Resilient Partitioning of Pervasive Computing Services

Engineer Bainomugisha

Promotor:
Prof. Dr. Wolfgang De Meuter

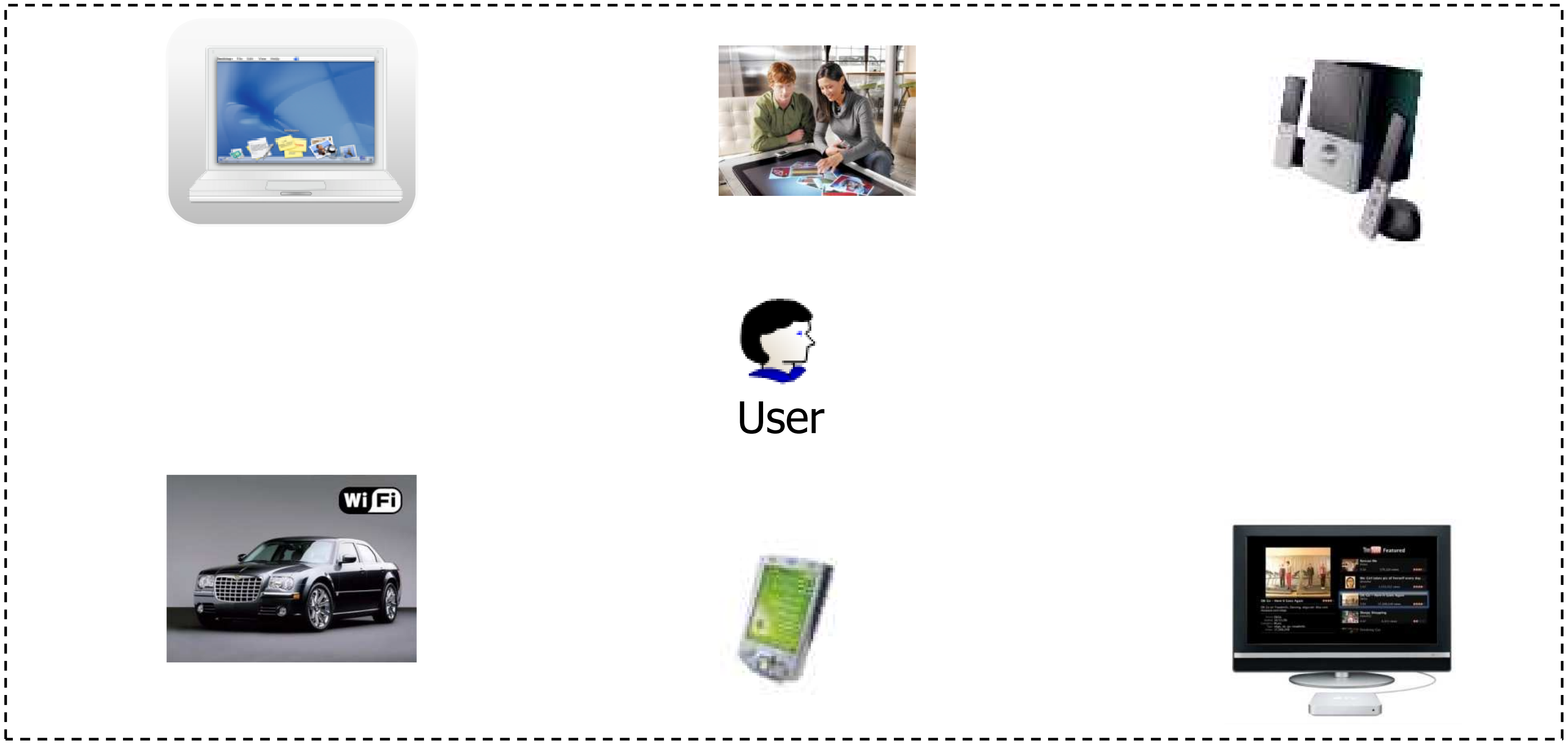
Advisors:
Jorge Vallejos
Elisa Gonzalez Boix

Programming Technology Lab
Vrije Universiteit Brussel
Brussels, Belgium

Master Thesis, Defense 08 Sept. 2008

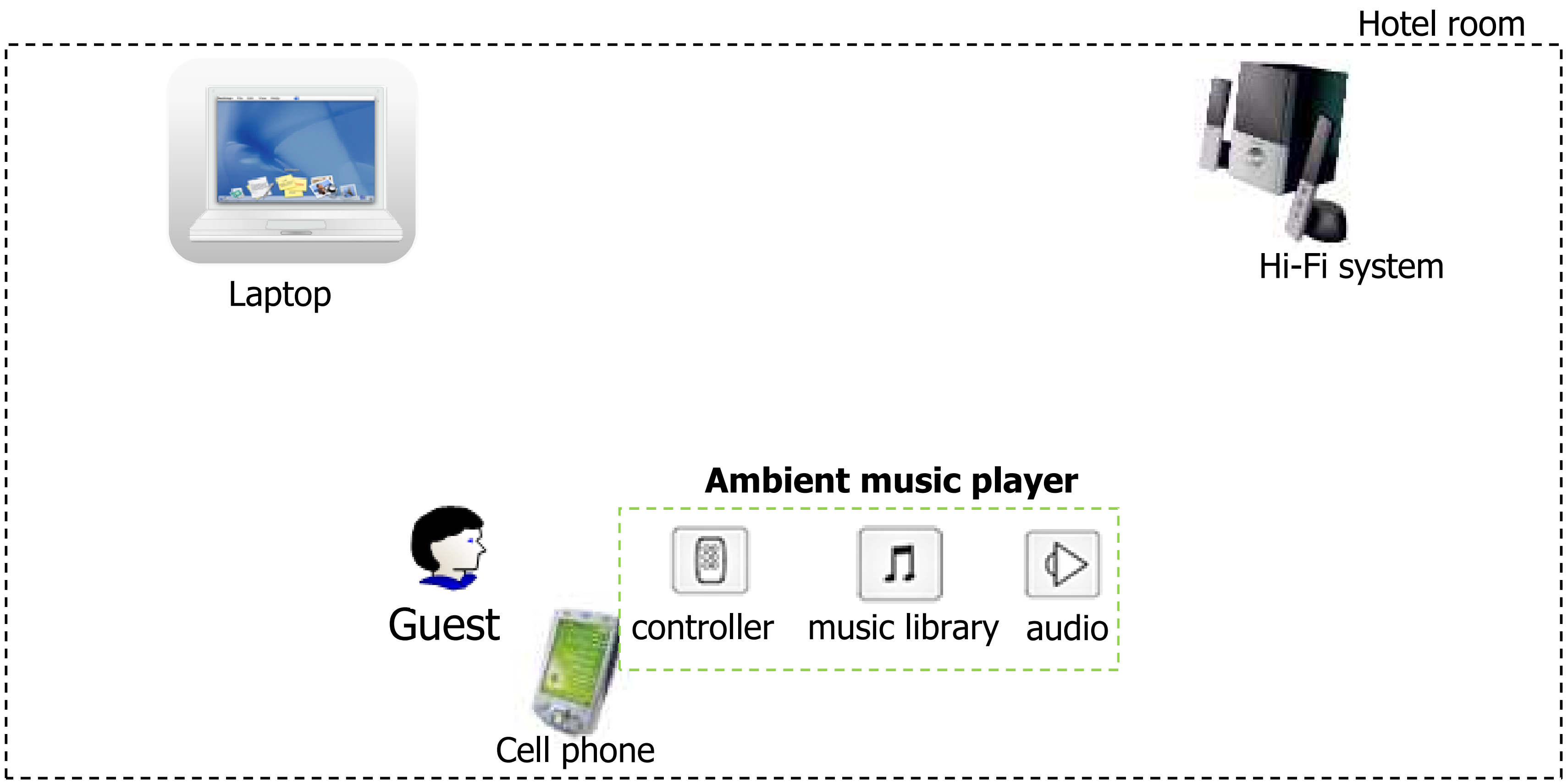
Pervasive Computing Environment

Computational power is available everywhere (Weiser, 1993)



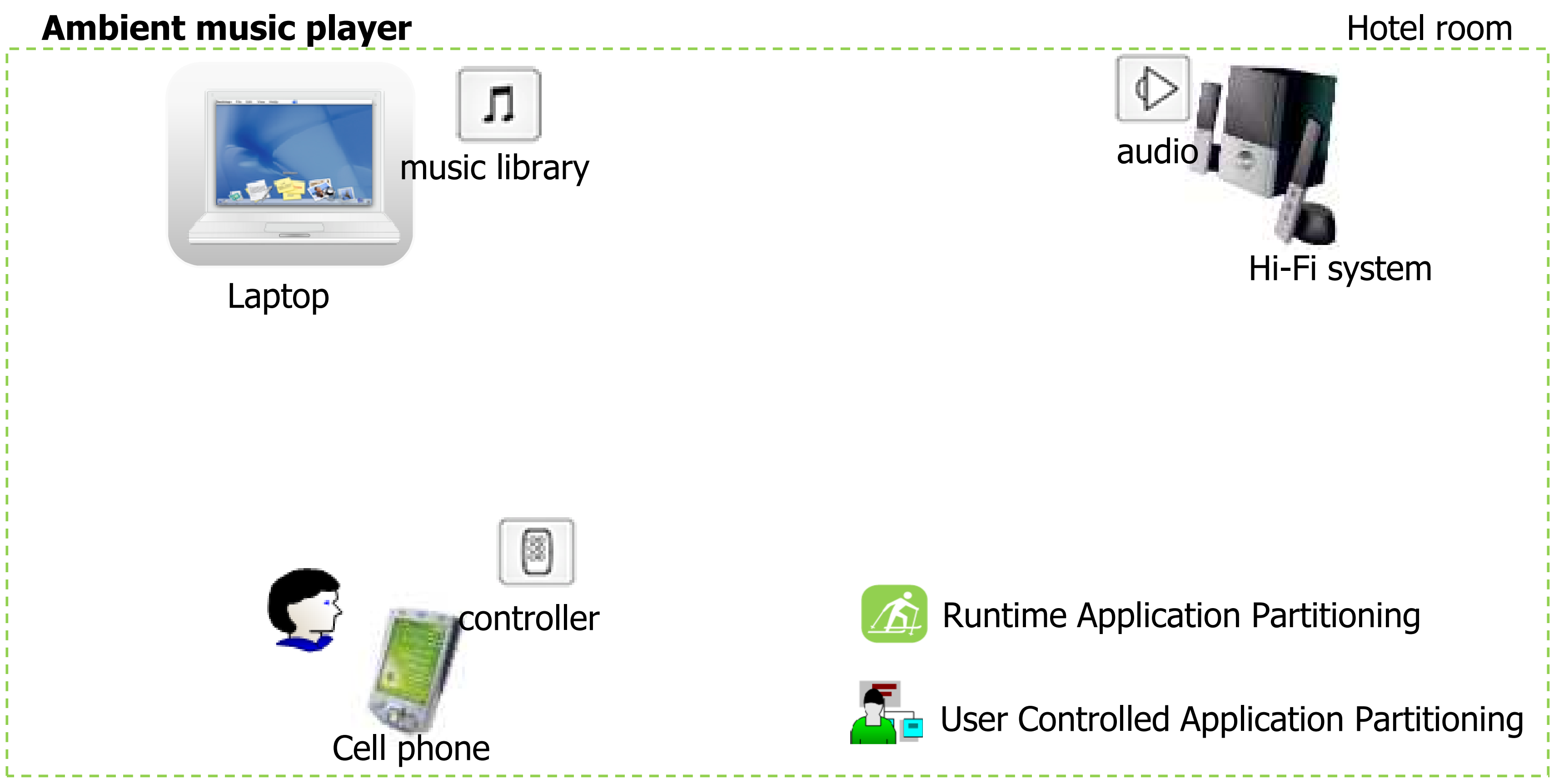
Partitioning of Pervasive Computing Services

