A Unified Service Mobility Model

open source project codename *chameleon*

by Vincent Verdot.

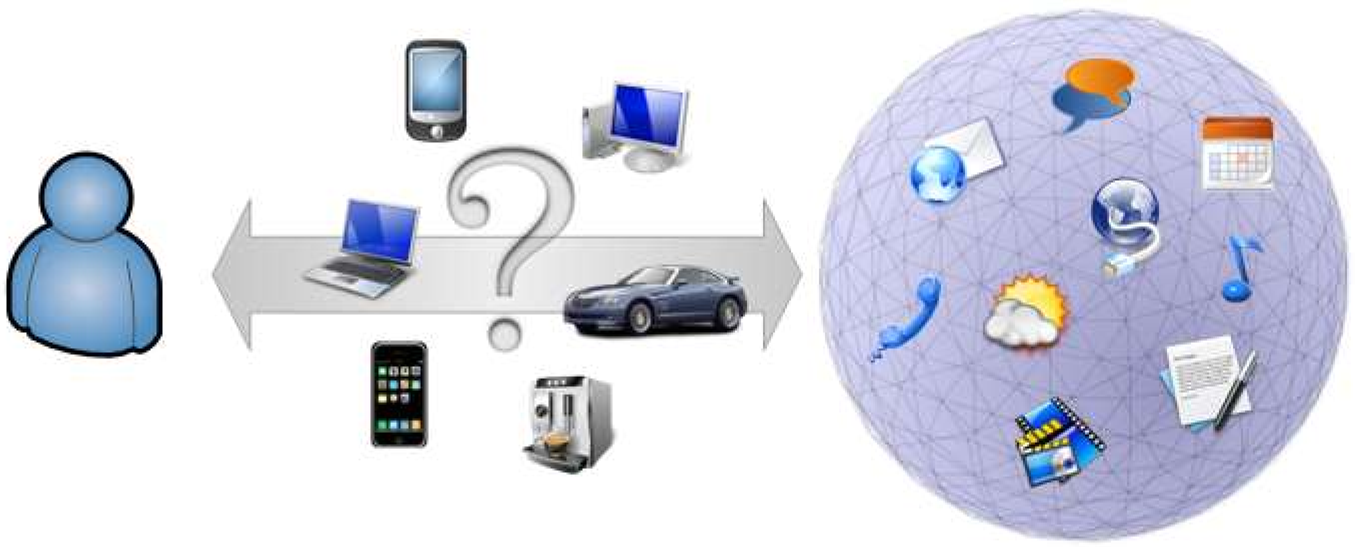
The Challenge

- Providing a **Unified Service Mobility Model** (US2M).
  - *Objective 1*: Identify the technical issues.
  - *Objective 2*: Define the concepts.
  - *Objective 3*: Design an architecture.
  - *Objective 4*: Propose an implementation

- "Unified": the model must irrespectively handle:
  - any type of service,
    - communication, text-edition, video-streaming, web browsing…
  - on any device.
    - regardless of capabilities, operating system, applications…

