Context-Aware Service Composition and Execution in Pervasive Computing Environments

« Application to Content Adaptation »

PhD Thesis Presentation
by

Yaser Fawaz

Supervisors:  Prof. Lionel Brunie (Advisor)
Dr. Marian Scuturici (Co-advisor)

Laboratoire d'Informatique en Images et Systèmes d'information
Outline

- Introduction
- Related Work
- Proposed Approach
- Implementation and Experiments
- Conclusion
Introduction

Mobile Ad-hoc Network (MANET): can support pervasive computing (Any Time, Any Where, Any Device) effectively

Challenges in an infrastructure-less pervasive computing environment (MANET)
- Devices have limited capacities (CPU power, storage size, Screen dimensions)
- Devices are heterogeneous in terms of software capability. e.g. data formats supported
- Users have different preferences and contexts

Content adaptation is an efficient solution to overcome these challenges

Objective
- Allow devices in a dynamic infrastructure-less pervasive computing environment such as a MANET to perform content adaptation
Motivation: MANET at Restaurant-Office-Workplace…

Content adaptation process is needed in an infrastructure-less pervasive computing environment!
Related Work

- The existing adaptation approaches:
  - Client-based adaptation approach
  - Server-based adaptation approach
  - Proxy-based adaptation approach
  - Service-based adaptation approach

- Service composition: is a technique of creating a composite service with the help of smaller, simpler and easily executable services

- Service composition is appropriate to perform content adaptation in a dynamic pervasive computing environment such as a MANET
## Existing Service Composition Approaches

<table>
<thead>
<tr>
<th></th>
<th>eFlow</th>
<th>DCAF</th>
<th>HTGA</th>
<th>BDSCP</th>
<th>FTSCP</th>
<th>MoSCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure-less</td>
<td>-</td>
<td>-</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Context-Awareness</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Optimization of execution time</td>
<td>-</td>
<td>++</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reliability</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Decentralized Services Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>+</td>
<td>-</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

(++) Comprehensive  + Partial  - Limited or none
Our Objective

Propose a service composition approach which

- Supports the infrastructure-less environment
- Identifies the components of the composite service at run time
- Considers the dynamicity of services
- Optimises the composite service in terms of total completion time
- Implements a decentralized (Mesh-based) service execution pattern
- Is fault tolerant
Proposed Approach

Deployment of ConAMi on a MANET

Co-located devices create a MANET

Devices agree to share information and services

Devices advertise about its capacities and its services to all devices in the environment

Deployment of ConAMi on a MANET
ConAMi: Context-Aware Content Adaptation Middleware

Architecture of the ConAMi middleware
ConAMi: Context-Aware Content Adaptation Middleware

The central part of ConAMi

Selects and composes the appropriate adaptation services

Adaptation Plan Generator

Decides the number and the type of adaptation transformations required to transform the content into the appropriate form.

Contains local data, which may be text, image, audio, video

Local Data

Stores the description of all services available in the environment

Service Registry

Converts data from one form into another. Examples of such services are: text translator, text to audio converter, image resizer etc

Adaptation Ontology

Stores context data including user’s preferences, devices’ capabilities and network conditions

Context Repository

Stores rules which are defined by the user

Adaptation Rules

Detects faults which may occur due to unreachability of devices or other kinds of faults such as service discovery failure, service execution failure, etc

Action Monitor

Executes local services to perform content adaptation and forwards adapted data for subsequent adaptation.

Action Monitor

Stores ontology and metadata of concepts used in the content adaptation process

Another devices

Another devices

Another devices

Another devices
Important Terms

Content Adaptation Process: is a process of transforming the data from one format into another.
- Atomic / Simple adaptation process: is carried out in one adaptation step. e.g.
- Complex adaptation process: is carried out in a number of adaptation steps, and every adaptation step is called an adaptation task or adaptation transformation. e.g.

Adaptation Service: is a software component that performs an adaptation task. Examples of these services are: text to audio transformer, color reducer, image resizer, etc. It is described as:

\[ S = (IdService, IdTask, IdDevice, PR, TTL, In(t), Out(t)) \]

Services A and Z: are predefined services which do nothing. They represent the initial and the goal states of the composite service

Service composition plan: is a set of services that form the composite service

Time To Leave (TTL): is the time after which the service is no more available

Contractor Peer: a device which coordinates and supervises the content adaptation process

Adaptation Problem: is a four-tuple \( (f, S_A, S_Z, T) \) where:
- \( f \): the file for which the adaptation process is done
- \( S_A \): is the initial state of the file \( f \), i.e., the state of \( f \) before performing the adaptation process
- \( S_Z \): is the goal state of the file \( f \), i.e., the state of \( f \) after performing the adaptation process
- \( T \): is a list of adaptation tasks
Functional Model of our Approach

Collaborative Adaptation System

- Content Metadata
- Content Data
- Context Data
- Adaptation Service Description

Delivers

Adapted Data
Content Data and Metadata Description

The content data: Text, Audio, Image, Video

The content metadata: a set of attributes such as Identifier, Type, Format, Size, Language,…etc

The MPEG-7 standard is used to describe the content metadata.