Scenario: Hotel's Ambient Services



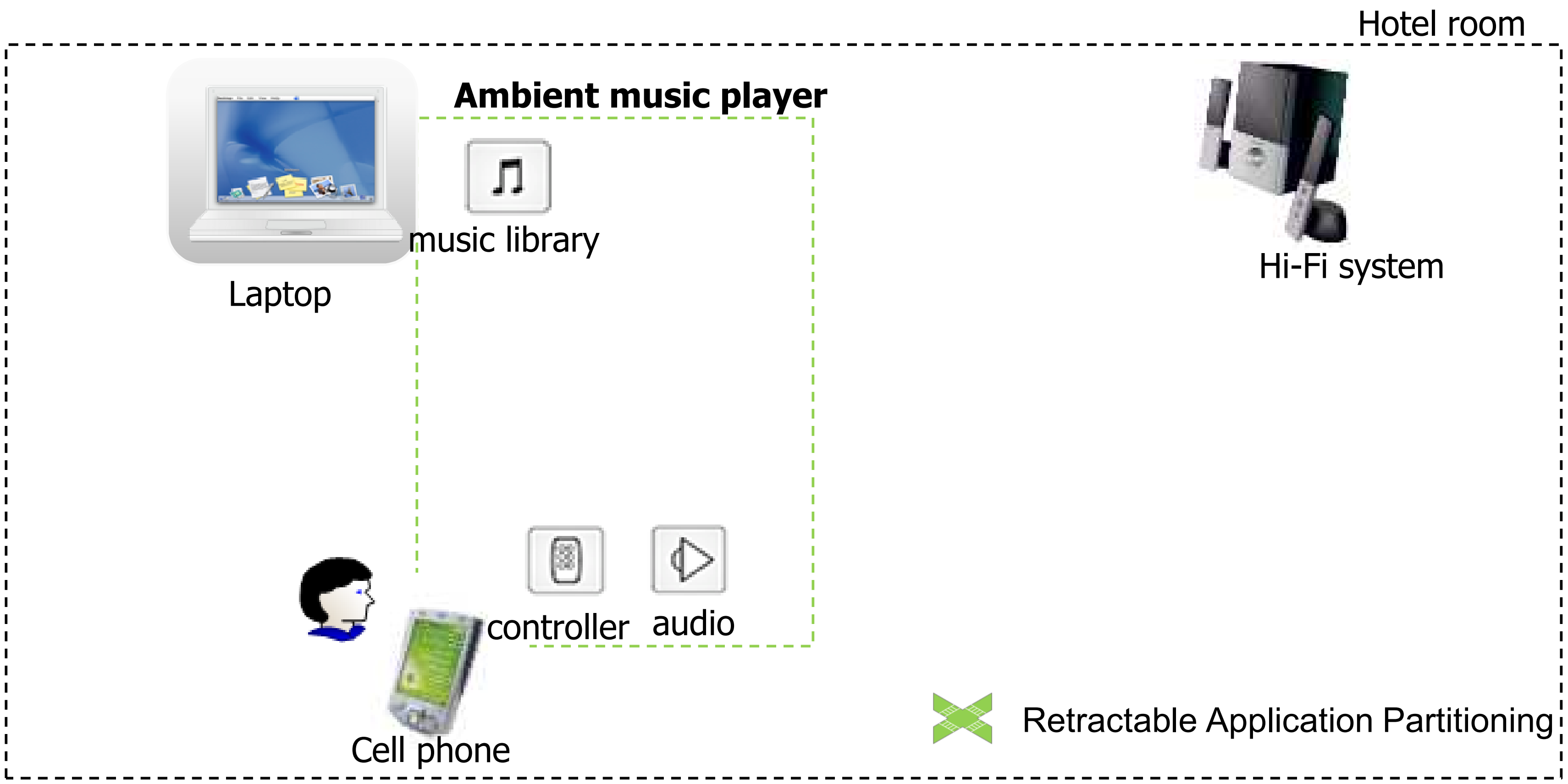
Partitioning of Pervasive Computing Services

An application can be decomposed to run on multiple devices

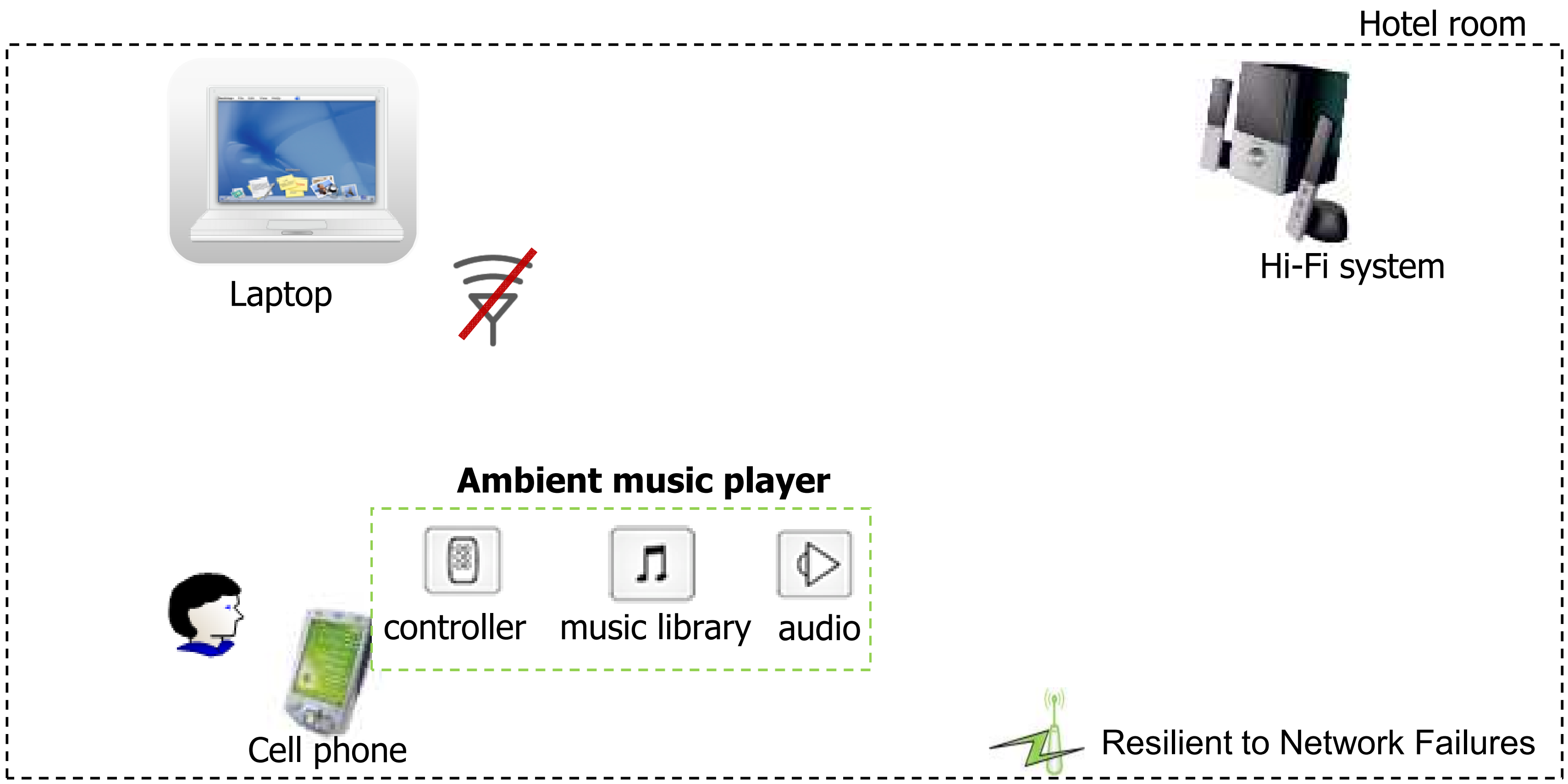


Partitioning of Pervasive Computing Services

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



Partitioning of Pervasive Computing Services



Resilient Service Partitioning

- Runtime Application Partitioning
- User Controlled Application Partitioning
- Retractable Application Partitioning
- Resilient to Network Failures

Survey of Related Work for Service Partitioning

	Runtime Partitioning	User Controlled	Retractable	Resilient to Network Failures
Object-oriented partitioning				
J-Orchestra	✗ (compilation)	✗ (programmer)	✗	✗
Addistant	✗ (compilation)	✗ (programmer)	✗	✗
JavaParty	✓	✗ (load-balancing)	✗	✗
Doorastha	✓	✗ (runtime system)	✗	✗
Component-oriented and agent-oriented partitioning				
Coign	✗ (compilation)	✗ (algorithm)	✗	✗
AdJava	✓	✗ (load-balancing)	✗	✗
Hydra	✓	✓	✓	✗

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

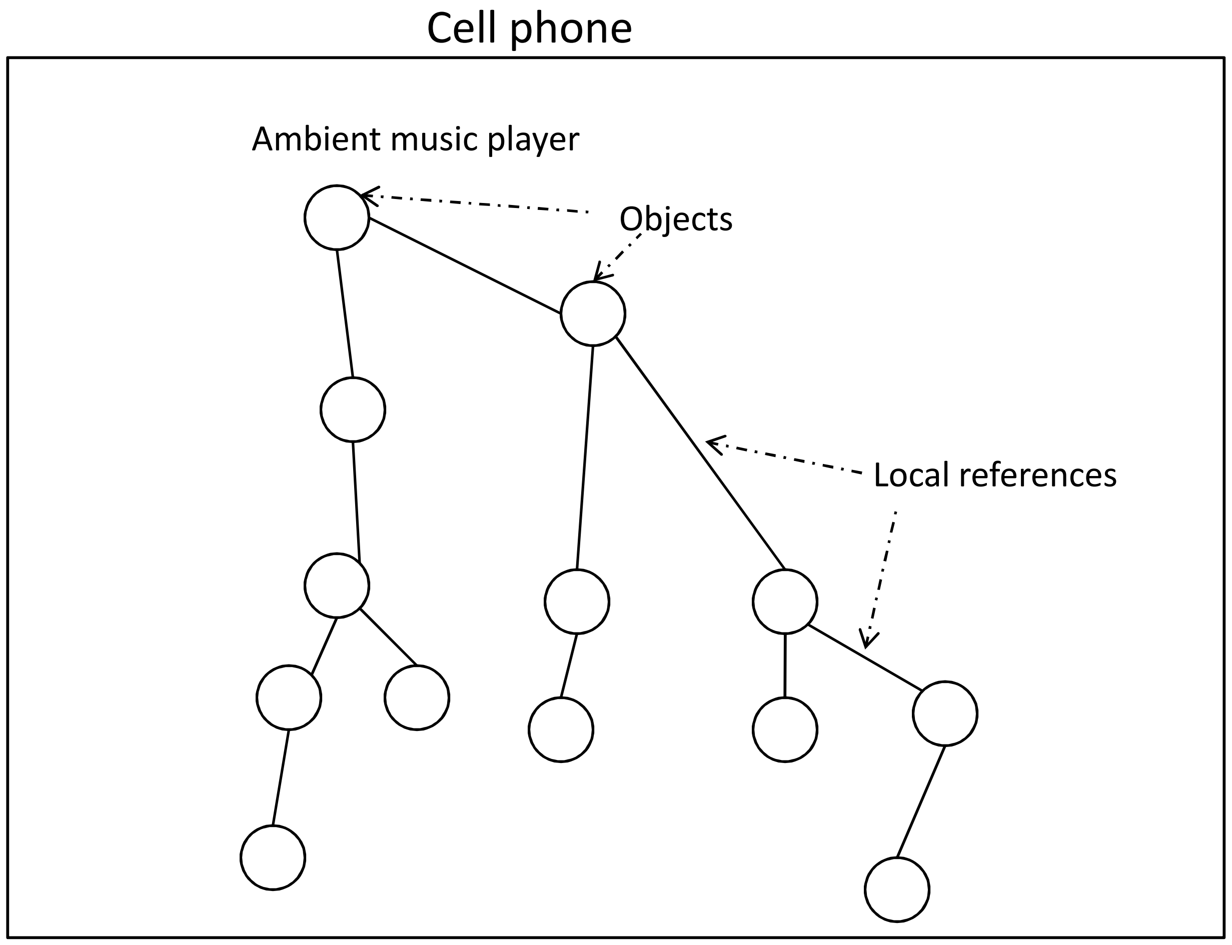
Two partitioning operations:

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

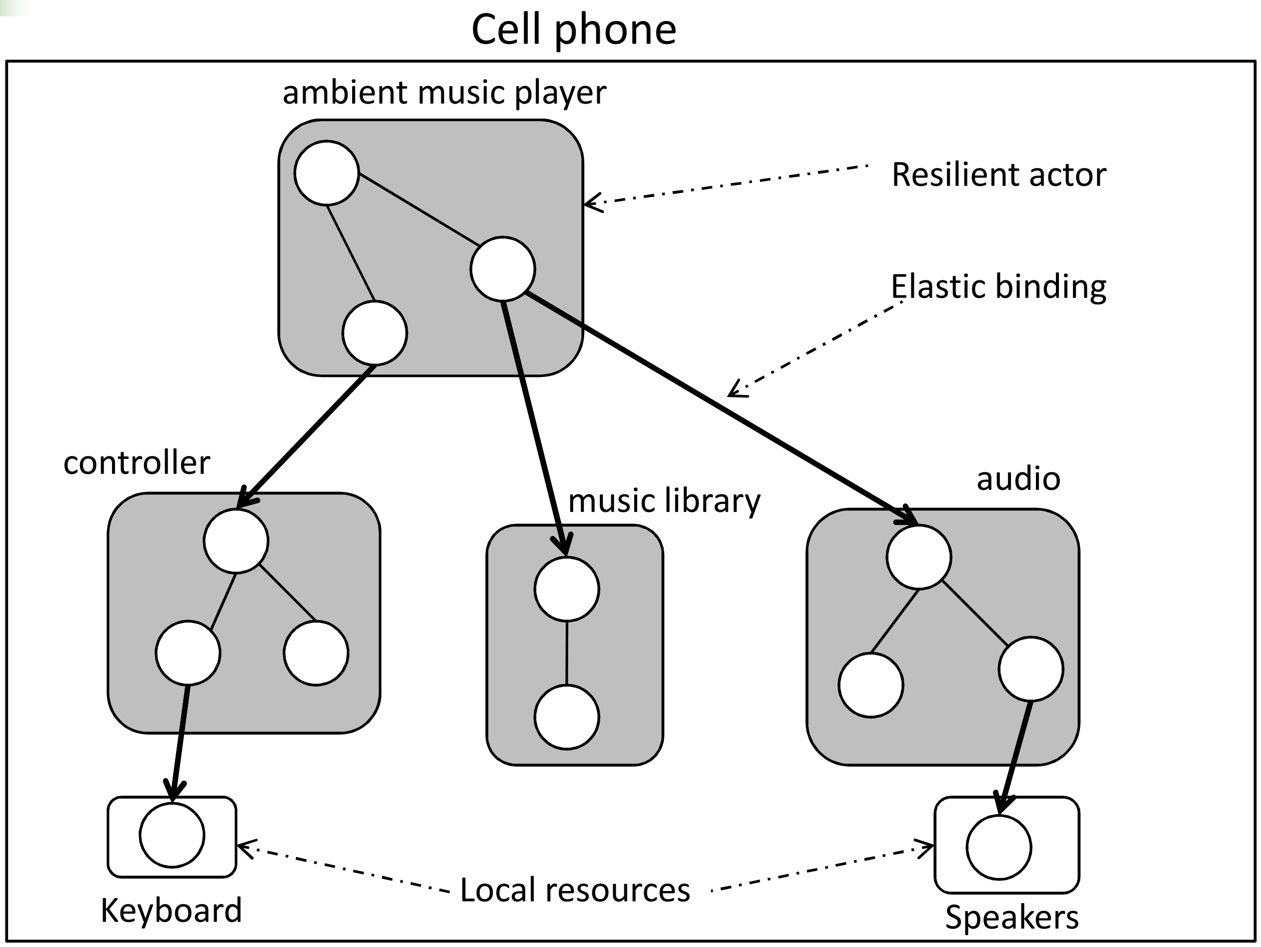
Resilient Strategies:

