MOBILE MIDDLEWARE

Kennedy Kamwendo
Stein Kristiansen

INF 5040
2008
MOBILE MIDDLEWARE

Outline
- Introduction
- CARISMA
- MADAM
- Comparison
MOBILE APPLICATIONS - INTRODUCTION

- Types and use of mobile devices increasing
  - More applications being developed for Mobile devices

- Mobile devices have limited resources
  - Energy, memory, cpu, b/w, etc
  - Devices operate in changing environments
  - Need for mobile applications to adapt to changing environment
  - Context changes impact on mobile application design
**Mobile Middleware**

- Middleware is a general term for any programming that serves to mediate between two separate and often already existing programs.

- Ease the design and development of context aware mobile applications by:
  - Abstracting the low level device details
  - Taking appropriate measurements
  - Collaborating with participating devices
  - Offer services to the Application level
CARISMA

- Context Aware Reflective Middleware System for Mobile Applications
  - Licia Capra, Wolfgang Emmerich, Cecilia Mascolo - Oct 2003

- CARISMA - project carried out at University College London.

- Middleware model that exploits reflection to enable context-aware interactions between mobile applications.

- Context - everything that can influence the behavior of an application
CARISMA

- Not fully transparent middleware system
- Employs reflection to alter middleware behavior
- Change of middleware behaviour dynamically altered by application
- Reflection introduces conflicts
THE REFLECTIVE MODEL

- Mobile devices change operating context rapidly
- Context changes are monitored by middleware
- Context configurations determine the policies applied in a given service for a particular context
- Reflective model allows applications to dynamically change middleware behaviour

![Diagram showing the reflective model with two subdiagrams: (a) showing absorption and reflection, and (b) showing reflective API and application profile connections to middleware.](image)
Profiles

- The behaviour of middleware with respect to an application
- Set of associations between services, policies and context configurations
- Profiles passed down to middleware from application
- One policy per time for each service.
- Applications use Reflective API to dynamically change their profiles in tune with changing environment
CONFLICTS

- Reflection introduces conflicts – when more than one policy is valid

- Two types of Conflicts:
  - Intrapofile conflict
    - Local to middleware instance i.e. on same device
  - Interprofile conflict
    - Distributed among various middleware instances i.e. different devices
EXAMPLE APPLICATION

- Conference Application
- Reminder of the next Talk
  - Service:
    - talkReminder
  - Policies Context:
    - soundAlert  Outdoor
    - vibraAlert  Attending another talk
    - silentAlert Using the PDA

```
talkReminder
soundAlert
  location = outdoor
vibraAlert
  location = conferenceroom
silentAlert
UserFocus = on
```
Example Application

- Conference Application
- Message Exchange
  - Service:
    - MessagingService
  - Policies
    - charMsg
    - plainMsg
    - compressedMsg
    - encryptedMsg
  - Contexts
    - Battery
    - bandwidth
EXAMPLE APPLICATION CONT.

- Distributed application
- If Alice’s battery < 40% and Claire’s bandwidth > 50%
  - No conflict.
- If Alice’s battery > 40% and Claire’s bandwidth > 50%
  - Interprofile conflict!

% Alice
messagingService
plainMsg
battery < 40%
compressedMsg
battery > 40%

% Claire
messagingService
plainMsg
bandwidth > 50%
compressedMsg
bandwidth < 50%

% Bob
messagingService
plainMsg
RESOLVING CONFLICTS

- Requirements on the solution
  - Dynamic
    - Static conflict resolution not realistic
  - Simple
    - Should not consume too many resources
  - Customizable
    - Applications must be involved in the resolution
CONFLICT RESOLUTION

- Auction Protocol used
- **ALL** participating applications must bid on **ALL** conflicting policies
- Utility function is used to determine bid amount
- Middleware computes solution set
  - Set of common preferred policies
- If set is empty no solution is possible
- If only one element in set, then no conflict exists
- If more than one element, then middleware must compute solution
CONFLICT RESOLUTION CONT.

