Application Development for Mobile and Ubiquitous Computing

4. Disconnected Operations

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

Application Development

Mobile Business Applications

Cross-Platform Development

Mobile Web Applications

Android

iOS

Mobile Middleware

Location-based Services

Disconnected Operations

Communication Mechanisms

Energy Awareness

Enabling Technologies and Challenges
Motivating Example

Social Fitness App
- Wristband/Smart Watch + Smartphone/Tablet App
- Server component for data storage and user management

Functionality
- Automatic detection and tracking of sports activities
- Recording of activity states and content (images, videos, track record, curve with pulse, etc.)
- Activity Timeline - Posting own activities and see others' activities in an integrated timeline with text, images, videos, etc.
- Management of training schedule and planning of training activities
- Managing competitions with ranking
Social Fitness App - Architecture

User and Team data
(workout, challenges, etc.)

App

App

App

App

Server
Social Fitness App – Offline Challenge

- Retrieving timeline entries

- App gets blocked
- Showing outdated timeline entries from cache
Social Fitness App – Offline Challenge

- **Uploading local workout data**

- **Storing data locally until connection is available**

- **User might think that data is sent or activity is performed**

Dr. Thomas Springer
Conflicting updates for team activity

- Local changes on outdated copies
- Conflicting changes due to concurrent updates

Dr. Thomas Springer
Application Development - 4. Disconnected Operations
Motivation

- **Challenge:** disconnecteds, i.e. longer phases of unreachability
  - foreseeable (due to energy saving, communication costs, location changes)
  - unforeseeable (uncovered regions, unavailability of servers, network congestions)

- **Goal:**
  - availability of application functionality and data also during phases of disconnection
    - Logic and data needs to be locally available
  - transparency for user and developer:
    - user perspective: application usage without interruptions
    - developer perspective: hiding disconnections from applications
Lecture Overview

- **Disconnected Operations**
  - Coda filesystem
  - Mobile Database Systems
  - Generalization of concepts

- **Weakly Connected Operations**
  - Concepts
  - Weakly-connection support in Coda

- **Offline-first Principle**
Distributed file system with disconnection support
- Goal: high availability and scalability
Disconnection Handling in Coda Cache Manager

Noble, Satyanarayanan (Carnegie Mellon University)

Hoarding

- hoarding database, prefetching

Emulation

- server emulation
- files from cache
- logging of updates

Reintegration

- reply of logs
- automated conflict resolution
- application specific conflict resolver

Disconnection

- logical reconnection
- physical reconnection

Dr. Thomas Springer
- **Operation while connected**
  - direct access to remote file servers
  - client-side **caching of whole files**
  - write-through changes

- **Hoarding**
  - Prefetching of files for offline availability
  
  - *hoard profile: script for specifying a set of files to be cached*
    - indication of their importance based on *hoard priority*
    - tool support for creating hoard profiles
Sample Hoard Profile

# Personal files
a /coda/usr/jjk d+
a /coda/usr/jjk/papers 100:d+
a /coda/usr/jjk/papers/sosp 1000:d+

# System files
a /usr/bin 100:d+
a /usr/etc 100:d+
a /usr/include 100:d+
a /usr/lib 100:d+
a /usr/local/gnu d+
a /usr/local/rocs d+
a /usr/ucb d+

# X11 files
# (from X11 maintainer)
a /usr/X11/bin/X
a /usr/X11/bin/Xvga
a /usr/X11/bin/mwm
a /usr/X11/bin/startx
a /usr/X11/bin/xclock
a /usr/X11/bin/xinit
a /usr/X11/bin/xterm
a /usr/X11/include/X11/bitmaps c+
a /usr/X11/lib/app-defaults d+
a /usr/X11/lib/fonts/misc c+
a /usr/X11/lib/system.mwmrc

(a)  (b)

# Venus source files
# (shared among Coda developers)
a /coda/project/coda/src/venus 100:c+
a /coda/project/coda/include 100:c+
a /coda/project/coda/lib c+

(c)

+[c|d] c … immediate children, d… all descendants, + … current and future files
-priorities, higher value means higher priority
Prioritized caching algorithm
- balance between
  - short-term need of caching for performance and
  - long-term need of caching for availability (hoarding)
- Least recently used scheme (LRU) for cache management
- combined with per device prioritized file list (hoard database)
- priority (hoard priority, recent usage)

Hoard walking - periodic cache updates or on request
- first phase
  - update file sets (e.g. new files in folder created by other clients)
- second phase
  - re-evaluation of all priorities in cache and hoard database
  - update cached files that were changed remotely
- Operation while disconnected
  - No access to remote file servers
  - Operation on **locally cached files**
  - Error message for access to uncached files