- that specify different behaviors of the partitioning operations

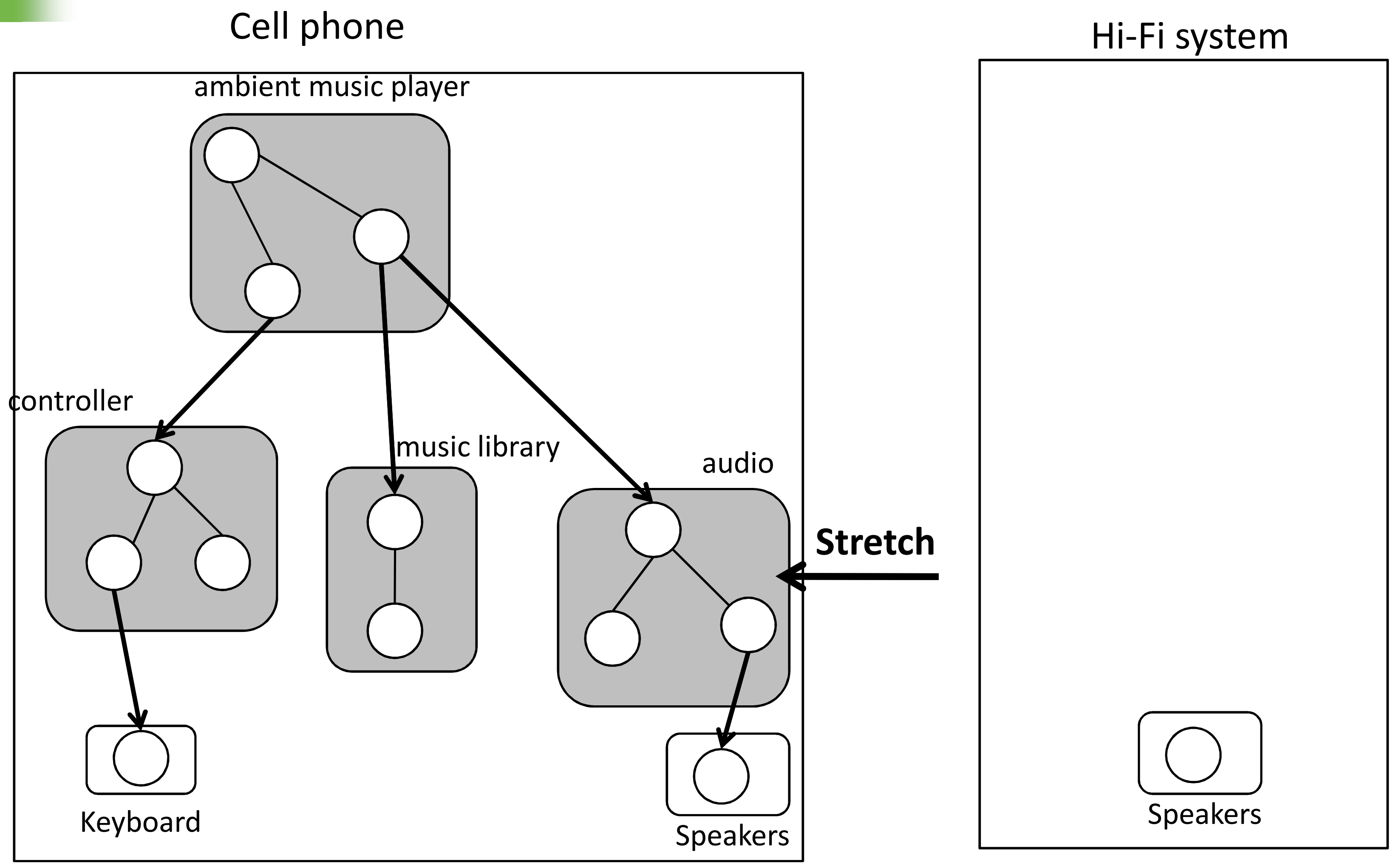
Service Partitioning



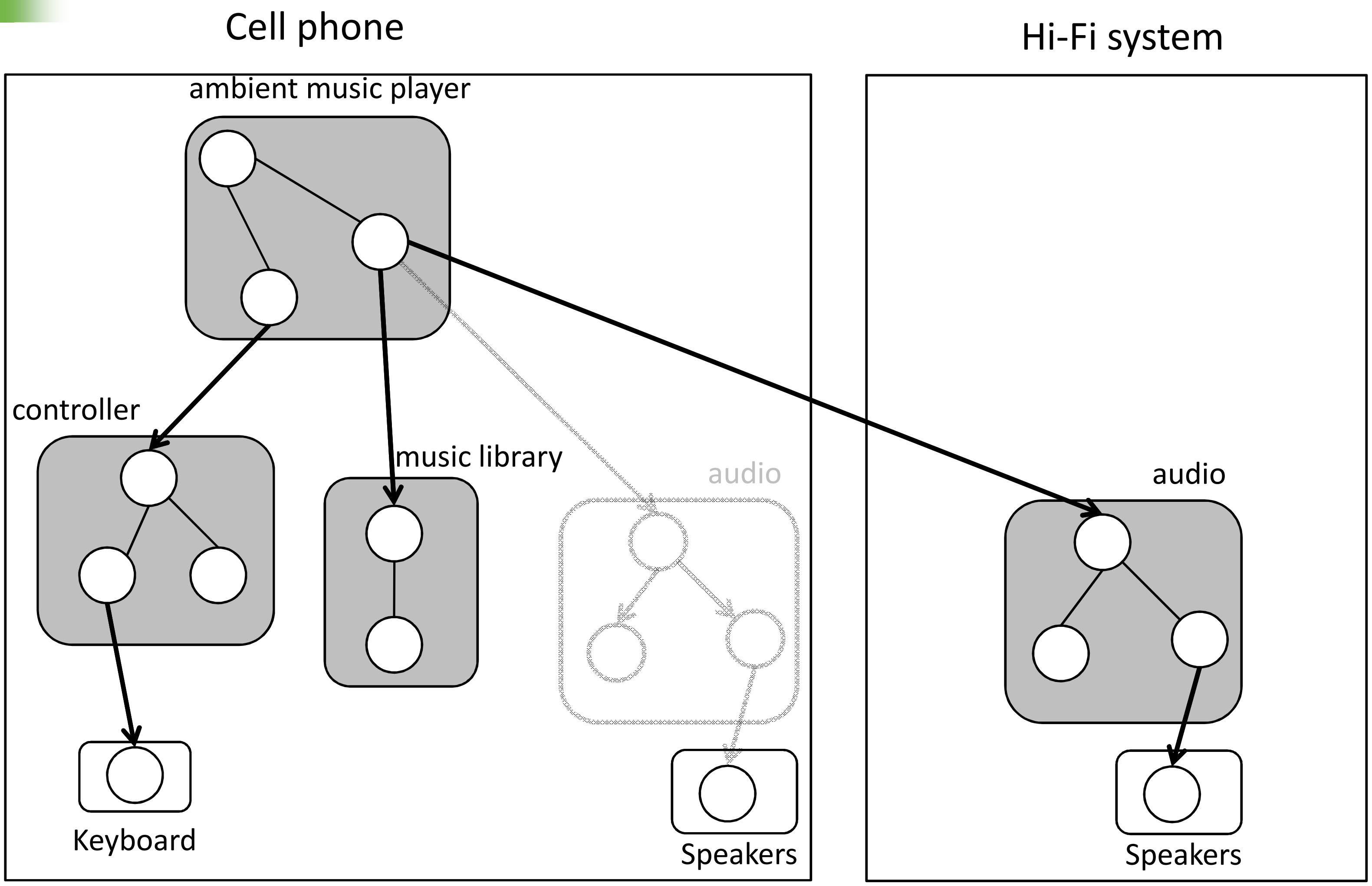
Service Partitioning



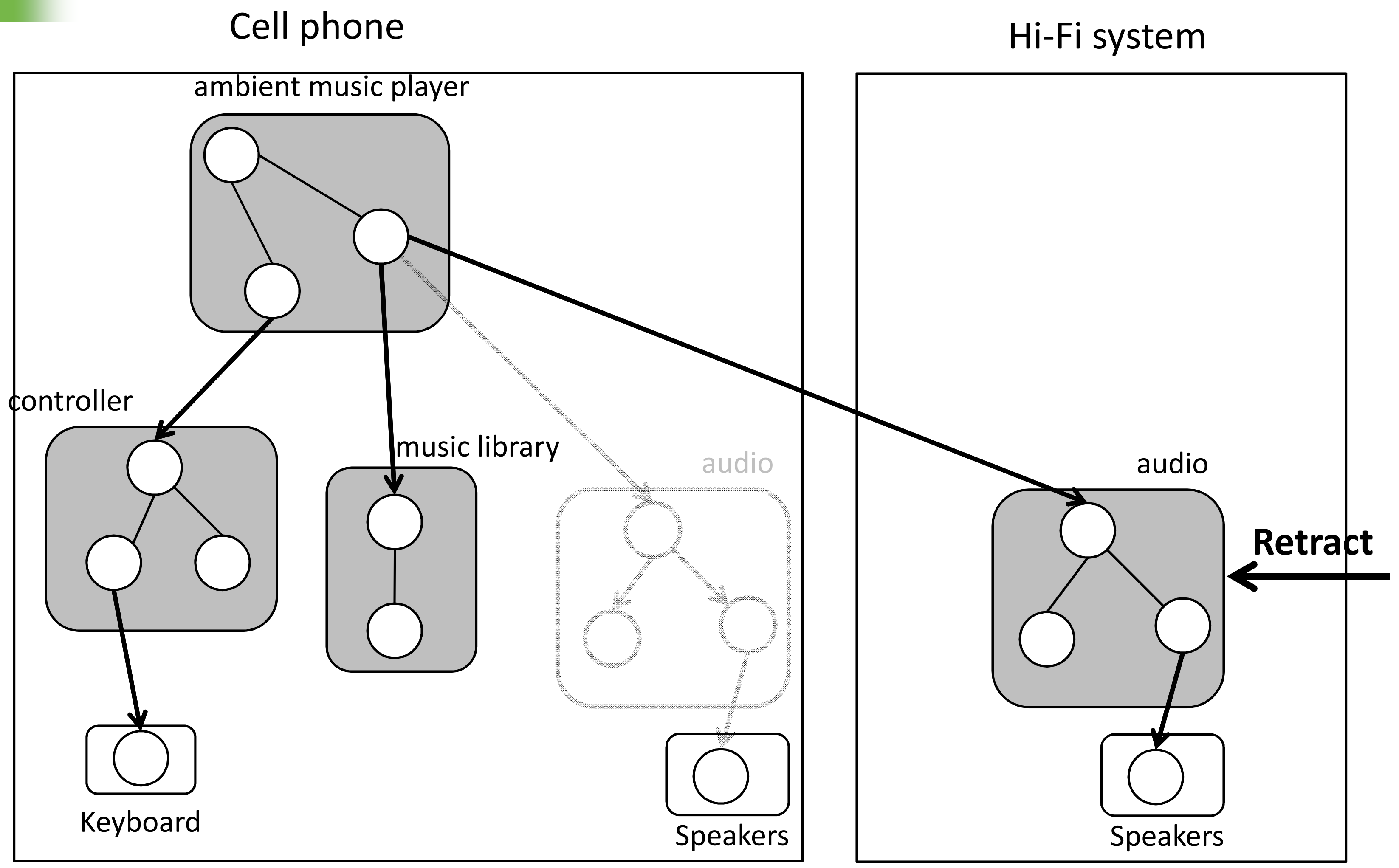
Service Partitioning