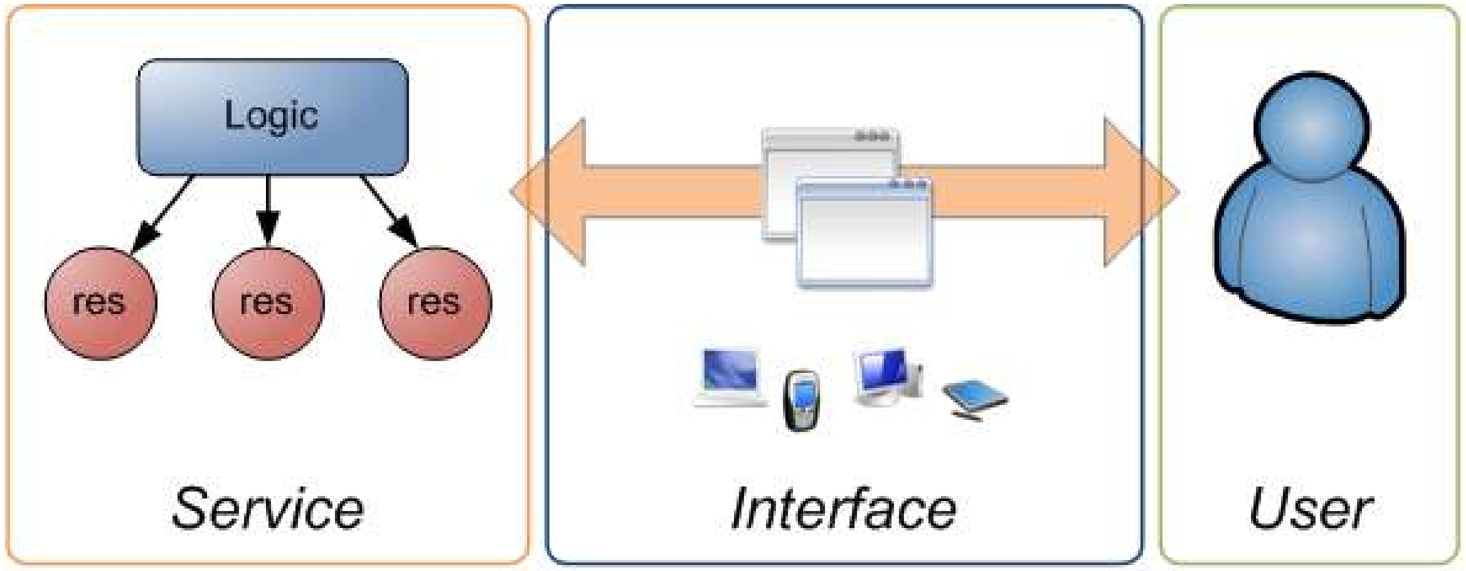
The Concept

- A user interacts with his Personal Service Environment
  - Each device is a potential interface to his services.
  - Services are no more tied to their host.



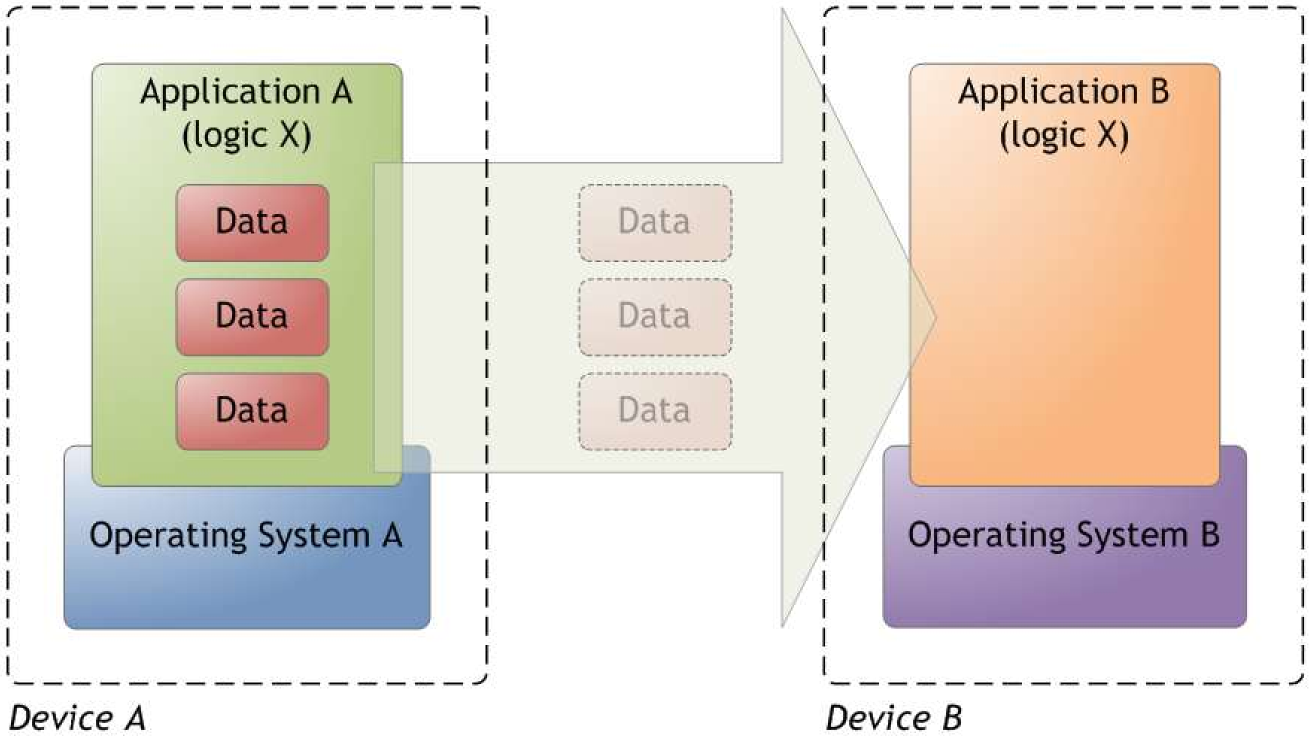
The Idea

- Service = logic + resources
  - *A service is an algorithm (the logic) that processes a set of resources (the context) via an application, and which delivers a functionality of its own to at least one user.*

