Resilient Partitioning of Pervasive Computing Services

Engineer Bainomugisha

Promotor:
Prof. Dr. Wolfgang De Meuter

Advisors:
Jorge Vallejos
Elisa Gonzalez Boix

Pervasive Computing Environment

Computational power is available everywhere (Weiser, 1993)
Partitioning of Pervasive Computing Services

**Scenario:** Hotel’s Ambient Services

- **Hotel room**
  - Laptop
  - Hi-Fi system
  - Guest
  - Cell phone
  - Ambient music player
    - Controller
    - Music library
    - Audio
Partitioning of Pervasive Computing Services

An application can be decomposed to run on multiple devices

Ambient music player

- Laptop
- Music library
- Cell phone
- Controller

Hi-Fi system

- Audio

Hotel room

- Runtime Application Partitioning
- User Controlled Application Partitioning
Partitioning of Pervasive Computing Services

Applications (whole or part) are not constrained to run on one device

- Laptop
  - Ambient music player
    - music library
- Cell phone
  - Controller
  - Audio
- Hi-Fi system
  - Hotel room

Retractable Application Partitioning
Partitioning of Pervasive Computing Services

Hotel room

Laptop
Hi-Fi system

Cell phone

Ambient music player
controller music library audio

Resilient to Network Failures
Resilient Service Partitioning

- Runtime Application Partitioning
- User Controlled Application Partitioning
- Retractable Application Partitioning
- Resilient to Network Failures
### Survey of Related Work for Service Partitioning

<table>
<thead>
<tr>
<th></th>
<th>Runtime Partitioning</th>
<th>User Controlled</th>
<th>Retractable</th>
<th>Resilient to Network Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object-oriented partitioning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J-Orchestra</td>
<td>✗ (compilation)</td>
<td>✗ (programmer)</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Addistant</td>
<td>✗ (compilation)</td>
<td>✗ (programmer)</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>JavaParty</td>
<td>✓ (load-balancing)</td>
<td>✗ (runtime system)</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Doorastha</td>
<td>✓ (runtime system)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Component-oriented and agent-oriented partitioning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coign</td>
<td>✗ (compilation)</td>
<td>✗ (algorithm)</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>AdJava</td>
<td>✓ (load-balancing)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hydra</td>
<td>✓ (load-balancing)</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
</tr>
</tbody>
</table>
Resilient Actor Model

A resilient actor:

- is a **modular** – encloses application functionality
- is **active** – has its own thread of execution
- defines **elastic bindings** to other resilient actors
Resilient Actor Model

A resilient actor:

- is a **modular** – encloses application functionality
- is **active** – has its own thread of execution
- defines **elastic bindings** to other resilient actors

Two partitioning operations:

- **Stretch**: moves the resilient actor from one device to another
- **Retract**: moves the stretched actor back to original device
Resilient Actor Model

A resilient actor:

- is a **modular** – encloses application functionality
- is **active** – has its own thread of execution
- defines **elastic bindings** to other resilient actors

Two partitioning operations:

- **Stretch**: moves the resilient actor from one device to another
- **Retract**: moves the stretched actor back to original device

Resilient Strategies:

- that specify different behaviors of the partitioning operations
Service Partitioning

Cell phone

Ambient music player

Objects

Local references
Service Partitioning

Cell phone

ambient music player

controller

music library

audio

Keyboard

Local resources

Speakers
Service Partitioning

Cell phone

ambient music player

controller

music library

audio

Keyboard

Speakers

Hi-Fi system

Speakers

Stretch
Manual Retraction

Cell phone

- ambient music player
- controller
- music library
- Keyboard
- Speakers

Hi-Fi system

- audio
- Speakers
Manual Retraction

Cell phone

- ambient music player
- controller
- music library
- audio
- Keyboard

Hi-Fi system

- audio
- Speakers
- Retract
Manual Retraction

Cell phone

ambient music player

controller

music library

audio

Keyboard

Speakers

Hi-Fi system

Speakers
Automatic Retraction

Cell phone

- ambient music player
- controller
- music library
- audio
- Keyboard

Hi-Fi system

- audio
- Speakers
- Retract
Automatic Retraction

Cell phone

ambient music player

controller

music library

audio

Keyboard

Speakers

Hi-Fi system

Speakers
Resilient Strategies

Stretch and Retract operations can have different implementations.
Resilient Strategies

**Propagation**: Partitioning operations proceed through elastic bindings.

[Diagram showing the relationship between Cell phone, Hi-Fi system, and strategies like Rebind, Copy, Move, Standstill, Controller, Music Library, Audio, Speakers, Keyboard, etc.]
Stretch Operation with Move Strategy

Cell phone

- ambient music player
- controller
- music library
- keyboard

Hi-Fi system

- audio
- speakers

[Move Strategy]
Retract Operation with Move Strategy

Cell phone

ambient music player

controller

music library

Copy of audio

[Move Strategy]

Retract

Hi-Fi system

Copy of audio

audio

Speakers
After Retract Operation with Move Strategy

Cell phone
- ambient music player
- [Move Strategy]
- controller
- music library
- audio
- Keyboard
- Speakers

Hi-Fi system
- Speakers
Stretch Operation with Copy Strategy

Cell phone

Laptop

ambient music player

[Copy Strategy]

music library

controller

Keyboard

audio

Speakers

Copy of music library
Retract Operation with Copy Strategy

Cell phone

- ambient music player
- [Copy Strategy]
- [Move Strategy]
- music library
- Keyboard
- controller