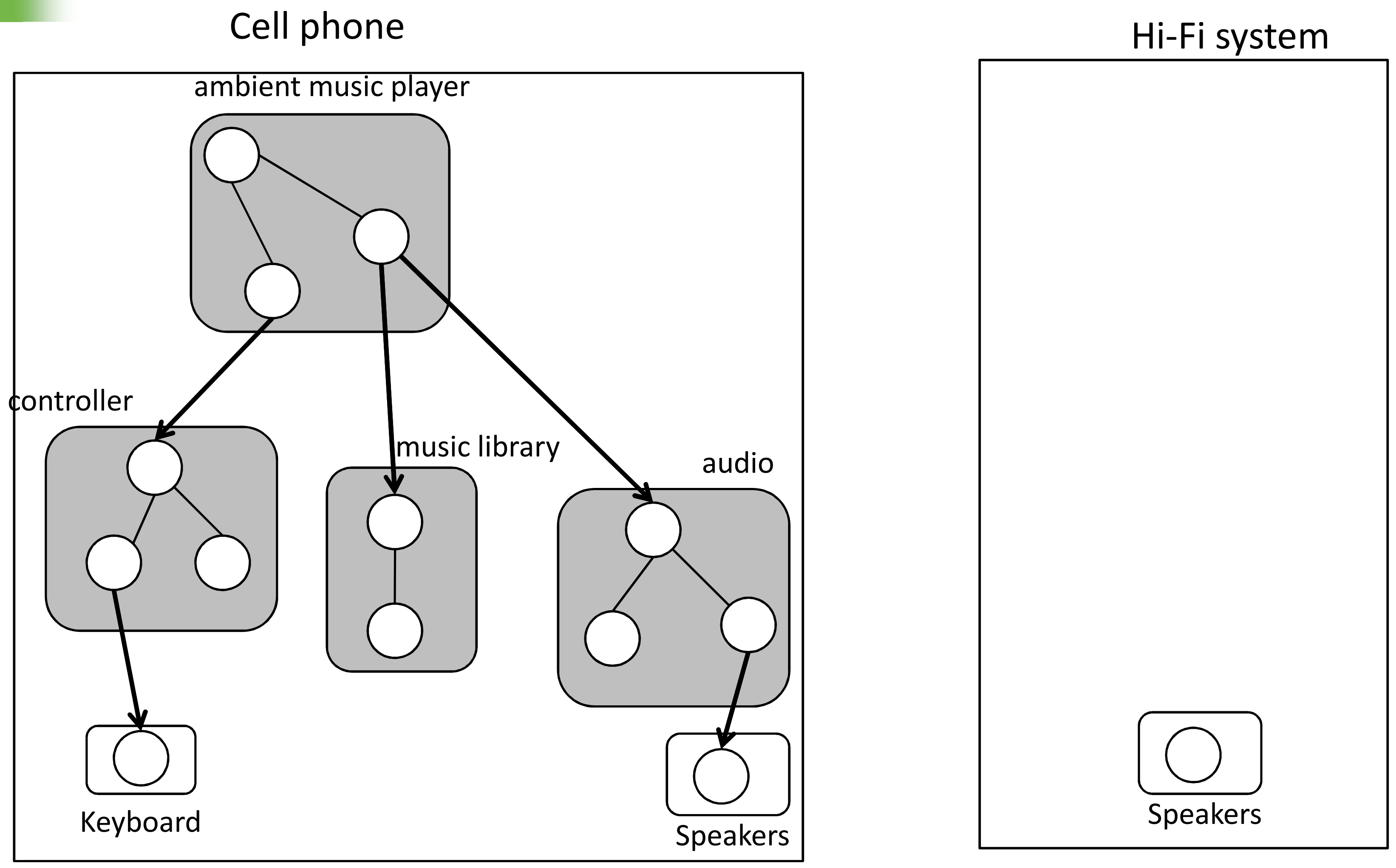
Manual Retraction



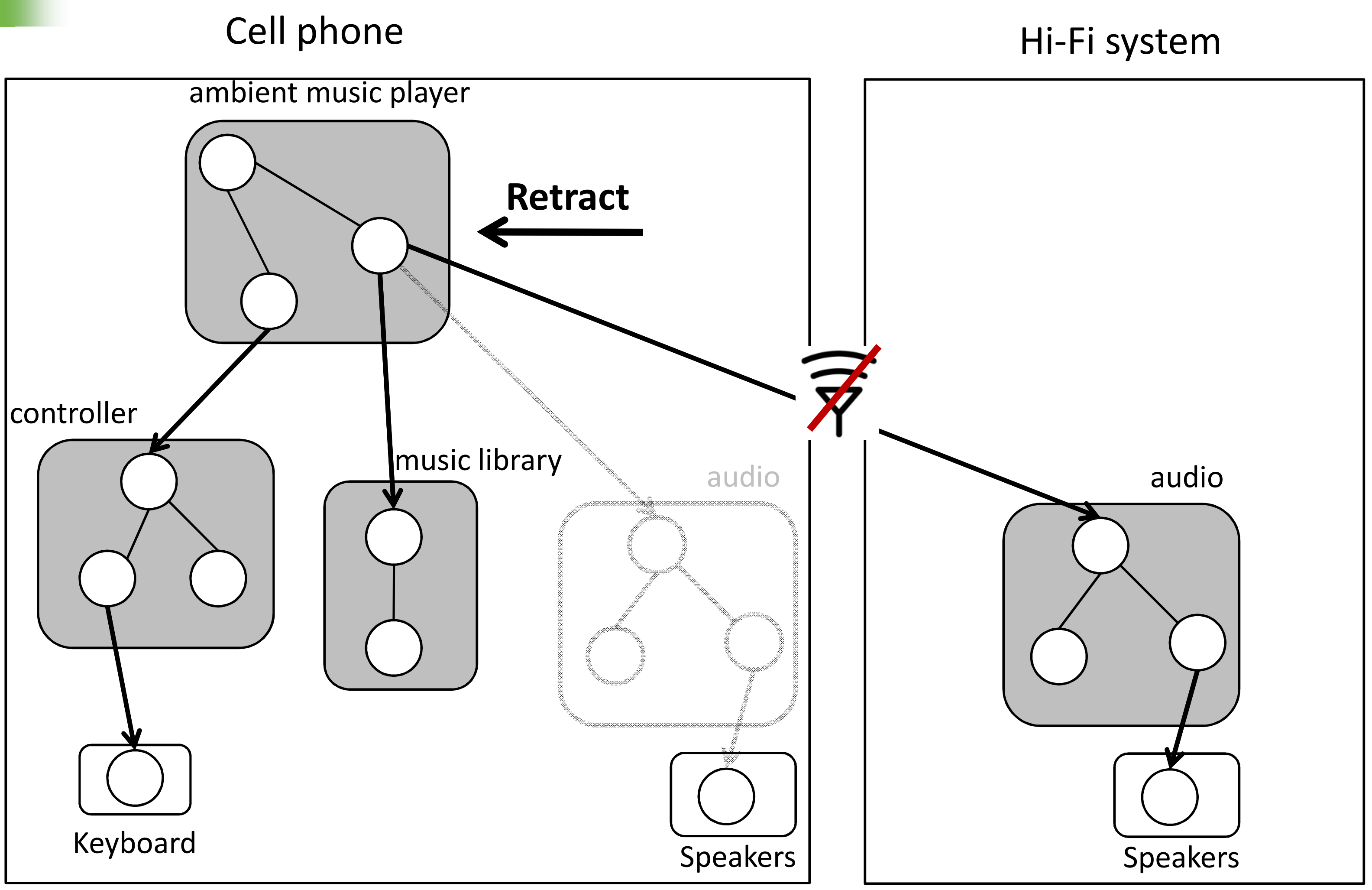
Manual Retraction



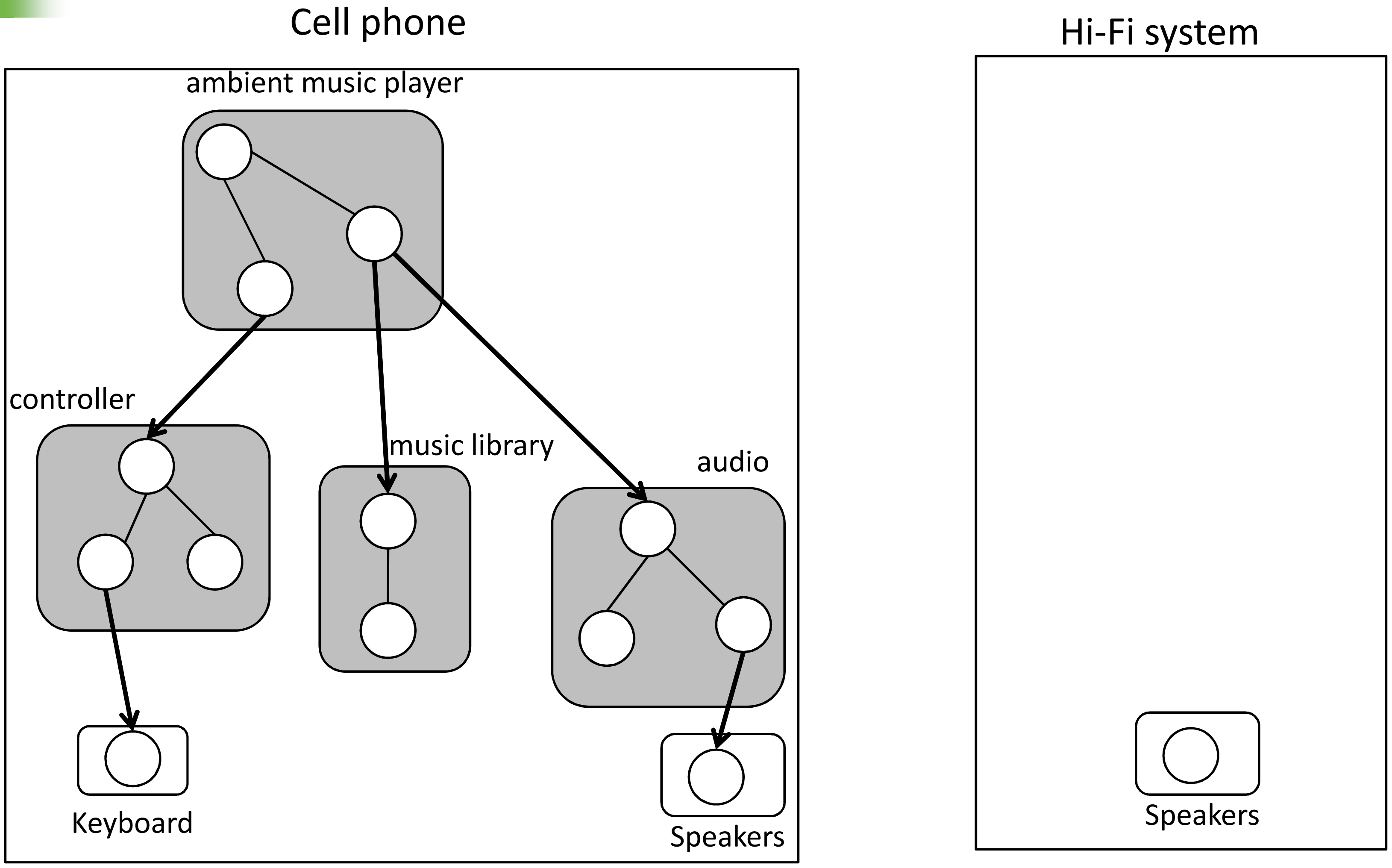
Manual Retraction



Automatic Retraction

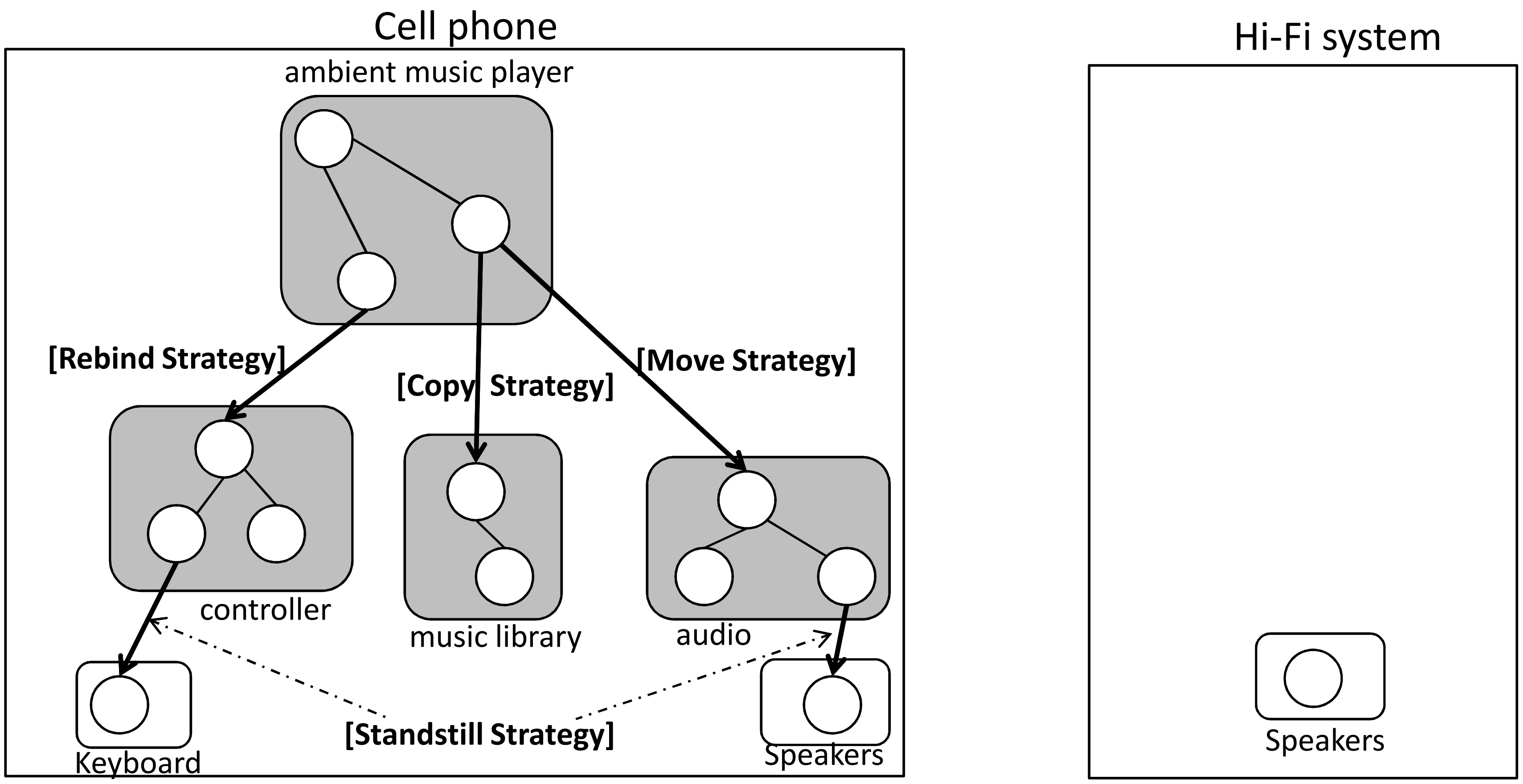


Automatic Retraction



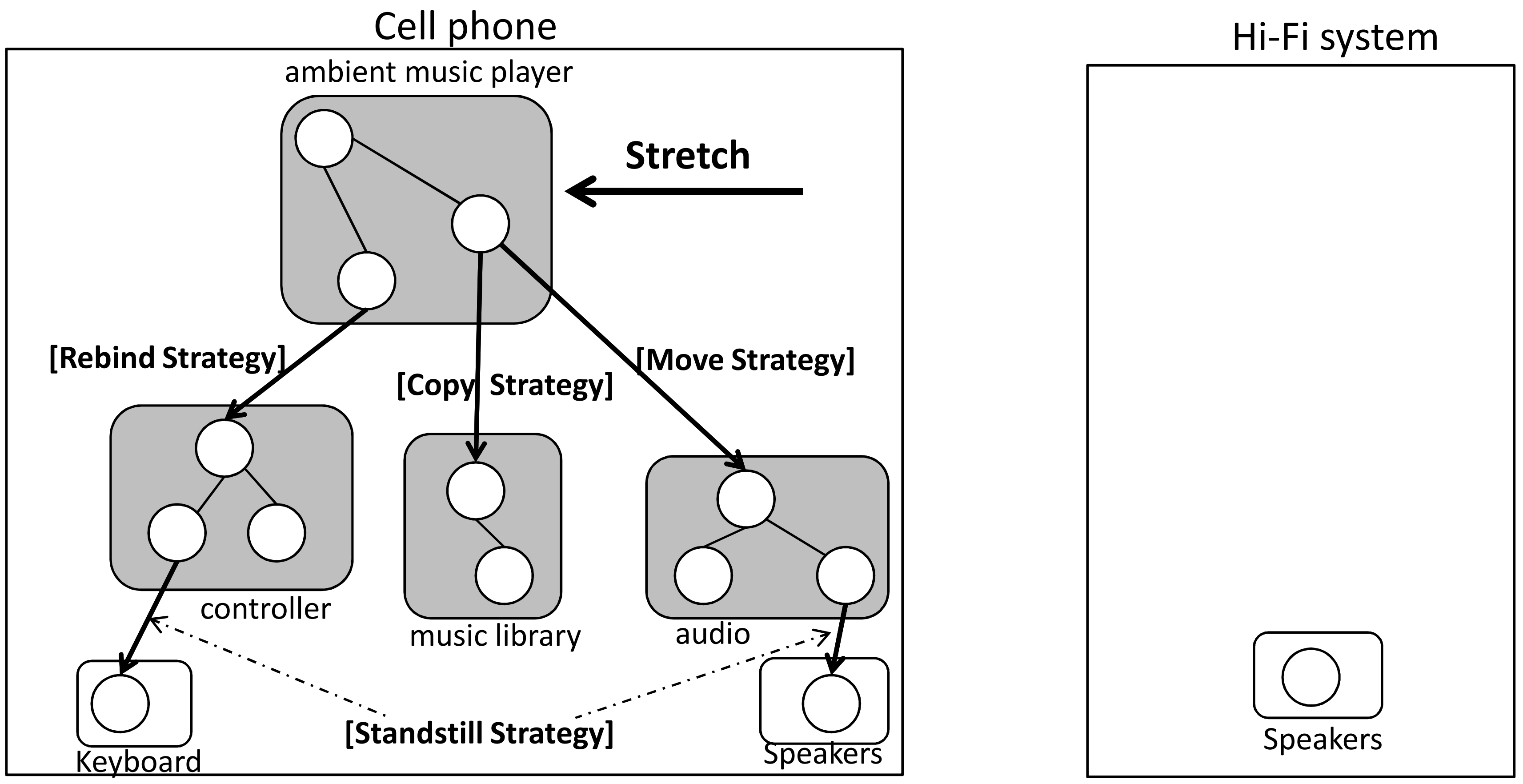