- Bets for policies have values
- Bets values depend on how important a policy is to an application – utility function
- Policy with highest sum of bids wins

- All auctions are isolated
- Applications pay for winning policy

Kenedy Kamwendo & Stein Kristiansen
MESSAGING EXAMPLE

charMsg: {(battery, 4), (bandwidth, 10),
          (availability, 10)}
plainMsg: {(battery, 3), (bandwidth, 6),
           (availability, 7)}
compressedMsg: {(battery, 5), (bandwidth, 4),
                 (availability, 5)}
encryptedMsg: {(battery, 6), (bandwidth, 7),
               (availability, 4), (privacy, 10)}

% peer 1    % peer 2    % peer 3
messegingService
charMsg
  bandwidth > 70%
plainMsg
  bandwidth < 70%
compressedMsg
  bandwidth < 35%
encryptedMsg
  battery > 50%

% peer 2
messegingService
plainMsg
  battery < 50%
compressedMsg
  bandwidth < 40%

% peer 3
messegingService
plainMsg
  compressedMsg
  bandwidth < 40%
)}

P1 = {plainMsg; compressedMsg; encryptedMsg}
P2 = {fplainMsg; compressedMsg}
P3 = {plainMsg; compressedMsg; encryptedMsg}
LMAX = 10  WMAX = 10  RQMAX = 5

% peer 1  battery  4
bandwidth 3
availability 10

% peer 2  battery  7
bandwidth 9

% peer 3  privacy 10
battery  10

\[ b_{pi} = \frac{L_{\text{MAX}} - l_R^* w + l_B^* w}{L_{\text{MAX}} R^* L_{\text{MAX}} B^* RQ_{\text{MAX}} R} \]

\[ b_{p1} = \frac{0 - 3^* 4 + (0 - 6)^* 3 + 7^* 10}{10^* 10^* 5} \]

\[ b_{p1} = 0.22 \]

<table>
<thead>
<tr>
<th>peer</th>
<th>plainMsg</th>
<th>compressedMsg</th>
</tr>
</thead>
<tbody>
<tr>
<td>peer_1</td>
<td>0.22</td>
<td>+</td>
</tr>
<tr>
<td>peer_2</td>
<td>0.17</td>
<td>+</td>
</tr>
<tr>
<td>peer_3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.39</td>
<td>0.354</td>
</tr>
</tbody>
</table>

\[ q_1 = q_1 - \frac{0.22 - 0.17}{1} - \frac{0.17 - 0}{2} - \frac{0}{3} \]
IMPACT OF REFLECTION
CONTEXT AWARENESS

The graph illustrates the relationship between the number of policies and the time (ms) required for processing. The lines represent different scenarios:

- **1 context, 1 resource**
- **3 contexts, 5 resources**
- **5 contexts, 10 resources**

As the number of policies increases, the time required also increases. The graph shows a clear trend of increasing time with an increase in the number of policies.
IMPACT OF UTILITY FUNCTION PARAMETERS

The diagram illustrates the impact of utility function parameters on response times for different numbers of conflicting policies. The x-axis represents the utility function parameters, while the y-axis shows the response time in milliseconds (ms). The graph compares three scenarios: 2 conflicting policies, 5 conflicting policies, and 10 conflicting policies. The data shows variations in response times as the parameters change.
CONFLICT RESOLUTION MECHANISM

![Graph showing the relationship between the number of policies and the number of conflicts. The graph compares the number of conflicts with and without conflicts.](image)
DISTRIBUTED CONFLICTS

![Graph showing the relationship between the number of devices and milliseconds (ms.) with different policies and contexts.

