Mobile Platforms and Middleware

Mobile Communication and Mobile Computing
Prof. Dr. Alexander Schill
http://www.rn.inf.tu-dresden.de
## Devices and Operating Systems

<table>
<thead>
<tr>
<th>OS</th>
<th>based on</th>
<th>Flash</th>
<th>JVM</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Phone</td>
<td>Windows</td>
<td>yes (restricted)</td>
<td>yes</td>
<td>closed</td>
</tr>
<tr>
<td>Blackberry OS</td>
<td>Proprietary BB OS</td>
<td>yes</td>
<td>yes</td>
<td>closed</td>
</tr>
<tr>
<td>iOS</td>
<td>Mach/BSD Kernel</td>
<td>no (Cocoa Touch)</td>
<td></td>
<td>closed</td>
</tr>
<tr>
<td>Android</td>
<td>Linux Kernel</td>
<td>no</td>
<td>yes (modified)</td>
<td>open</td>
</tr>
</tbody>
</table>
What is Android?

- Open source software stack for mobile devices
  - an Operating System
  - a Middleware
  - a set of basic applications
- Android Software Development Kit
  - Developer Tools
  - Emulator
  - Sample Code
  - Android Library
- Development Language
  - Java
- Virtual Machine
  - Dalvik (GNU/Linux kernel)
Android Architecture

- Application Framework (allows reuse and exchange of components)
- Programming in Java, 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
Linux Kernel:
- 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 official certification
  - basically just the syntax of the progr. language is the same
  - Dalvik byte code (must be compiled for Dalvik VM)
  - no full Java ME, no full Java SE (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
### Building Blocks - Activities

- **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
Example: Activities

- src(loopplayer.rn.mc)/LoopPlayerActivity.java
  - Extends Activity class provided by Android
  - Refers to layout definition in res/layout/main.xml

```java
public class LoopPlayerActivity extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        checkIfServiceIsRunning();
        ...
    }

    ...
}
```

Save previous Activity to stack

User interface element (view)

Connect to Service
Building Blocks - Services

- **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).
Example: Services

In Service Class:

```
public class LoopPlayerService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        return binder;
    }
    @Override
    public void onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "starting service", Toast.LENGTH_LONG).show();
        return START_NOT_STICKY;
    }
}
```

In the Activity Class:

```
private void checkIfServiceIsRunning() {
    Intent intent = new Intent(this, MyService);
    if (LoopPlayerService.isRunning()) {
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    } else {
        startService(intent);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }
}
```
Building Blocks - Intents

- **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 parameter data
  - Common action values are MAIN (to open an app), VIEW (to activate the browser), and CALL (to call a phone number)
  - Intent Filter expresses 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
Example: Intents

```java
public class LoopPlayerActivity extends Activity {

public void onCreate(Bundle savedInstanceState) {
    ...
    final Button browseBtn = (Button) findViewById(R.id.browseBtn);
    browseBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent i = new Intent(v.getContext, BrowseActivity.class);
            i.putExtra("FilePath", filePath.toString);
            startActivityForResult(i, Constants.SELECT_AUDIO_REQUEST_CODE);
        }
    });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == Constants.SELECT_AUDIO_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            setFileUri((Uri)data.getExtras().get("fileUri"));
        } else {
            Toast.makeText(getApplicationContext(), Constants.ERROR, 45).show();
        }
    }
}}
```

Start Activity B via Intent and wait for results

Evaluate received results from Activity B
Further Building Blocks

- **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="loopplayer.rn.mc">
    <application android:label="@string/app_name">
        <activity android:name=".LoopPlayerActivity">
            <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
Android Summary

- Java-based application development
  - Optimized Virtual Machine
  - Cross-compilation of bytecode

- Innovative concept for application programming
  - Activities, services and intents

- Lifecycle of processes controlled by runtime system
  - Applications can be stopped or restored at any time
Mobile Computing: Further Problems