Resilient Strategies

Stretch and Retract operations can have different implementations

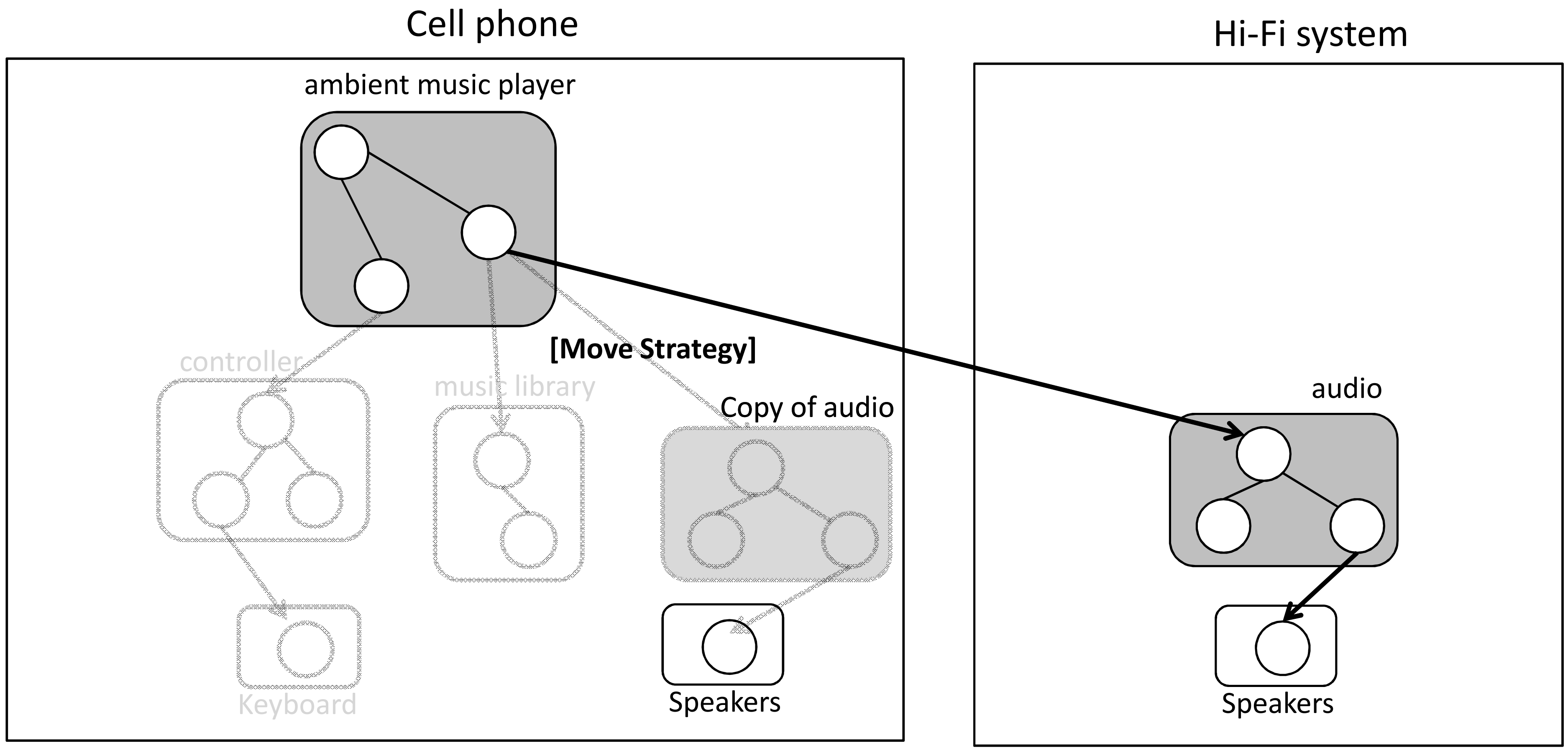


Resilient Strategies

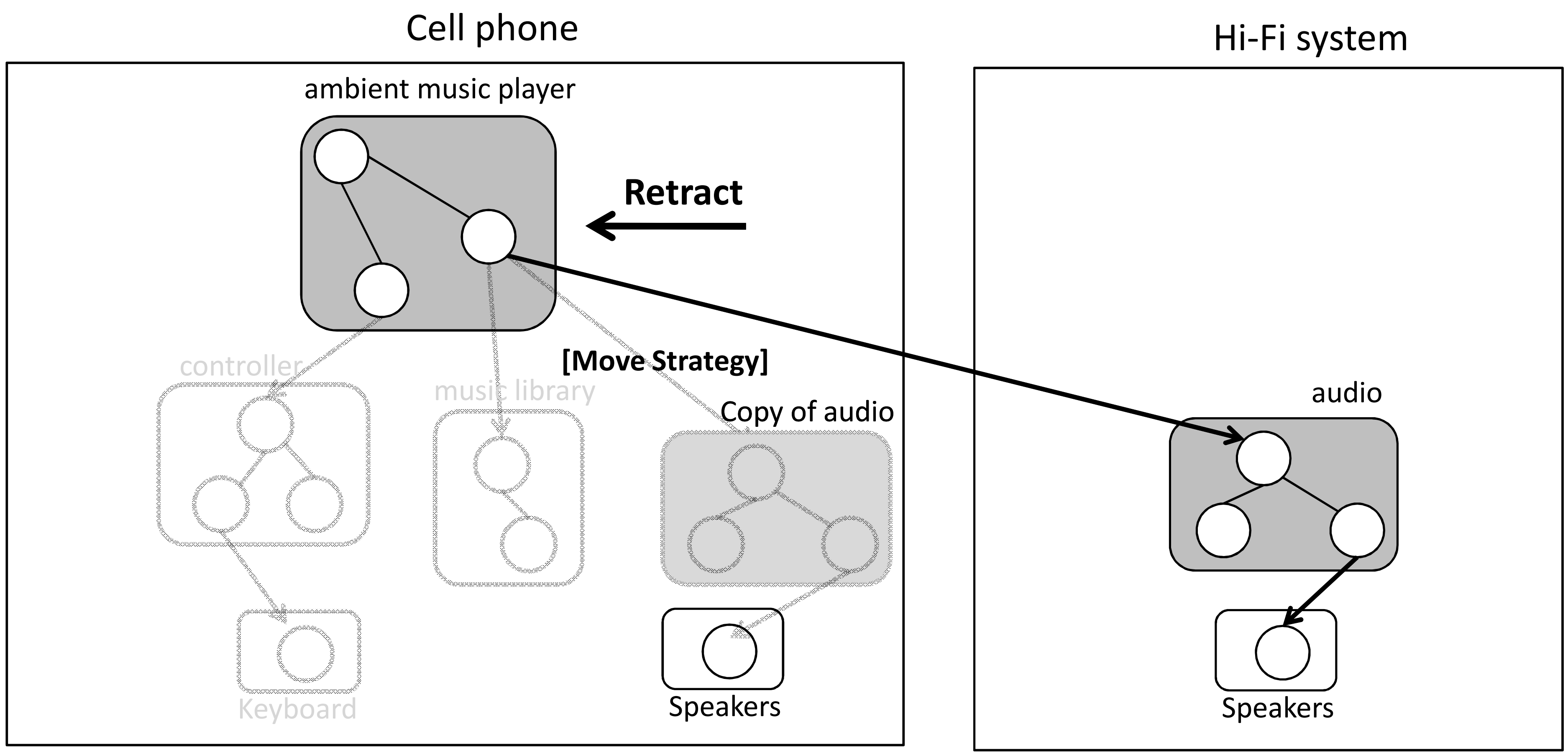
Propagation: Partitioning operations proceed through elastic bindings