- Emulation
  - virtual file system emulates file server operations
  - retrieves files from Venus cache manager
  - changes written to local copy
  - local file changes immediately visible
  - versioning of files for conflict detection
  - Logging of local changes for later reply to server

  - Local modification log (change modify log, cml)
    - modification record log entry contains copy of the performed file operation and the version state of all involved files
• Local Modification Log (change modify log, cml)
  • Contains list of modification records, i.e. locally performed modifications
  • each modification record contains copy of the performed file operation and the version state of all involved files

• Optimization of modification log
  • any operation which overwrites the effects of previous operations cancel corresponding record logs (e.g. delete cancels create and write operations)
    • reduce required storage for change modify log
    • increases reintegration performance
Operation when reconnected
- again direct access to remote file servers
- **Log of local changes**
- Changed and outdated files in cache

Reintegration
- logged actions sent to server and replayed
- conflict detection based on version comparison

- conflicts
  - file changed in parallel at client and server
  - same name for a newly created directory
  - parallel change of directory attributes at client and server
Reintegration

- automatic conflict resolution
  - generic conflict resolution
  - application specific conflict resolution
    - application specific conflict resolvers provided by application developer (includes knowledge about file content)

- tool for manual conflict resolution
Mobile Database Systems

- **Embedded Mobile Databases**
  - Local lightweight database embedded into application
  - Example: SQLite

- **Mobile Client/Server Databases**
  - Lightweight database synchronized with database server
  - Examples: DB2 Everyplace, Oracle 10g lite, Tamino Mobile Suite etc.
SQLite
- Embedded SQL database engine
- Serverless, reads and writes directly to ordinary disk files
- Features
  - Support of local ACID transactions including nested transactions
  - Implements most of SQL92
  - Self-contained (no external dependencies)
  - Cross-platform support (db files can be moved)
  - Small code footprint: < 300KB fully configured or less than 180KB with optional features omitted.
Local data access

+ persistency for App data
+ simple programming API
+ runs on local device, embedded into application
+ No dependency on established, permanent network connection
+ Small footprint
+ Good performance

- DB in the scope of one application
  - no sharing of data between applications
  - no synchronization with backend
A mobile database system
- consists of a set of components distributed over mobile and stationary devices

Two architecture concepts
1. Extended Client/Server system
2. Middleware system with replication server
1. Extended Client/Server system

- Client runs lightweight replica containing a subset of the master DB
- Lightweight DB performs operations locally in disconnected mode (server emulation)
- Resynchronisation not permanently but
  - triggered by client or
  - periodically performed by system

Figure according to „Mobile Datenbanken und Informationssysteme“
Hagen Höpfner, Can Türker, Birgitta König-Ries
2. Middleware system with replication server

- **Intermediary middleware server for replication providing**
  - decoupling of Mobile DB and Master DB – multiple sources can be integrated at the replication server (transparent for client and server)
  - replication server can disburden data sources
  - access control at replication server

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

- **Architecture**: Middleware system with replication server

Figure according to „Mobile Datenbanken und Informationssysteme“ Hagen Höpfner, Can Türker, Birgitta König-Ries
Tools
- Mobile Device Administration Center (MDAC)
- XML Scripting tool

Allow definition of part of Master DB to be replicated
- at granularity of tables
- specification of horizontal or vertical filters to select particular rows or columns
- target table can be created out of data from multiple tables (join, merge)
- target tables can be read-only or read-write
- read-write restrictions
  - no aggregations for row/column selection
  - source table must not be created based on join or merge of other tables
  - just tables replicated (no integrity definition nor trigger)
Steps for synchronizing local changes with Master DB

1. Sync request from Mobile Client
2. Write updates to Admin Control DB
3. Input queue to Middleware
4. Copy updates, resolve conflicts to Mirror table
5. Capture changes to DB2 Log

Middleware:
- Staging table
- Change Data Tables
- Mirror DB

Master DB

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 DB updated with data from mirror DB
- User-based conflict resolution
  - based on program for log search
  - application-specific conflict resolution implementable
Disconnected Work with Mobile DBs

- Hoarding
  - Replication of data from Master DB
  - Explicitly defined by administrator using a tool
  - Prefetching of data according to defined scheme

- Emulation
  - App uses local DB to manipulate data
  - even if connection is established

- Reintegration
  - Synchronization of local DB with Master DB
  - Explicitly or periodically triggered
• NoSQL database solution with sync support

• **Couchbase Lite**
  - Embedded database with CRUD and query functionality
  - JSON and binary data representation supported
  - Queries based on N1QL (declarative query language that extends SQL for JSON)