- decoupling/re-connection → emulation / queuing of services (example: WebSphere MQ)
- dynamic change of Quality-of-Service-properties → adaptation and selective message download (example: Mobile eMail)
- Disconnected operations in file systems and databases
Message Queuing: WebSphere MQ
Example for decoupled communication

- dynamic coupling between applications and local Queues via logon/logoff; therefore suited for mobile scenarios

- basic concepts: Messages, Queues with Queue Manager, and Message Channels between them

- use of Queues for transmission or receiving; mixed use is also possible

- Internet Gateway, C++- and Java-Support

- support of essential operating system platforms

- support of mobile devices with WebSphere MQ Everyplace
1. Establish connection to queue manager (local or remote) -> MQCONN
   • Authentication performed during connection
2. Open a particular queue object -> MQOPEN
   • Authorization is checked
3. Application passes message to queue (=MQPUT)
4. Message is forwarded to machine 2 via the (unidirectional) message channel or is queued in disconnected mode (disconnected operation)
5. Server application eventually gets message via MQGET command
6. Reply (optional) follow via separate queues and channels
Load balancing (selective delivery) or Parallel processing (replicated delivery)

Access to Server via multiple Clients

Queue, with optional support of message priorities
Message Queuing: Assessment

**Advantages**

+ simple manageability
+ robust message delivery
+ flexible application fields (for instance load balancing, parallelization, batch-transmission of branch data etc.)
+ loose coupling of programs for Mobile Computing

**Disadvantages**

- interaction model is different than with procedures/method invocations
- only key features standardized, many proprietary extensions exist
Mobile eMail: Characteristics

- support for POP3, IMAP, SMTP, SSL, HTML rendering
- multiple accounts and unlimited message sizes
- automatic mail fetching, filtering and synchronization
- handwritten note-taking software and speech recognition
- disconnected operation mode often explicitly supported (queuing)
- dedicated apps for webmail support (e.g. gmail, yahoo mail)
Mobile File System Example: CODA

- distributed file system, which offers robust and transparent access to data also in the case of server shut-down or network failure

- based on AFS (Andrew File System, distributed file system in UNIX-environment)

- model of “Disconnected Operations”: Client keeps Read- and Write-access on the data via a local buffer (Cache) also during temporary disconnection from network

- with re-connection system forwards changes and recognizes potential conflicts

- available for different operating systems (for instance Linux, Solaris, Windows)
CODA system model

Replicated Server:
High availability

Network communication
at file open and close

Disconnected Client:
local data access
on Cache

Client
("Whole-File-Caching")
Consistency Properties (CODA)

- **Callback**
  - logic reference from server to the active client, used for immediate information about file changes by other clients
  - after connection failures the file in client cache remains valid until timeout termination (as a rule several minutes)

- **reduced consistency:**
  - Update/Update-conflict: parallel updates of same file
  - Delete/Update-conflict: parallel erasure / update of same file
  - Name/Name-conflict: generating of two files with same name

- conflict processing explicitly in interactive form based on file update log of modification operations, however low conflict probability
- “Cache-Misses”: searched file is not in the Client-Cache
  - processing failure in disconnected status
- priority list of important files per user
  - highest priority is always kept in the cache (for instance system programs, user profiles, address files etc.)
  - other priorities: exchange strategies correspondent to importance (e.g. least recently used)
  - dynamically generated files referenced via list of essential operations (for instance actual test protocol “<file>.tst” etc.)
- similar approaches have been integrated into platforms such as Windows for file replication


Generalization: Concurrency Control

- Exclusive read/write locks too restrictive
- Therefore, extended lock compatibility with explicit notifications