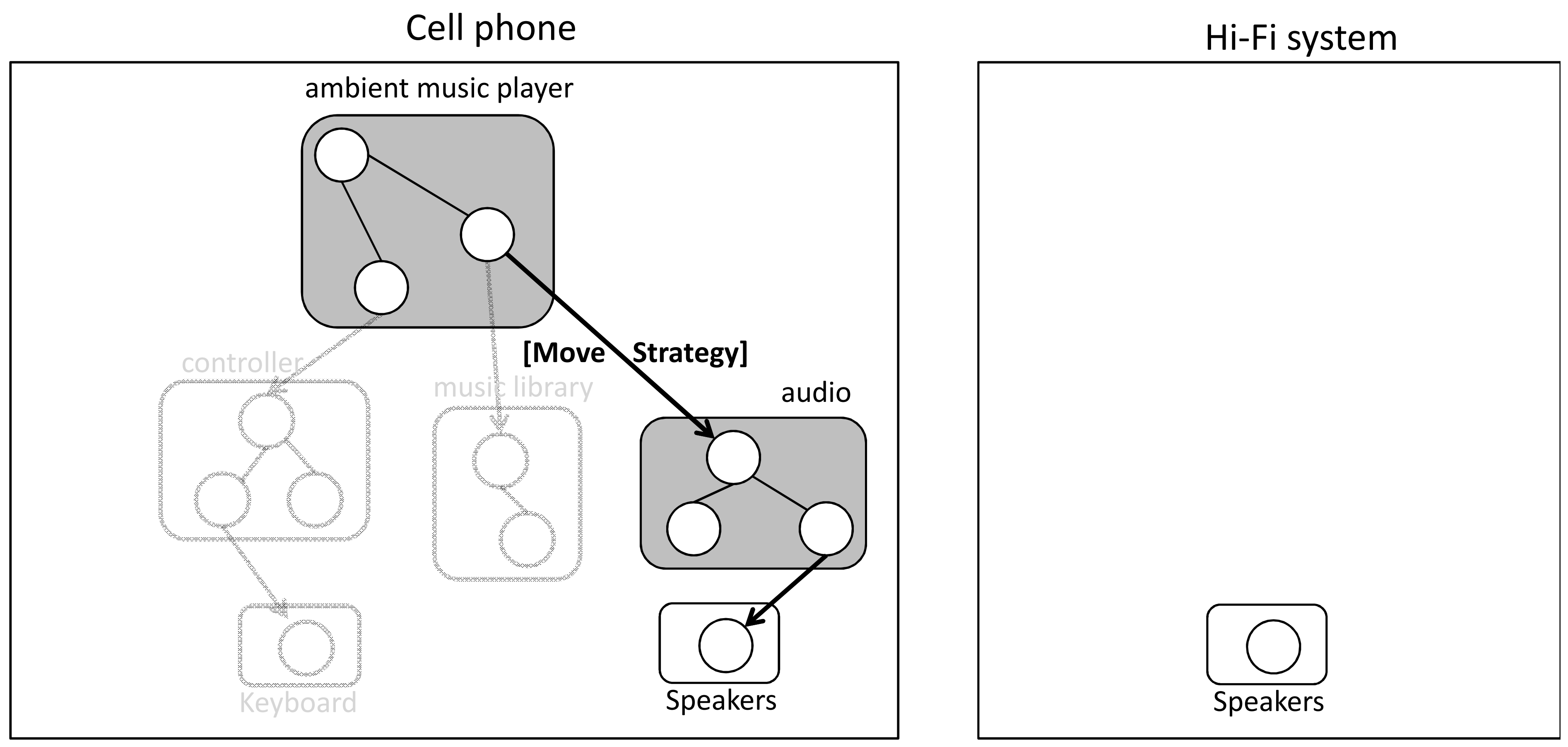
Stretch Operation with Move Strategy



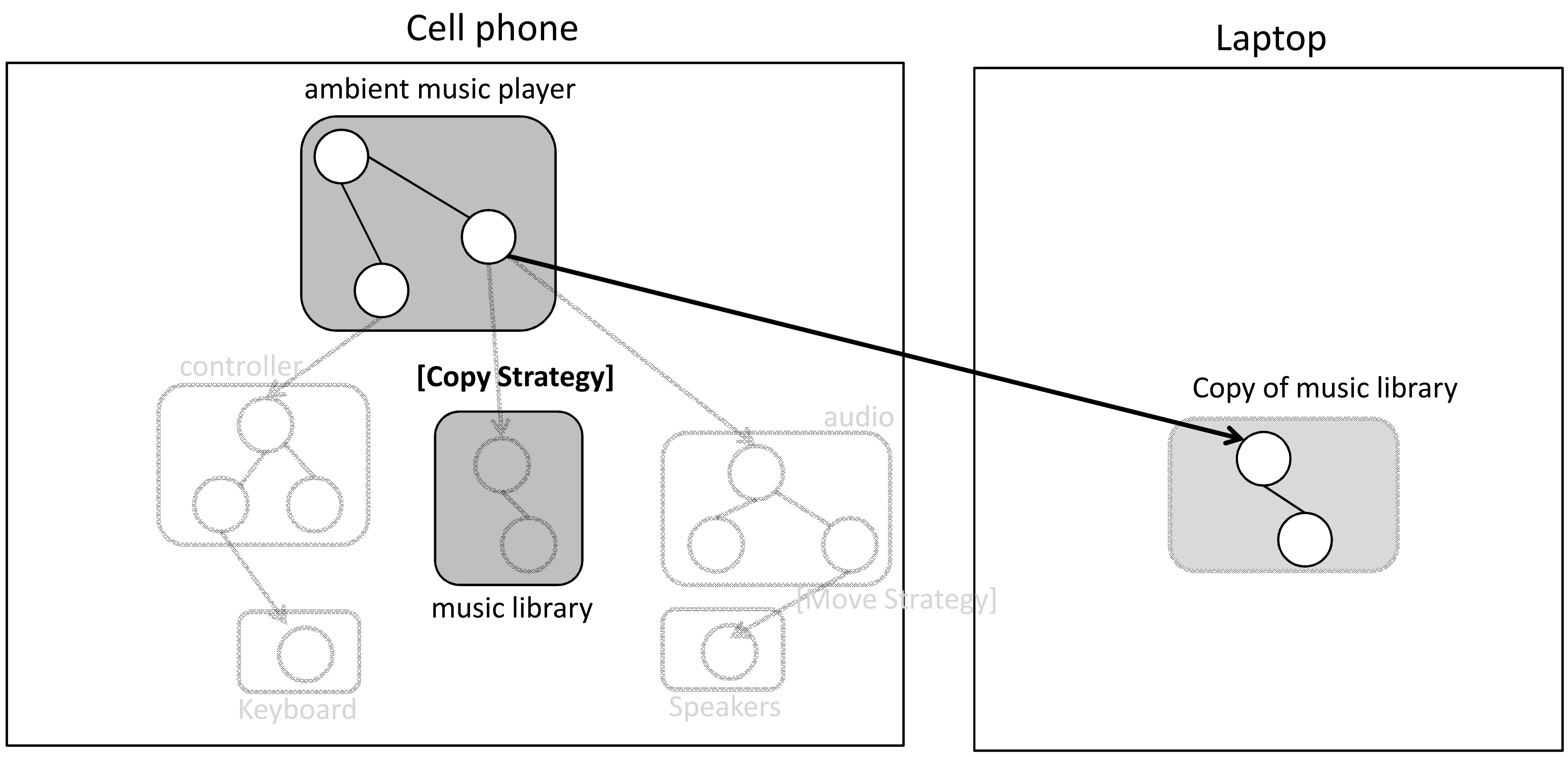
Retract Operation with Move Strategy



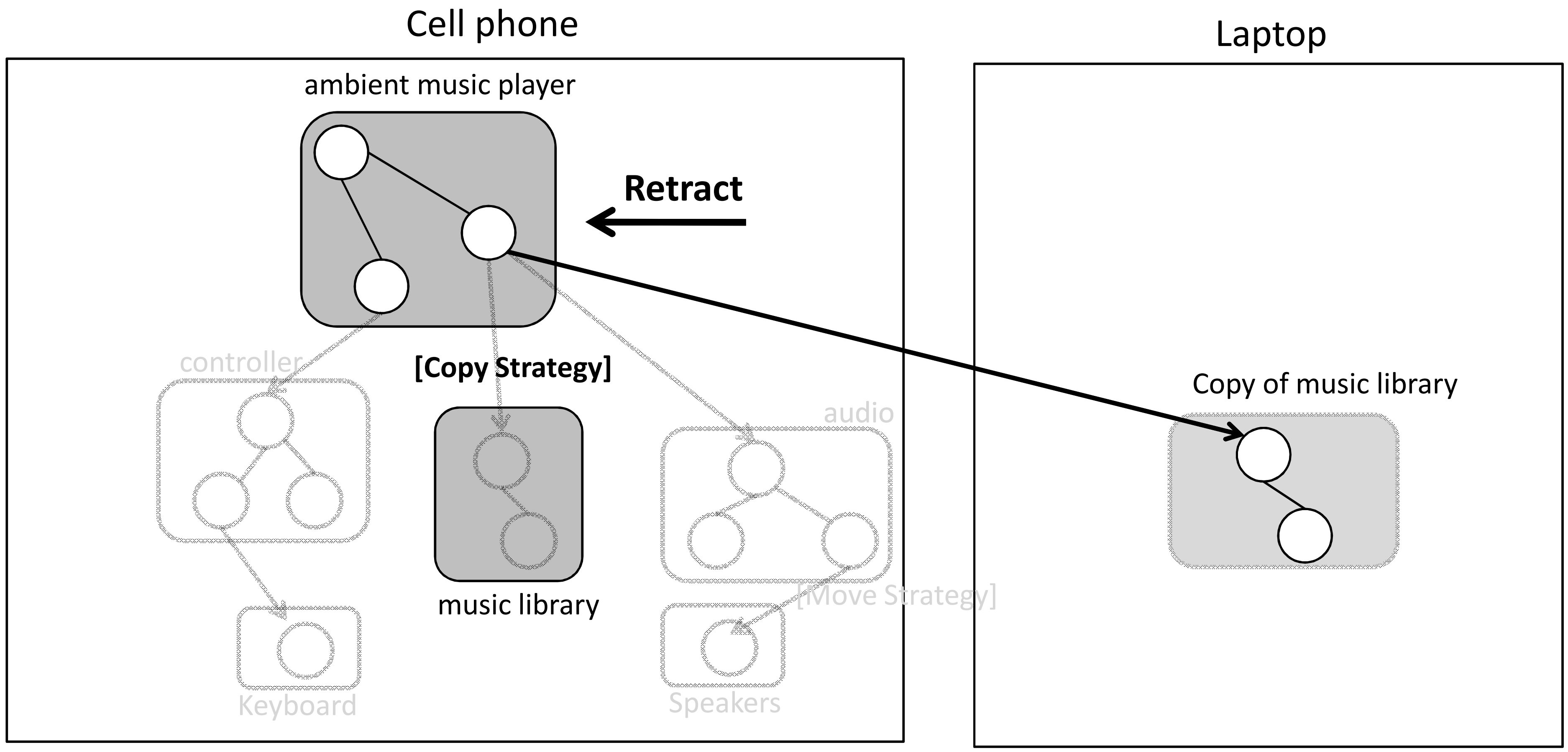
After Retract Operation with Move Strategy



