Client-Side Context Storage

Stephanie Riche
HP Labs Grenoble
HP Labs Grenoble

• Created in April 1999
• Collaboration with European labs & universities
• Research on local environment of appliances with personalization
Overview

Goal:  

*Enabling personalization of the digital experience*

Requirements:

1. *Get knowledge*: Rich user context information  
   User preferences & habits, user location, time, activity, environment, history of interaction with services…

2. *Store knowledge*: Protection of user Privacy  
   Access context information from anywhere, any user device in a trusted way

3. *Use knowledge*: Personalize the interaction  
   User interfaces, web services, smart spaces…
Existing personalization & context-aware architecture

Drawbacks:

- Low privacy protection
- Inconsistency of data

=> Frustration of the user.
User-centered architecture

Advantages:

• Privacy protection.
• Context data consistency
• Always available context

=> Unified & enhanced customer experience.
User-Centered Architecture

The context storage system on user devices can be used by remote context systems as:

- A “cache”
- An additional source of context information
Context Storage System Requirements

- Simple memory model, no “hot sync” effect:
  - Minimize user involvement to reconcile replicas
  - Awareness about information consistency: **dependable system**
    => limit system divergence
- Adapt to device characteristics
  => Selective replication, asynchronous communication
- Personal storage
  - Privacy enforcement: **privacy awareness & control**
    => Control & log of accesses
  - Personal usage
    => Adapt to user behavior
Prototype design

- Single master optimistic protocol
- Migration of master replicas along the user
- Pro-active replication based on observed access patterns
Demos

Visibility of all bookmarks stored in the profile store

Presentation of bookmarks relevant in a given context
### Context Storage System versus Distributed file systems

<table>
<thead>
<tr>
<th></th>
<th>Context Storage System</th>
<th>Distributed file systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access patterns</strong></td>
<td>Conflicts are frequent &amp; user involvement should be minimized</td>
<td>Conflict are rare &amp; user involvement is ok</td>
</tr>
<tr>
<td><strong>Number of users</strong></td>
<td>1</td>
<td>Usually hundreds</td>
</tr>
<tr>
<td><strong>Data requirements</strong></td>
<td>Usually unknown by the user but could be pro-actively fetched.</td>
<td>Usually unknown by the user but could be pro-actively fetched</td>
</tr>
<tr>
<td><strong>Data Types</strong></td>
<td>Heterogeneous, privacy requirements at a fine grain level</td>
<td>Files</td>
</tr>
</tbody>
</table>
Tests & experimentations

- **Goals**: Rapid experimentation of emerging concepts. Measure metrics to compare several approaches.

- **Experimentation framework**
  Framework allowing rapid experimentation on a distributed system. Address the issue of rapid deployment and execution of a distributed scenario.

- On each machine resides an agent host in charge of receiving proximity agents.

- A proximity agent can be deployed on a remote machine, can migrate with state information upon machines. Proximity agents can be remotely accessed for information collection or configuration purpose.
An aside - Client-side Federation

- A local profile system on a client device can be used as a “cache” for remote profile systems.
- Synchronization when device is connected to internet.
- Intercept calls to remote profile systems, and redirect locally.
- Remote services may call client device profile or remote one.
- Local device may have local only data as well as cached data from remote systems.