<table>
<thead>
<tr>
<th>1.</th>
<th>2.</th>
<th>dirty Read</th>
<th>read</th>
<th>write</th>
<th>checkout Read</th>
<th>checkout Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirty Read</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
</tr>
<tr>
<td>read</td>
<td>ok</td>
<td>ok</td>
<td>conflict</td>
<td>ok</td>
<td>conflict</td>
<td></td>
</tr>
<tr>
<td>write</td>
<td>ok</td>
<td>conflict</td>
<td>conflict</td>
<td>conflict</td>
<td>conflict</td>
<td></td>
</tr>
<tr>
<td>checkout Read</td>
<td>ok</td>
<td>ok</td>
<td>notific.</td>
<td>ok</td>
<td>notific.</td>
<td></td>
</tr>
<tr>
<td>checkout Write</td>
<td>ok</td>
<td>notific.</td>
<td>notific.</td>
<td>notific.</td>
<td>notific.</td>
<td></td>
</tr>
</tbody>
</table>

- dirty Read: access to potentially inconsistent version
- checkout Read: long-term read access (disconnected mode) with notification of other requests
- checkout Write: long-term write access, also with notification
Concurrency Control: Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>lock (read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grant lock (read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lock (checkoutWrite)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>free (read)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grant lock (checkoutWrite)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lock (dirtyRead)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grant lock (dirtyRead)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lock (write)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>notify</td>
</tr>
<tr>
<td></td>
<td></td>
<td>free (checkoutWrite)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grant lock (write)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>free (dirtyRead)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>free (write)</td>
</tr>
</tbody>
</table>

LockManager
Mobile Databases: Principle

Mobile Devices
- Small relational database
- Subset of backend tuples (selection, projection, join)

Synchronization Server

Backend
- Full relational database
- Query and transaction interfaces

Product examples:
- IBM DB2 Everyplace
- Oracle Lite
- Sybase SQL Anywhere

IBM DB2
Oracle
Sybase
other DBMS (via JDBC)
Example: IBM DB2 Everyplace

Architecture: Middleware system with replication server

Figure according to „Mobile Datenbanken und Informationssysteme“
Hagen Höpfner, Can Türker, Birgitta König-Ries
Example: IBM DB2 Everyplace

- **DB2 Everyplace:**
  - Lightweight database system for mobile and embedded devices
  - supports subset of SQL language
    - creation and manipulation of tables
    - not supported: stored procedures, user defined functions, trigger, views, sub-queries

- **Sync Client**
  - triggers synchronisation
  - Communication with Sync Server
  - Synchronization based on database tables

- **Sync Server**
  - User authentication
  - manages synchronisation between DB2 Everyplace and Master database
  - propagates all changes in DB2 Everyplace to Master DB and updates in Master database to DB2 Everyplace
Steps for synchronizing local changes with master database

1. After initiation Sync Client sends request to Sync Server

2. Sync Server authenticates request (request contains login and password of user), if successful request is enqueued, client receives ack

3. Changes are sent to Sync Server and stored in Staging table, enables continuous work at client

4. Changes in Staging table performed on mirror database, conflict detection and resolution

5. Changes in mirror table are logged in change table, log is used to propagate changes on master database
Steps for synchronizing local changes with master DB

1. Sync request
2. Admin Control DB
3. Write updates
4. Copy updates, resolve conflicts
5. Capture

Mobile Client

Middleware

Database Server
Detection based on version numbers

- For each table row version number maintained by Sync Server and Sync Client
- Differing version numbers mark conflicts

Conflict Handling

- Default:
  - changes rejected, transaction rolled back, logging of conflict
  - Mobile database updated with data from mirror database
- User-based conflict resolution
  - based on program for log search
  - application-specific conflict resolution implementable
sql-statement:
SELECT name, street FROM hotel
WHERE country="Norway"
and city="Hammerfest"
and numberOfStars>2

Result:
Hotell Skytterhuset,
Skytterveien 24
Some further readings

- Android:  
  code.google.com/android

- CODA file system:  
  www.coda.cs.cmu.edu

- IBM DB2 Everyplace:  
  www.ibm.com/software/data/db2/everyplace

- WebSphere MQ  
  www.ibm.com/software/integration/wmq/