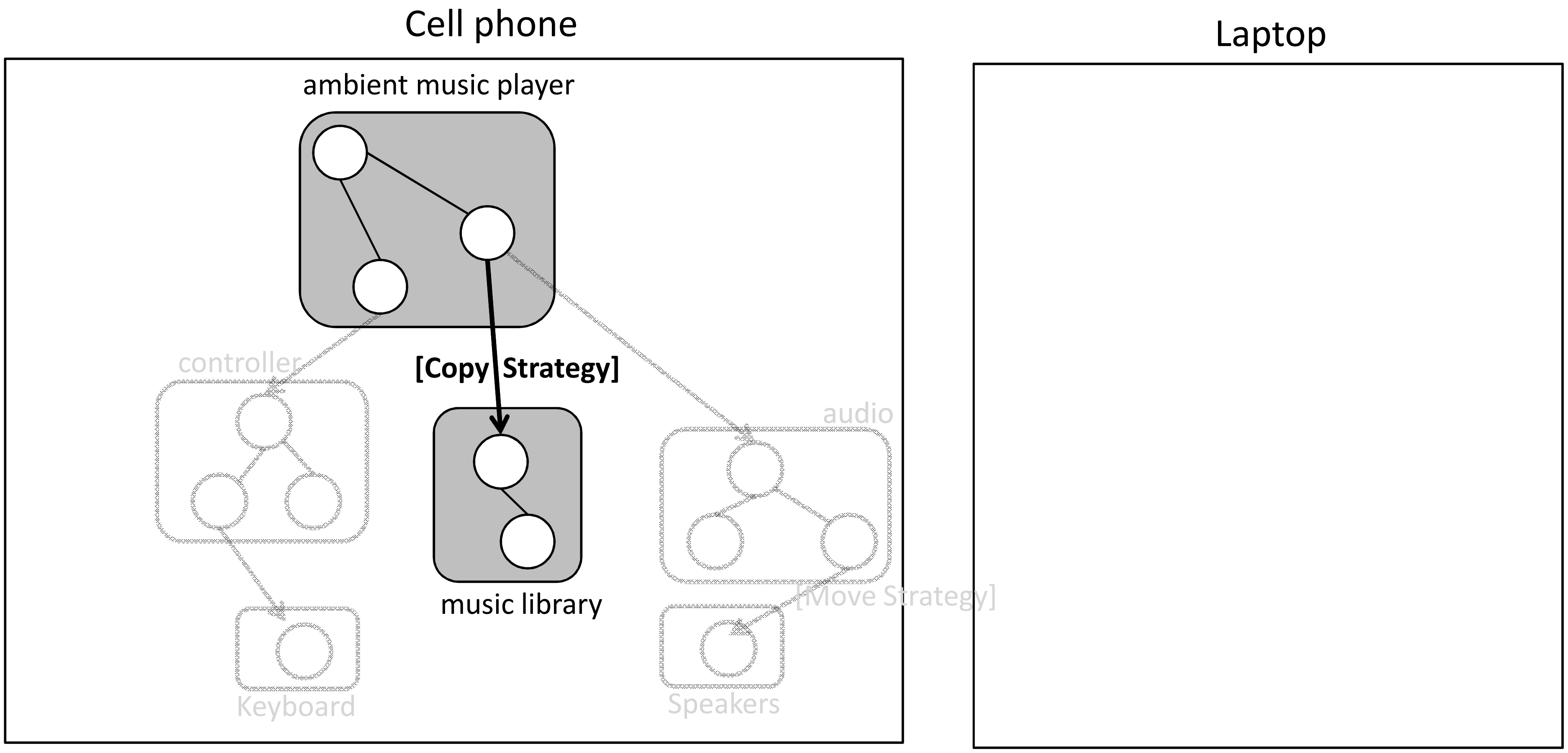
Stretch Operation with Copy Strategy



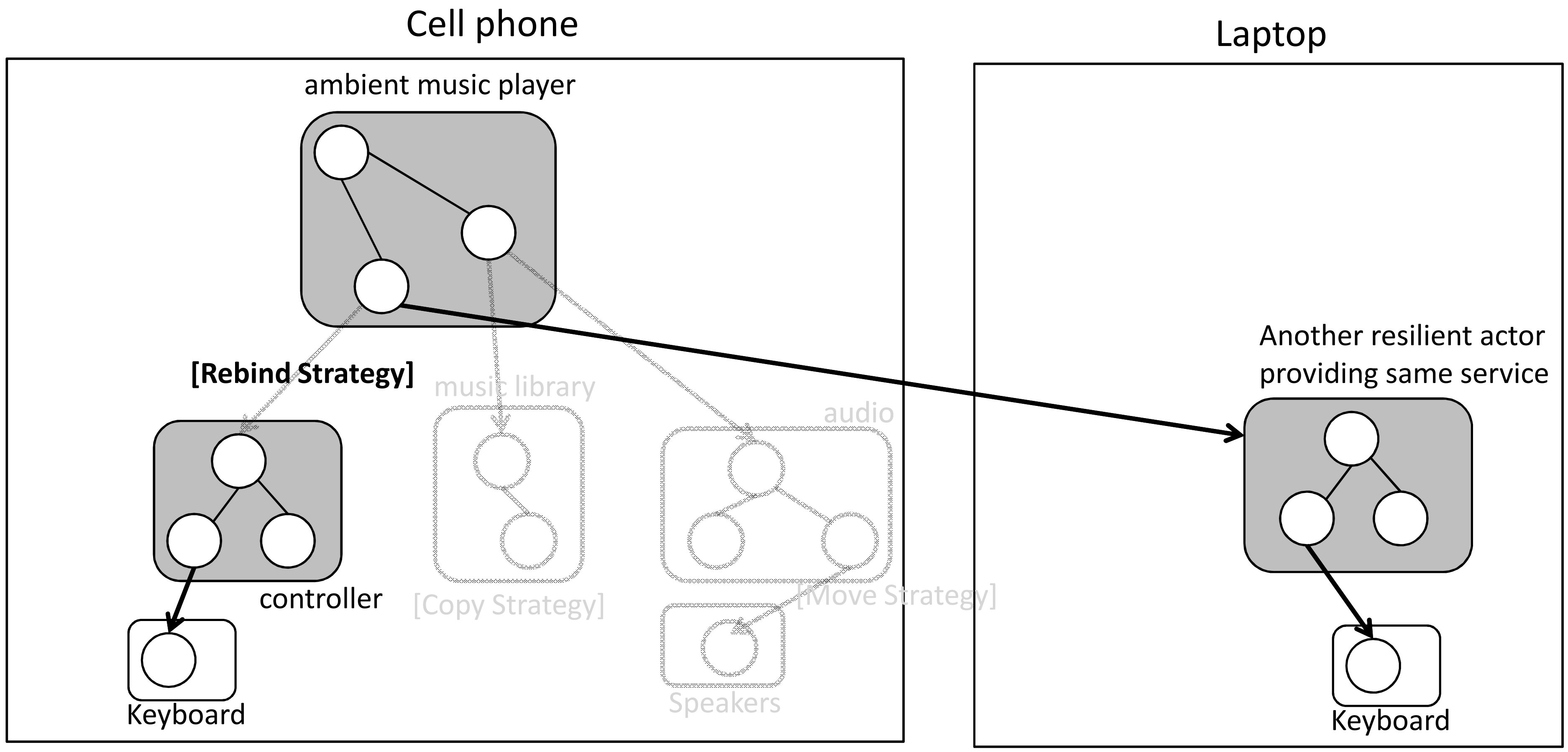
Retract Operation with Copy Strategy



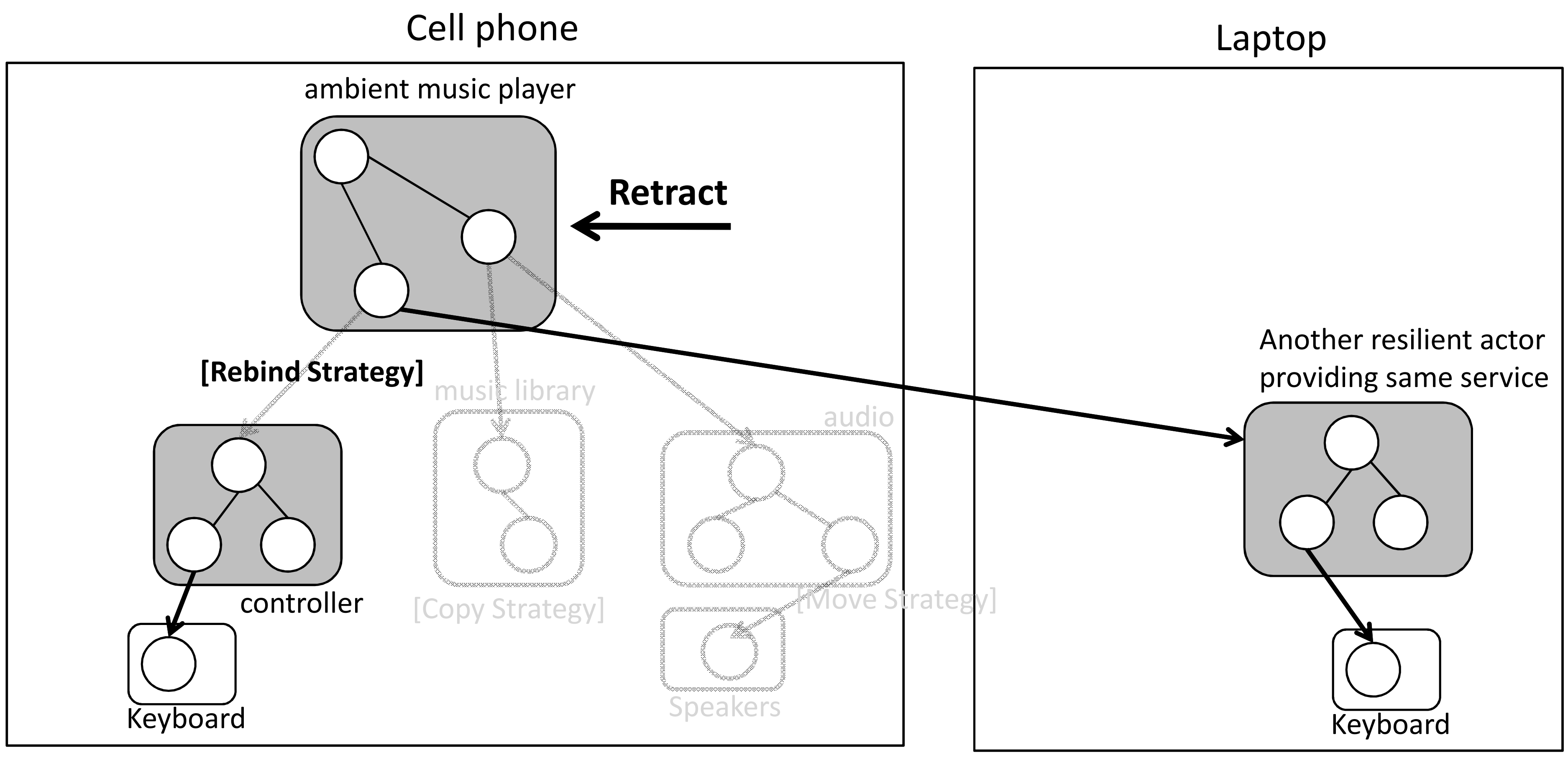
After Retract Operation with Copy Strategy