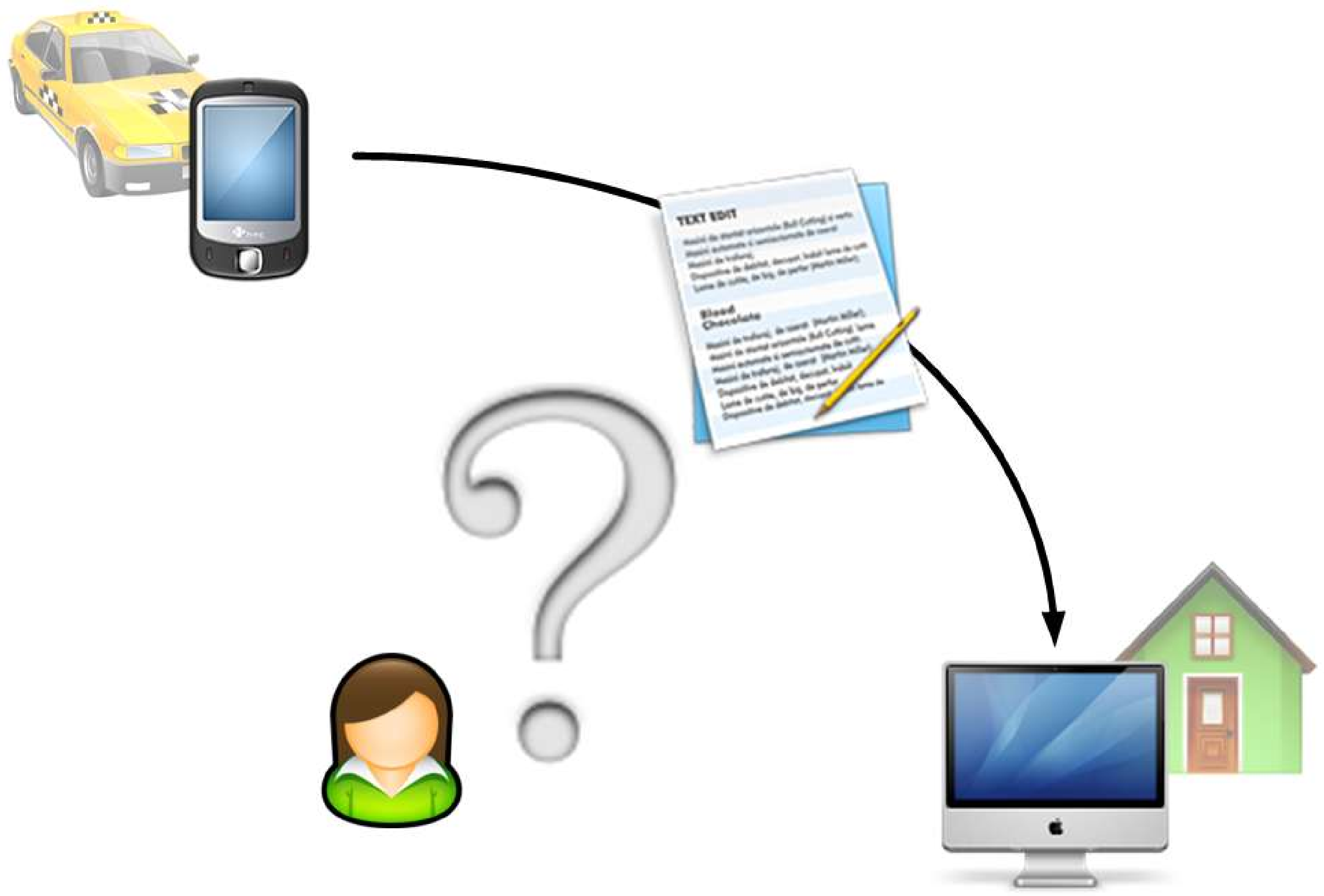


The Idea

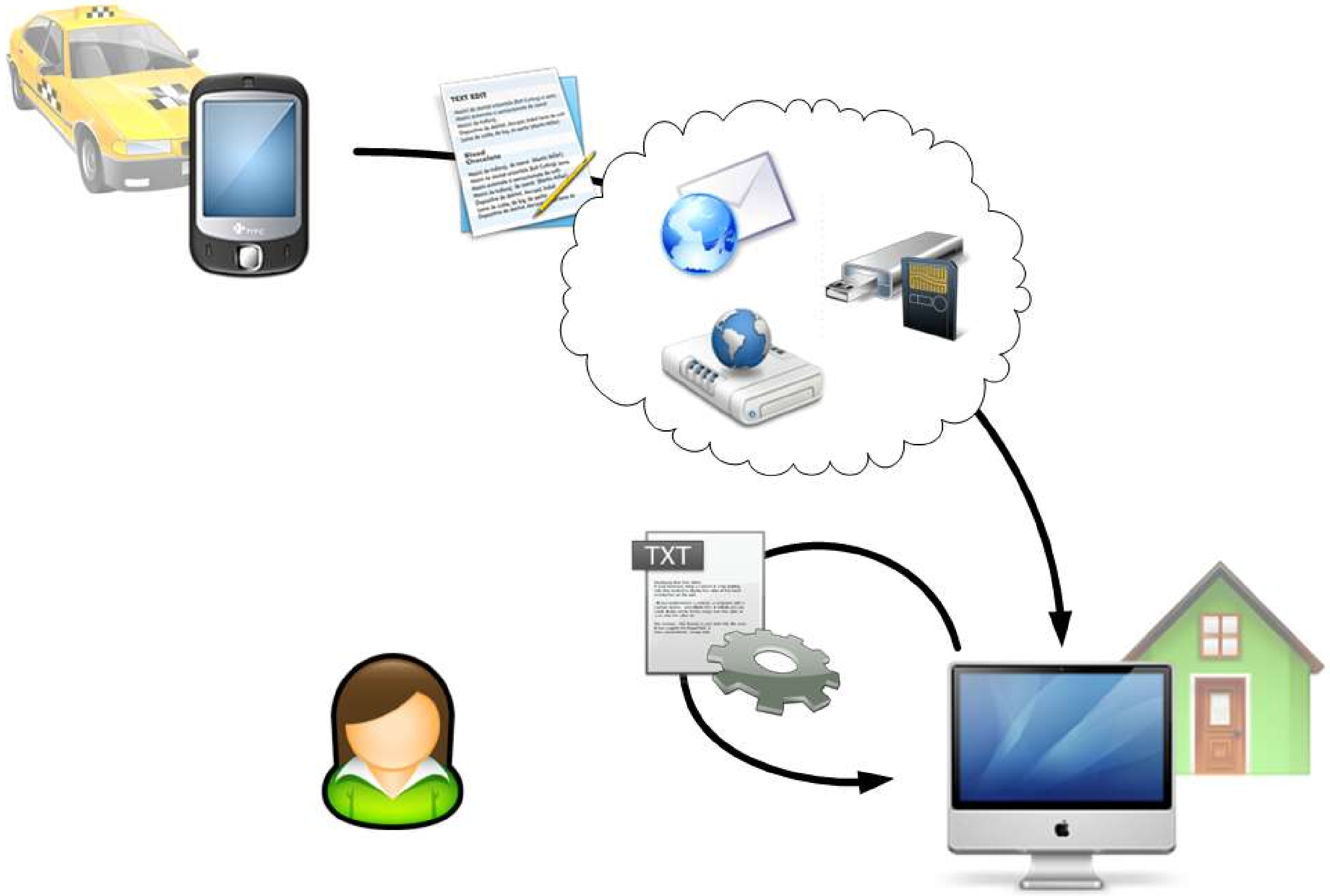
- The mobility of services consists in transfering these resources an optimized way.