• **Sync Gateway**
  - Sync between Couchbase Lite and Couchbase Server
  - Peer-to-Peer sync between Couchbase Lite instances
  - Runs on server or cloud

• **Data Access**
  - REST and Event-based APIs
  - JSON input and output

www.couchbase.com
Disconnected Operations - Principles

- introduction of proxy
- implementation of functionality for connections and disconnections (f.connected(), f.disconnected())
- transition function for proxy state between connection/disconnection

Dr. Thomas Springer

Application Development - 4. Disconnected Operations
Disconnected Operations – Generalization

- **Preparation phase**
  - ensure local availability of data and code
    - replication of data and caching
    - monitoring of user actions and prefetching
    - static or dynamic installation of emulation code on client

- **Disconnection phase**
  - local server emulation
    - changes of local data copies
    - recording of actions

- **Reintegration phase**
  - mediate local changes to server
    - conflict detection and resolution
Generalized Architecture

Application

read data

Caching
Prefetching
Write Log
Synchronization

Transfer control

write data

client-side Proxy

server-side Proxy

Server

Synchronization

Dr. Thomas Springer
Application Development - 4. Disconnected Operations 33
Weakly connected operations

- Limits of disconnected operations
  - limited task continuity due to lack of access to remote data
  - local changes not visible to remote clients
  - probability of conflicts raises with disconnection time
  - local resources required for caching, logging of operations and server emulation
  - bad performance and response times using low bandwidth links

- Weak connection support
  - mode of operation to hide network latencies
  - minimized disconnection times
  - lower resource requirements on mobile devices
  - increased consistency and task continuity
  - decreased probability of conflicts

- Low bandwidth links require explicit support
Support of weak connections

- Mechanism for adaptive and dynamic use of connection resources
  - load of new data and cache updates
  - reintegration of local changes

- Load of requested data has highest priority to support task continuity and short response time

- Cache updates and reintegration independent from response times
  - fast reintegration decreases size of local operation log and conflict probability
  - slower reintegration enables optimization of local operation log (e.g. eliminate operations),
    - reduced amount of data for network transfer
Partially Connected Operations

- Extension of Andrew File System (AFS)
- All file system writes are performed locally and logged
- Three prioritized classes of data transmissions for efficient network usage

1. Interactive communication
2. Read/Fetch data
3. Write data for reintegration

- Classes have different priorities
- One send queue for each class
- Specific send strategy to prioritize interactive communication and high availability of data in cache
Extension of disconnected operations in Coda

Implicitly 2 classes of data transmissions

1. Read operations for missed and outdated files with high priority
2. Trickle reintegration in background with low priority only if no read operations active

Calculation of threshold for max. file size for reading new files
- error message for larger files
Coda - Weakly Connected Operations

- resolving strong distinction between connected and disconnected
- if small bandwidth network connection available
  - operation in disconnected mode for writing
  - but performance of reintegration functionality for cache updates

Dr. Thomas Springer
Application Development - 4. Disconnected Operations
Ongoing background reintegration with minimally impact on foreground activity

- balance between
  - effectiveness of log optimization and
  - conflict probability
- minimal delay time (adjusted according current connectivity)
  - replay operation remains defined time in modify log (aging window) (10 min. based on simulations)
Idea of progressive enhancement
- Start with an assumption of base-level capabilities
- Than take advantage of advanced capabilities when they become available

Offline-first
- Assumption for base-level: no connection
- Progressively enhance when connection becomes available
- Shift in mindset: lack of connection is not an error
- Build your app to work without a connection
Social Fitness App – Offline Challenge

- Retrieving timeline entries

- Check availability of connection first
- Cache retrieved timelines when connection is available

Dr. Thomas Springer

Application Development - 4. Disconnected Operations
Social Fitness App – Offline Challenge

- Uploading local workout data

- Store workout data locally in persistent storage
- Sync with server when connection becomes available

Cycling

June 26, 2016

27.3 km: 26:39

„Erste Etappe war eine einzige Quälerei, beim nächsten Mal geht’s bestimmt besser“
Conflicting updates for team activity

Use weakly connected mechanisms to balance connectivity requirements and in-time data updates
Goal: Transparency of connection loss for programmer and user

Combination of Mechanisms
- Caching/Prefetching
- Emulation of server on client-device
- Delayed Write-back
- Balanced network usage

Disconnected/weakly connected operations is a blueprint for application specific solutions

Adopt Offline-first principle
References

- “Mobile Datenbanksysteme” Bela Mutschler, Günther Specht.
- „Mobile Datenbanken und Informationssysteme“ Hagen Höpfner, Can Türker, Birgitta König-Ries
- “UltraLite Database Management and Reference Sybase iAnywhere”