Laptop

- Copy of music library
- [Move Strategy]
- Speakers
- audio

Retract
After Retract Operation with Copy Strategy

Cell phone

ambient music player

controller

[Copy Strategy]

music library

Keyboard

Laptop

Speakers

Audio

[Move Strategy]
Stretch Operation with Rebind Strategy

Cell phone

- ambient music player
- [Rebind Strategy]
- controller
- Keyboard

Laptop

- Another resilient actor providing same service

Diagram:
- Keyboard
- Speakers
- Music library
- Audio
- Copy Strategy
- Move Strategy
Retract Operation with Rebind Strategy

Cell phone

- ambient music player
- [Rebind Strategy]
- controller
- Keyboard

Laptop

- Another resilient actor providing same service
- Keyboard

[Move Strategy]
[Copy Strategy]
After Retract Operation with Rebind Strategy

Cell phone

ambient music player

controller

Keyboard

[Rebind Strategy]

music library

[Copy Strategy]

Speakers

[Move Strategy]

Laptop

Another resilient actor providing same service

Keyboard
Resolution of Strategies

Resilient strategies can be defined on resilient actors and elastic bindings.
Resolution of Strategies

Resilient strategies can be defined on resilient actors and elastic bindings.

Which resilient strategy will be applied?
Resolution of Strategies

Resilient strategies can be defined on resilient actors and elastic bindings.

<table>
<thead>
<tr>
<th>Resilient strategy on elastic binding</th>
<th>Resilient strategy on the resilient Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>standstill</td>
<td>{move, copy, rebind, standstill}</td>
</tr>
<tr>
<td>rebind</td>
<td>{move, copy, rebind, standstill}</td>
</tr>
<tr>
<td>move</td>
<td>{move, copy, rebind, standstill}</td>
</tr>
<tr>
<td>copy</td>
<td>{move, copy, rebind, standstill}</td>
</tr>
</tbody>
</table>
Resilient Actors in AmbientTalk

- AmbientTalk: Actor-based language for pervasive computing (Van Cutsem et. al, 2007)
- Mechanisms for handling network failures
- We extend AmbientTalk with four language constructs for resilient partitioning:
  
  **actor**: `resilientAs:` and **bindTo**: `resilientAs:`
  
  **stretch**: and **retract**:
Resilient Actors in AmbientTalk (cont’d)

Implementation of the ambient music player using the resilient actor model

**Definition of a resilient actor**

```plaintext
def musicLibrary := actor: {...} resilientAs: [copy];
def audio := actor: {...} resilientAs: [standstill];
def ambientMusicPlayer := actor: { |controller|
def theController := bindTo: controller resilientAs: [move];
.....
} resilientAs: [standstill];
```

```plaintext
def controller := actor: { |audio, musicLibrary|
def type musicLibraryService;
def theAudio := bindTo: audio resilientAs: [standstill];
def theMusicLib := bindTo: musicLibrary resilientAs: [rebind(musicLibraryService)];
} resilientAs: [move];
```
Implementation of the ambient music player using the resilient actor model

Definition of a resilient actor

```plaintext
def musicLibrary := actor: {....} resilientAs: [copy];
def audio := actor: {...} resilientAs: [standstill];
```

```plaintext
def ambientMusicPlayer := actor: { |controller|
def theController := bindTo: controller resilientAs: [move];
....
resilientAs: [standstill];
```

```plaintext
def controller := actor: { [audio, musicLibrary]}
deftype musicLibraryService;
def theAudio := bindTo: audio resilientAs: [standstill];
def theMusicLib := bindTo: musicLibrary resilientAs: [rebind(musicLibraryService)];
}
resilientAs: [move];
```
Implementation of Resilient Actors

Resilient Actor

Resilient actor’s behavior

Meta Level

Base Level

mirror

nil object

Resilient strategy

Meta Level

Base Level

mirror

isolate object encapsulating behavior
Implementation of Elastic Binding

A
Proxy object
Resilient Actor

B
Resilient Actor

Conceptual elastic binding
Object reference

mirror
Strategy object
Base Level
Meta Level
Resilient Actor
Extensible Implementation

- New Resilient Strategies by Extension

  *e.g.* Proactive state replication as an extension of the copy strategy

```plaintext
def replicationStrategy := extend: copyStrategy with: {
  def stretch: location {
    super^stretch: location;
    whenever: seconds(10) elapsed: {
      //update resilient actor state
      }
    }
  }
```
Requirements and Solution Revisited

- **Runtime Application Partitioning**
  - Resilient Actors

- **User Controlled Application Partitioning**
  - *Stretch* Operation

- **Retractable Application Partitioning**
  - *Retract* Operation

- **Resilient to Network Failures**
  - Automatic Retraction
Future Work

- Identifying more Resilient Strategies
- Context-Dependent Selection of Resilient Strategies
- Integration with Service Discovery
- Applying the resilient actor model to large pervasive applications
Conclusion

- Need for Resilient Partitioning of Pervasive Computing Services
- The Resilient Actor Model
  - Resilient actors
  - Elastic bindings
  - Resilient strategies
- Extensible Implementation (Resilient Strategies)