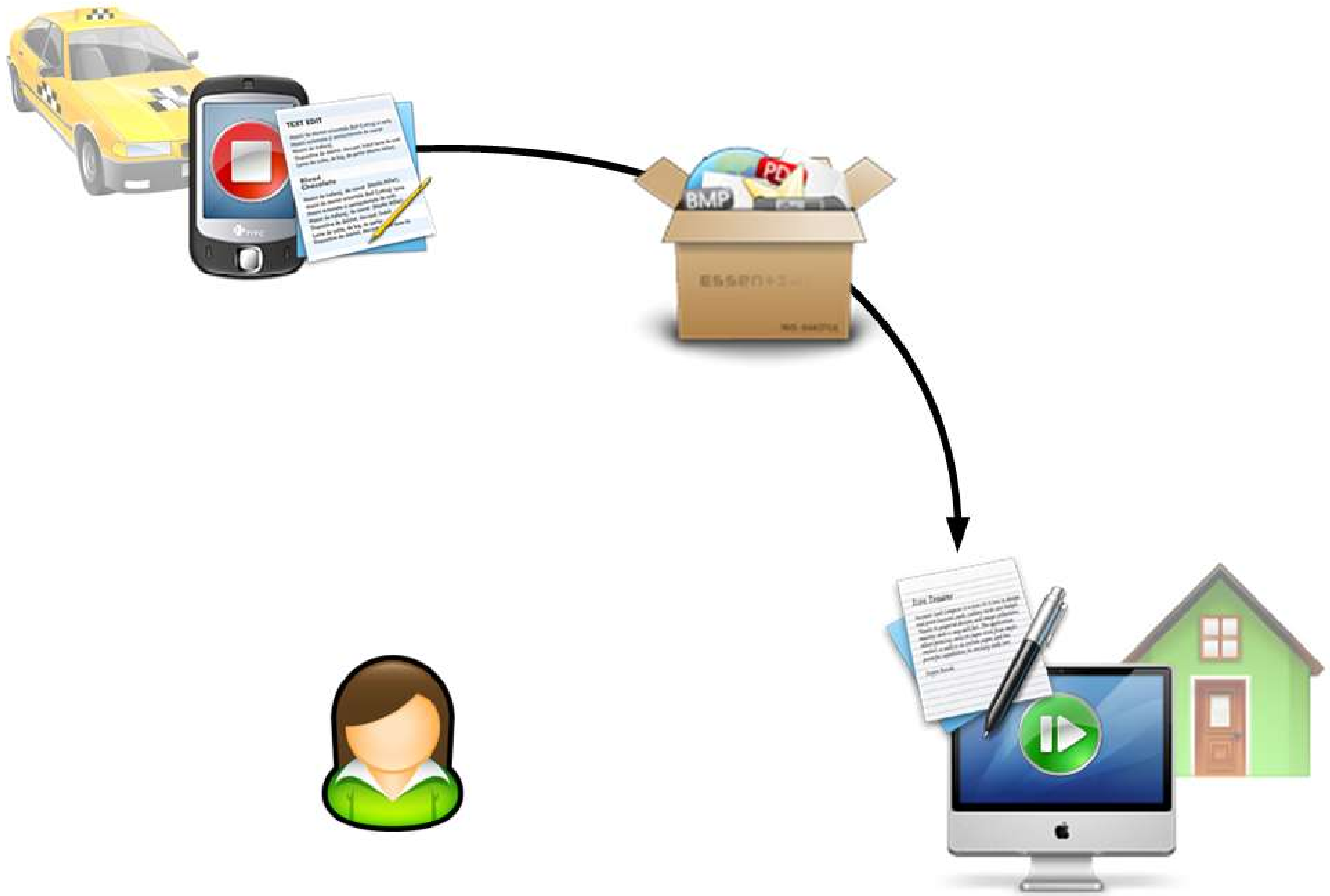
The Use Case



The Use Case: former method



The Use Case: new method



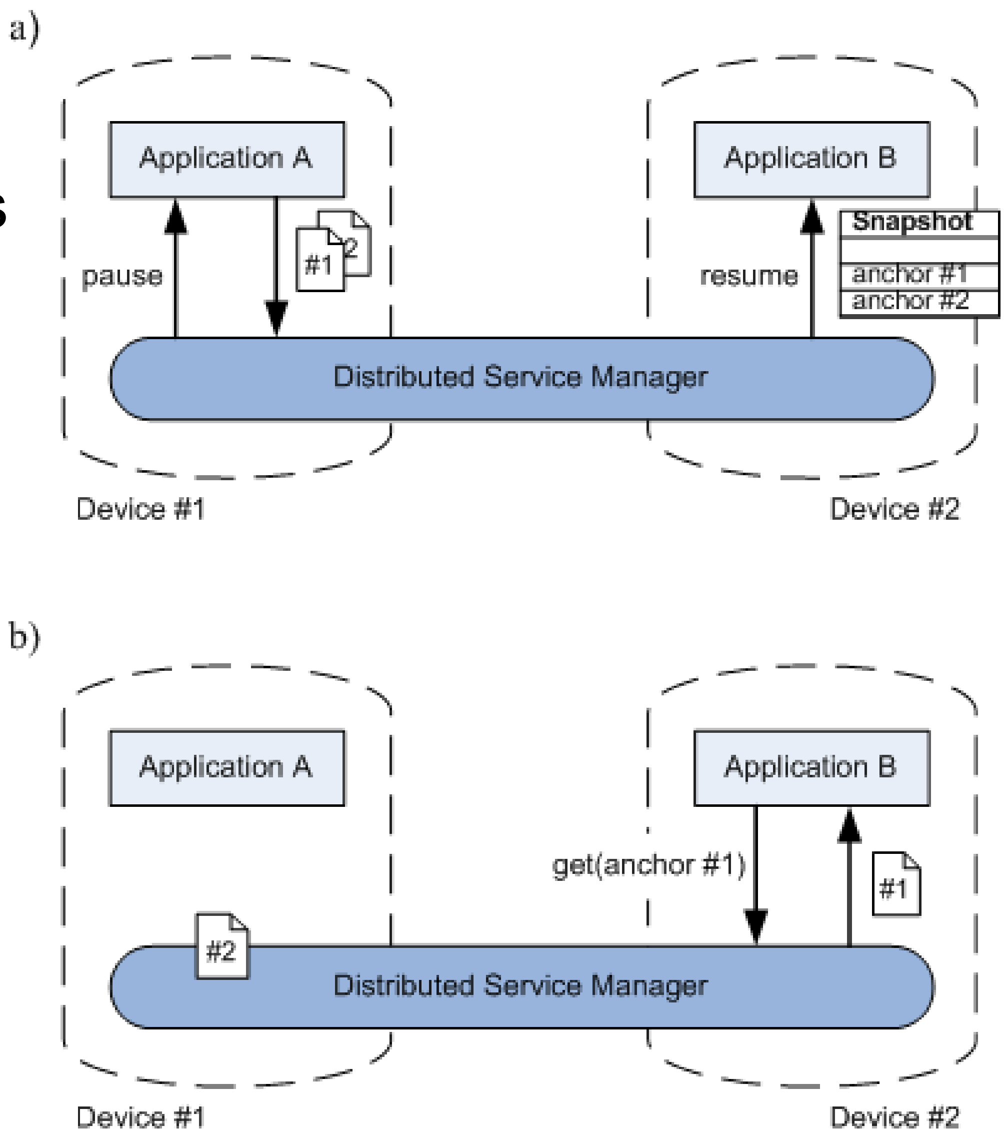
The System

- A distributed Service Manager (dSM), installed on every user's devices assures:
  - the control of local applications,
  - the management of user's PSE,
  - the exchange of service advertisements,
  - the transfer of resources.

- Service adapation is assured by the application itself.
  - Pause function
    - The application delivers a stable version of its context.
  - Resume function
    - The application starts from the provided stable context.