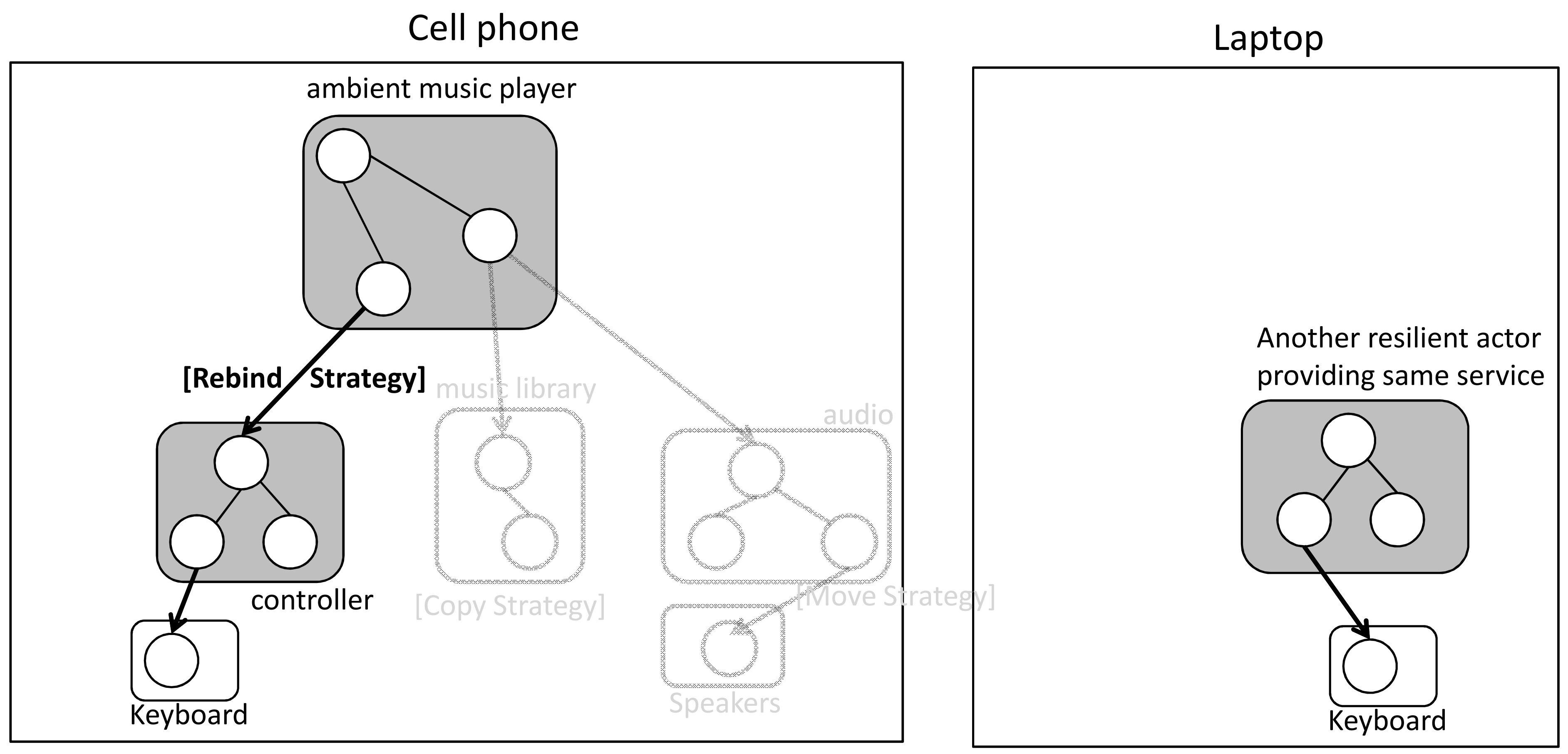
Stretch Operation with Rebind Strategy



Retract Operation with Rebind Strategy

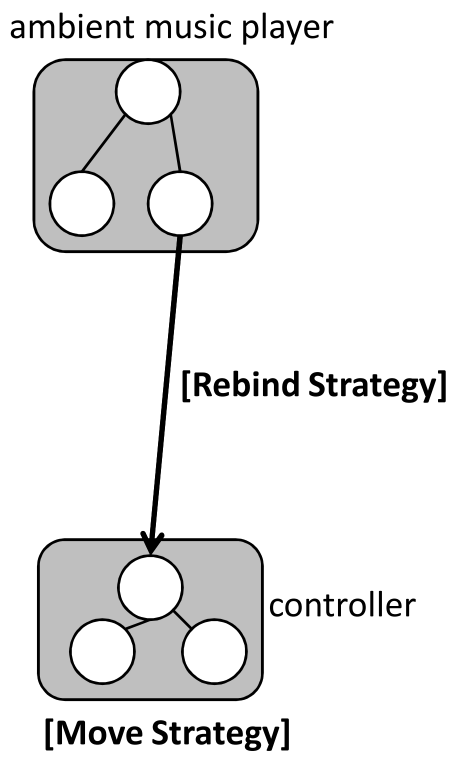


After Retract Operation with Rebind Strategy



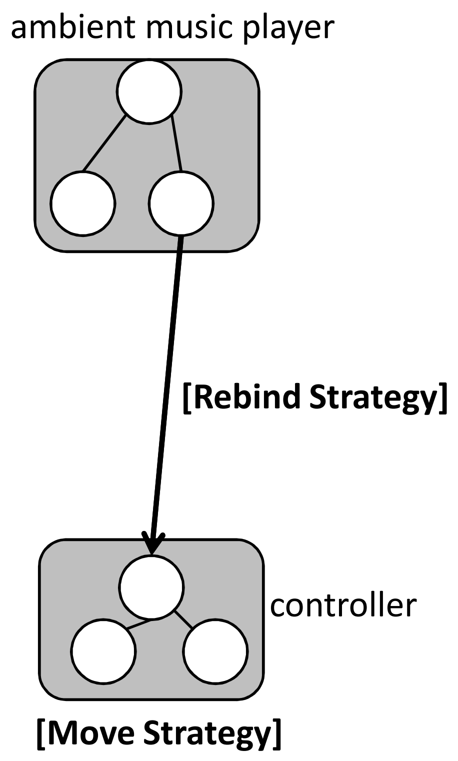
Resolution of Strategies

Resilient strategies can be defined on resilient actors and elastic bindings



Resolution of Strategies

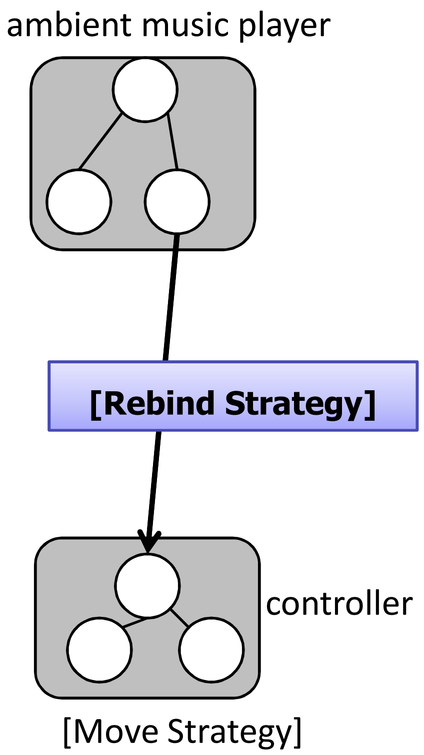
Resilient strategies can be defined on resilient actors and elastic bindings



Which resilient strategy will be applied ?

Resolution of Strategies

Resilient strategies can be defined on resilient actors and elastic bindings



Which resilient strategy will be applied ?

Resilient strategy on elastic binding	Resilient strategy on the resilient Actor
standstill	{move, copy, rebind, standstill}
rebind	{move, copy, rebind, standstill}
move	**{move, copy, rebind, standstill}**
copy	**{move, copy, rebind, standstill}**

Resilient Actors in AmbientTalk

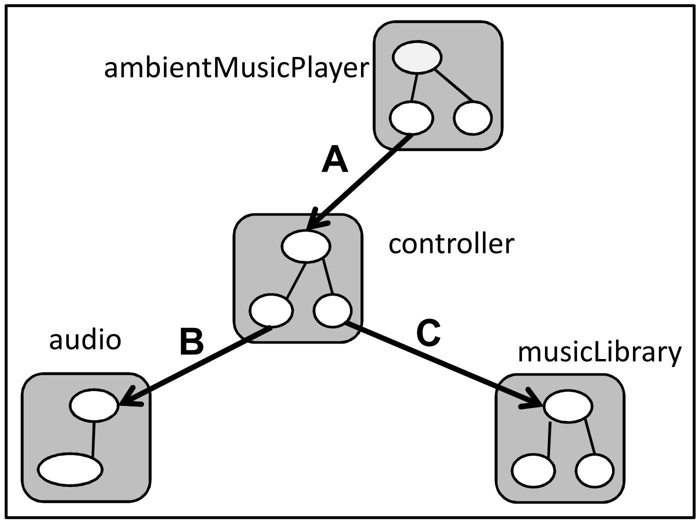
- AmbientTalk: Actor-based language for pervasive computing *(Van Cutsem et. al, 2007)*
- Mechanisms for handling network failures
- We extend AmbientTalk with four language constructs for resilient partitioning:

  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

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

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

A

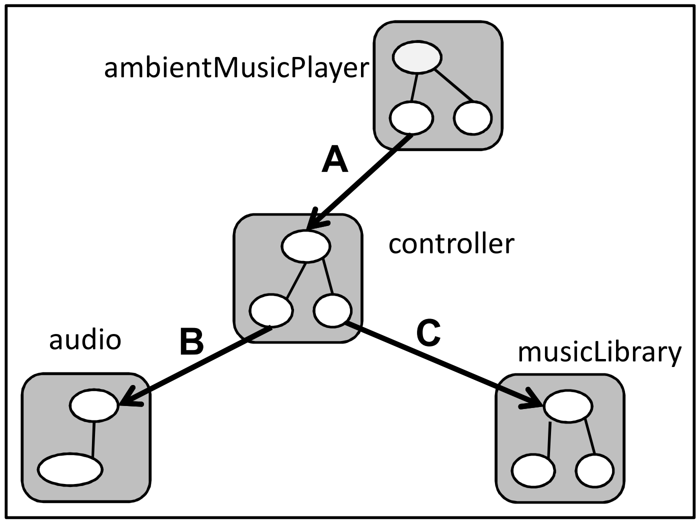
```
def  controller := actor: { |audio, musicLibrary|
      deftype musicLibraryService;
      def theAudio := bindTo: audio   resilientAs: [standstill];
      def theMusicLib := bindTo: musicLibrary  resilientAs: [rebind(musicLibraryService)];
} resilientAs: [move];
```

B

C

Resilient Actors in AmbientTalk (cont'd)

Implementation of the ambient music player using the resilient actor model



Definition of a resilient actor

def musicLibrary := **actor**: {….} **resilientAs**: [copy];
def audio := **actor**: {… } **resilientAs**: [standstill];

def ambientMusicPlayer := **actor**: { |controller|

def cellphone := **actor**: {….};

ambientMusicPlayer <- **stretch**: cellphone;

def controller := **actor**: { |audio, musicLibrary|

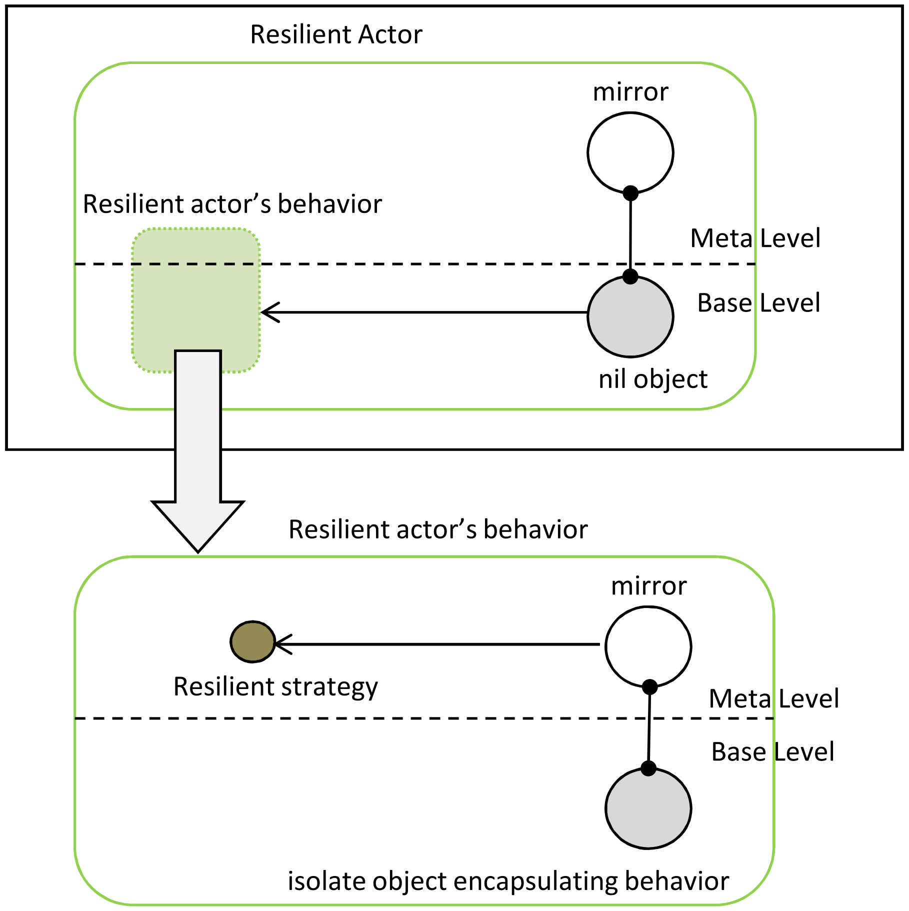
deftype musicLibraryService; B

def theAudio := **bindTo**: audio   **resilientAs**: [sta

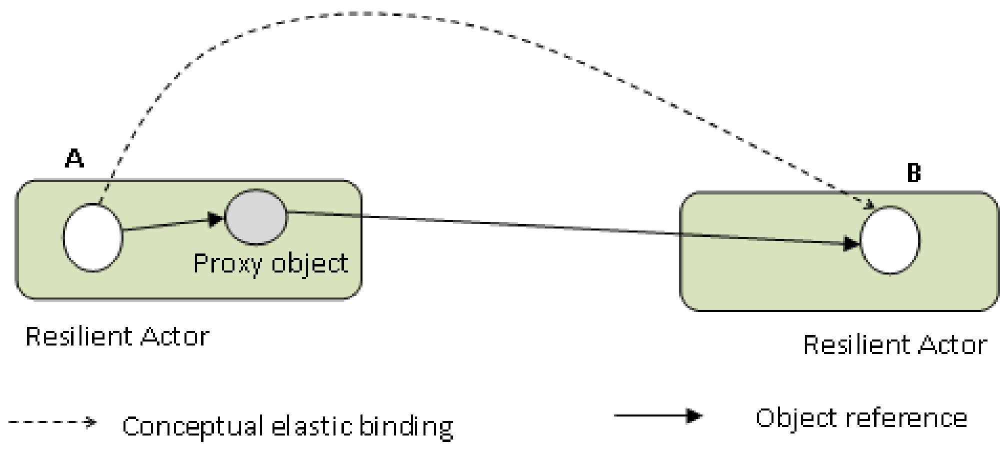
def theMusicLib := **bindTo**: musicLibrary **resilien**

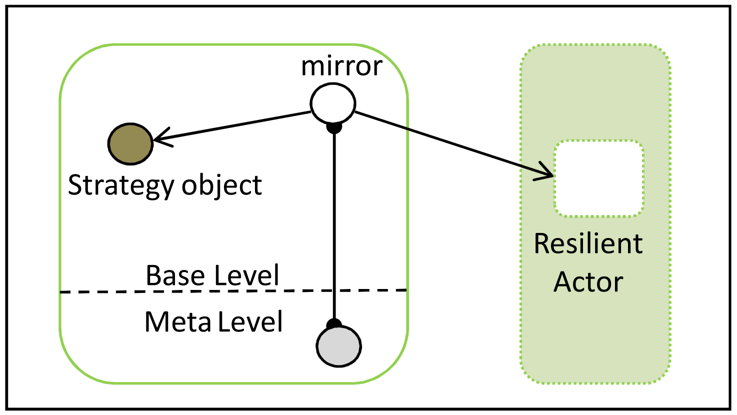
} **resilientAs**: [move];

Implementation of Resilient Actors



Implementation of Elastic Binding





Extensible Implementation

- New Resilient Strategies by Extension

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

```
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)