peers

Super-Peers

Rdv 1

Index

Peer A

Adv

Propagate (2)

Forward (3)

Peer B

Adv

Rdv 2

Index

Respond (4)

Query (1)

JXTA Virtual Network

Physical Network

Peer 1

Firewall

NAT

Peer 2

Adv

Dr. Thomas Springer

Peer groups provide a set of core services

- **Discovery Service**
  - used by members to search for group resources (e.g. peers, pipes, services)
- **Membership Service**
  - for controlling the join of new members
  - accept or reject of join request by voting or decision of designated group representative
- **Access Service**
  - for checking permissions to access services of peer group members
- **Pipe Service**
  - management of pipe connections between peers in the group
- **Monitoring Service**
  - allows monitoring of peers in the same group
Relay Super Peers

- Relay Super Peers
  - Proxy for peers with no bidirectional network connection (e.g. peers behind NAT or firewalls)
  - Relays store messages for peers
  - Peers can then poll messages

- Use of Relays is transparent for application
  - Peer leases connection to Relay for certain time frame
    - After timeout of lease Relay can be changed

- No static relation between peer and Relay (relay can be transparently changed)
JXTA Relay Super Peers

Diagram showing the structure of JXTA Relay Super Peers.
JXTA Protocols

- Two types of protocols
  - Core protocols - required
  - Standard Services – optional

- Core Protocols
  - Endpoint routing protocol – management or routes between peers
  - Peer resolver protocol – peer discovery based on resolver
- Mobile devices always represented by JXTA relay
  - acts as a proxy on behalf of its mobile peers
  - interacts with other peers and pipes, and uses group services
  - routes (relays) messages to or from mobile peers.
  - translates messages from JXTA XML to JXTA Binary Message format, and vice versa
Porting of JXME protocol to Android platform
Decentralized approach supports
- Spontaneous connections between mobile devices
- Distributed network load between wireless links
- Allows sharing of resources of mobile devices
- Required effort for coordination and management of topology changes

JXTA
- Superpeer-based middleware for P2P application
- Basic abstractions (Peer, Pipe, Advertisement)
- Core protocols and services
- JXME for integrating mobile devices
- PeerDroid for Android devices
- Android
  - www.android.com
  - http://code.google.com/android
- .Net Compact Framework:
  - msdn.microsoft.com/de-de/library/cc656764.aspx
- JXTA
  - https://jxta.dev.java.net/