The System

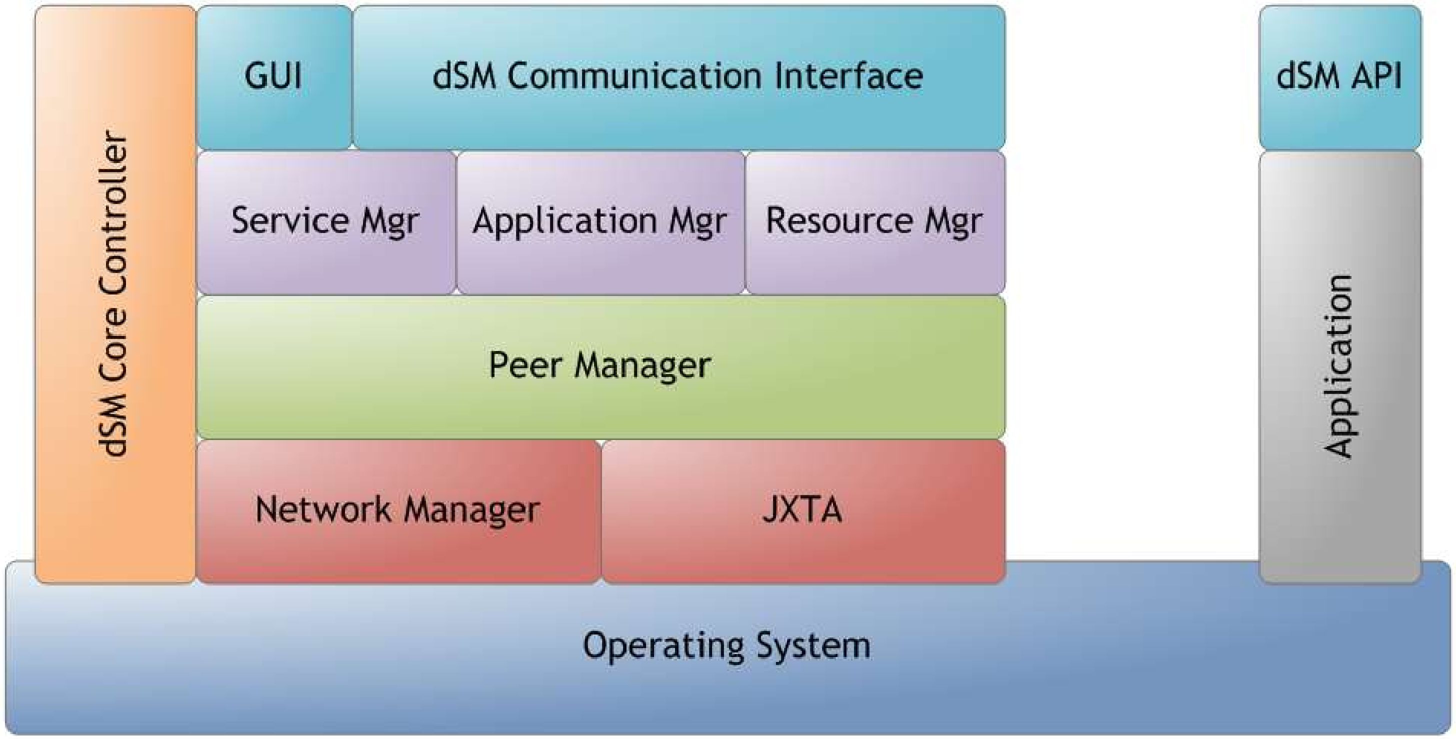
- An anchor-based mechanism guarantees an optimized resource transfer.
  - Resources are actually transferred on application request.
  - A lightweight snapshot describing the context resources is provided.
  - Optional or unsupported resources are discarded.



The Architecture

- The core mechanism, (preferably integrated to the host's OS) provides the following features:
  - application control,
  - service registration and advertisement,
  - PSE management (device discovery, authentication, ..),
  - GUI events.
- The dSM API, integrated to compliant applications, provides a high-level library of dSM functions
  - management of applications, services, resources…

The Architecture



The Prototype

- Open-Source code: Java
- P2P layer: JXTA
  - Poor performances (compatible J2ME)
- Proprietary XML messages for signaling
- Eclipse Rich Client Platform GUI
  - Offer a convenient application framework
  - dSM is independent from any GUI, it exposes events
- Code details
  - dSM .jar file 100KB, dSM API less than 40KB
- Under L-GPL v3 License

The Experimentations

- Text-Edition Service transfer (limited)
  - Same OS (Windows XP)
  - Same Application (Java Notepad)
  - Same type of device (PC)
  - Few resources available
    - Text body, cursor position and possibly typing history
  - Measures and simulation of other types of service
- Usability survey
  - How users currently transfer their services?
  - Identify and characterize the methods
  - Compare ease of use and intuitiveness of solutions

The Contributions

- Call for contribution in open-source community
  - Great project, so much manpower needed
  - High potential, innovative but too much for a single man
- Many tasks
  - System optimization
    - P2P overlay, Network Interfaces, Global code optimization,…
  - Graphical User Interface
    - Something more “sexy”
  - Portability to mobile architectures
  - Making more services “dSM-compliant”
- How to contribute
  - Now in launchpad as project **dSM**, codename chameleon.

Now Let's Get to Work!

Tank you.