- Solid line with diamonds: 3 policies, 2 conflicts, 1 context, 1 resource
- Dashed line with squares: 5 policies, 3 conflicts, 3 contexts, 5 resources

X-axis: N of Devices
Y-axis: ms.
MADAM

“A Comprehensive Solution for Application-level Adaption”

K. Geihs et al., 2008
OVERVIEW

- Model-driven adaptation of component-based applications
- Development methodology, tools and middleware
- Separates application and context monitoring/adaptation concerns
  - Relieve applications of adaptation concerns
  - Context relations and adaptation effect conflicts resolved by middleware
  - Middleware can evolve independently of applications
- Utilizes utility functions
  - To resolve conflicts in goals and adaptation effects
- Supports distributed adaptation
OVERVIEW

mobile user

context

user needs

battery

computing resources

network QoS

noise

position

light

monitors

describes dependency

adapts

used to derive application variant

architecture model

adaptation middleware

adaptable application

preferred quality

provided quality

Kenedy Kamwendo & Stein Kristiansen
COMPONENT FRAMEWORK

- Component frameworks describe the composition of *component types*
- Component types represent *variation points* into which *components (implementations)* can be inserted (described by *plans*)
  - A specific composition of components is called an *application variant*
- Components can be atomic or composed
- Components have properties
  - Services and required resources
  - Given by property predictors
  - Attached to *ports*
Middleware Functionalities

- Context monitor
  - Separate context middleware allows for new elements + new forms of reasoning

- Make decisions for adaption
  - Many mappings between context and adaptation
  - Utility function $f(\text{predicted properties, current context})$ finds variant with highest utility

- Reconfigure applications
  - Application adapted by re-composition
  - Focus on compositional adaption, but also supports parameterization
MADAM ARCHITECTURE
**MADAM ARCHITECTURE**

- Keeps state of resources
- Accessed by **Context Manager** and **Adaptation Manager**

---

**Diagram**:

- Resource Manager
  - Local Resource Management
  - Global Resource Management
  - Context Listener
  - Device Reconfiguration
  - Application Reconfiguration
  - Plan Listener
  - Component Management
  - Binding Management
  - Core
  - Instance Management

**Nodes**:

- Context Manager
- Adaptation Manager
- Configurator
MADAM ARCHITECTURE

- Provides platform-independent interface
- Instantiate/initialize/remove/connect/disconnect component instances
- Collaborates with Adaptation Manager to launch and manage applications
**CONTEXT MANAGEMENT**

- Context model: a simple and extensible model of application context
  - Allows users to specify arbitrary types of context entities and values
  - Categories of context elements: user, computing and environment
  - *ContextEntities* form a tree, where leaves are values

- Context manager
  - Hub among *sensors, reasoners* and *consumers*
  - Supports dynamic plug-in of sensors & reasoners (specializations of sensors)
DISTRIBUTED CONTEXT
ADAPTATION MANAGER

- Adapts the device, and set of running applications to best fit the current context
- Two sub-components: AdaptionCoordinator and BuilderandPlanner
- BuilderandPlanner provides configuration templates describing alternative application variants
- AdaptionCoordinator:
  - Evaluates configuration templates, and computes utility only for those for which we have enough resources
  - Selects ‘best’ variants, and choosing the one with highest utility (according to utility-function)
  - Configurator (possibly remote) is requested to instantiate the corresponding application configuration
CONFIGURATION MANAGEMENT

- Orchestrates changes between sets of application variants
- Realized by Configurator
  - Receives configuration plans from AM
  - Adds/Removes/(Dis)Connects instances
  - Configures parameters
  - Change device settings
  - Compares current conf. with new to find minimal sequence of steps
  - Preserve/transfer component state
APPLICATION LIFECYCLE

- **Launch:**
  - Adaptation manager recursively retrieves plans + registers for context changes
  - *Configurator* instantiates and configures device

- **Run-Time:**
  - Adaptation manager triggered by context manager
  - May replace variant, and invoke configurator

- **Shutdown:**
  - Adaptation manager performs cleanup
  - Finds the best configuration for remaining applications
DISTRIBUTION

- Master-Slave
- Master: Typically the one with core application components
  - Makes adaption decisions
  - Considers distributed context
  - Potentially delegates configuration operations to slave nodes
- Centralized = scalability problems
  - ...but mobile devices run few applications, hence has few connections to other devices
Prototype

- Implemented in Java
  - Supports cross-platform portability
  - Can be run on devices supporting the J2ME CDC Personal Profile specification
  - Or devices supporting the Sun JRE J2SE 1.3.1 specification
- UDP/IP for distributed context management
- Adaptation and resource managers utilize Java RMI
PILOT APPLICATIONS

Pilot 1: Applications for service technicians (by Condat AG for Daimler Chrysler AG and Deutsche Bahn AG)
- PDA for inspection, measurements and maintenance of technical equipment
- Ex.: Adapt to communication link

Pilot 2: SatMotion (by Integrasys SA)
- PDA controls measurement equipment and server to find optimal alignment of satellite antenna
- PDA receives signals from server, visualizes and evaluates, then user replies with optimal alignment
- Ex.: Adaptation of operation mode - *two-way, one-way or local*, based on bandwidth
## Performance

### Table I. Devices used for the measurements.

<table>
<thead>
<tr>
<th>Device</th>
<th>Processor</th>
<th>Memory</th>
<th>OS</th>
<th>Java VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop PC</td>
<td>Pentium 4 2.4 GHz</td>
<td>1 GB</td>
<td>MS Windows XP SP2</td>
<td>JDK 1.3.1.15</td>
</tr>
<tr>
<td>PDA HP iPAQ 6345</td>
<td>Texas instruments 68 MHz</td>
<td>64 MB</td>
<td>MS Windows Mobile 2003</td>
<td>CrèMe 4.00</td>
</tr>
</tbody>
</table>

### Table II. Selected measurement results.

<table>
<thead>
<tr>
<th>Metric</th>
<th>PDA</th>
<th>Desktop PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory (base)</td>
<td>6–8 MB</td>
<td>6–8 MB</td>
</tr>
<tr>
<td>CPU (continuous)</td>
<td>13%</td>
<td>7%</td>
</tr>
<tr>
<td>Context latency (local)</td>
<td>190 ms</td>
<td>23 ms</td>
</tr>
<tr>
<td>Context latency (remote)</td>
<td>287 ms</td>
<td>70 ms</td>
</tr>
<tr>
<td>Planning (skeleton test application, 100 variants)</td>
<td>787 ms</td>
<td>87 ms</td>
</tr>
<tr>
<td>Planning (skeleton test application, 1000 variants)</td>
<td>1125 ms</td>
<td>112 ms</td>
</tr>
<tr>
<td>Planning (pilot 1, 12 variants)</td>
<td>742 ms</td>
<td>87 ms</td>
</tr>
<tr>
<td>Planning (pilot 2, 180 variants)</td>
<td>1300 ms</td>
<td>430 ms</td>
</tr>
<tr>
<td>Reconfiguration (tiny test application)</td>
<td>715 ms</td>
<td>94 ms</td>
</tr>
<tr>
<td>Reconfiguration (pilot 2)</td>
<td>4400 ms</td>
<td>1468 ms</td>
</tr>
</tbody>
</table>
CARISMA vs. MADAM

- Both support adaptation according to context
- Both utilize customizable utility functions

<table>
<thead>
<tr>
<th>Difference</th>
<th>CARISMA</th>
<th>MADAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>Middleware</td>
<td>Development methodology, tools, middleware</td>
</tr>
<tr>
<td>Adaptation approach</td>
<td>Policies</td>
<td>Variants</td>
</tr>
<tr>
<td>Adaptation extent</td>
<td>Fixed number of policies</td>
<td>Open-ended range of alternative adaptations</td>
</tr>
<tr>
<td>Distribution paradigm</td>
<td>Peer-to-peer</td>
<td>Master-Slave</td>
</tr>
<tr>
<td>Distributed elements</td>
<td>Policies and decision making</td>
<td>Application and Context</td>
</tr>
</tbody>
</table>
Thank you!

Any questions?