Sample content metadata description for an image:

```
< ?xml version="1.0" encoding="UTF-8" >
<ContentDescription>
  <Content type="Image">
    <ContentInformation ID ="Image_ID">
      <ContentTitle>Image_SIP</ContentTitle>
      <ContentDescription>The photo was taken on the birthday of Lionel</ContentDescription>
    </ContentInformation>
    <ContentProfile>
      <ContentFormat>JPEG</ContentFormat>
      <ContentSize>5 MB</ContentSize>
      <ContentWidth>600 Pixels</ContentWidth>
      <ContentHeight>800 Pixels</ContentHeight>
      <ContentColor>32</ContentColor>
    </ContentProfile>
  </Content>
</ContentDescription>
```
Adaptation Service Description

- Service description enables service discovery, service composition and service selection.

- WSDL (Web Service Description Language) is not adequate for adaptation services since it does not include semantic information.

- Adaptation service description should include semantic information such as input/output data format of the service, effect of executing the service, execution time, TTL, ...etc.

- Semantic service description can be achieved using ontology.

- We adapt the service description ontology developed in our laboratory [Berhe et al.]
Adaptation Service Description

Ontology

An excerpt of the graphical representation
Context Data Representation / Cona model based on EHRAAM [Ejigu et al 2007]

Entity

Hierarchical relation

Relation (entity & attribute type relationships)

Axiomatic relation

Metadata relation

Activity

Resource

User

Device

Data

Network

Location

Language

Adaptation Process

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

Video

Image

Text

Audio

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

applyTo

engagedIn

hasPreference

hasUserLang

hasOwner

(hasOwner (by inverse axiom))

connectedTo

(locatedIn (By inverse axiom))

hasScreenWidth

Width

hasScreenHeight

Height

hasScreenColor

Color

hasSize

Size

hasFormat

Format

hasType

Type

hasBandwidth

Bandwidth

hasMemorySize

Memory

hasContentLang

Language

isA

hasPrecision

User

Device

Network

PP

AudioToText

Video

Image

Text

Audio

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText

PP

VideoResizing

ImageResizing

AudioSummarizing

TextSummarizing

AudioSummarizing

AudioToText
The functional model of the CAS module:

- **HCoM (Hybrid Context Management)**: Uses an ontology approach to manage context semantics and a relational approach to manage context data.
- **RAID (Reasoning, Aggregation, Interpretation and Decision)**: Provides the core context-aware service. It populates the ontology with context data and then applies rules and axioms for reasoning and decision on the actions to be triggered.

The CAS module consists of two components [Ejigue et al]:
- The HCoM (Hybrid Context Management) model: It uses an ontology approach to manage context semantics and a relational approach to manage context data.
- The RAID (Reasoning, Aggregation, Interpretation, and Decision) engine: It provides the core context-aware service. It populates the ontology with context data and then applies rules and axioms for reasoning and decision on the actions to be triggered.
An excerpt of the context ontology for content adaptation:

```xml
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.cona.fr/ContentOntology.owl#"
    xml:base="http://www.cona.fr/ContentOntology.owl">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="AudioSummarizer">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="Service"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Outdoor">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="Location"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="SIPgroup">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="User"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Device">
        <rdfs:subClassOf>
            .....
        </rdfs:subClassOf>
    </owl:Class>
</rdf:RDF>
```
Sample Rules for Content Adaptation

We have defined transformation rules that are easily extensible to add new transformation functionalities and user, device, network constraints.

@prefix cona: <http://www.cona.fr/ContentOntology.owl#>

#Text Rules

[Device Limitation:]
(?X cona:isPresentedTo ?U)
(?X rdf:type cona:Text_Full)
(?U cona:isUsing ?D)
(?D cona:hasMemory cona:Limited)
-> (?X cona:hasAssociatedTask cona:Text_Summarization)

[User Preference:]
(?X cona:isPresentedTo ?U)
(?X rdf:type cona:Text_Full)
(?U cona:hasUserLanguage ?UL)
(?X cona:hasContentLanguage ?CL)
notEqual(?UL, ?CL)
-> (?X cona:hasAssociatedTask cona:TextTranslation)

[User’s Context:]
(?X cona:isPresentedTo ?U)
(?X rdf:type cona:Text_Full)
(?U cona:hasUser’s context engaged in activity)
-> (?X cona:hasAssociatedTask cona:TextToAudioConversion)
Performance of CAS Module

Response time of context-aware service versus number of RDF triples
The CAS module produces a list of adaptation tasks called Transformation Planning:

- Text Summarization
- Text Translation
- Text to Audio Transformation

Every adaptation task can be carried out using several adaptation services.

The challenges are how our system selects the appropriate services in the environment and how it initiates collaboration among peers to carry out the adaptation services.
Adaptation Plan Generator (APG) Module Functionality

Adaptation Tree Algorithm: it returns sets of nodes which represent the service composition plan

**Input:** t_i, St_i (for i = 1 to n+1 where n is the number of adaptation tasks)

**Output:** an adaptation tree (AT)

**Global variables**
- `greenLeaves`: a variable to store new green nodes
- `newParents`: a variable to store new parents nodes
- `newLeaves`: a variable to store new nodes
- `St_i`: a set of services correspondent to an adaptation task t_i
- `t_i`: an adaptation task
- `SNC_ij` is a subset of St_i and contains services which are incompatible to the service S_j represented by the node n_j
- `NS`: a set of neutralization services
- `S`: a set of adaptation services and neutralization services

**Begin**

1. Create a node n_0 for service A
   - `n_0.serv = A`
   - `n_0.finTime = Initialization Time` // set finishing Time of n_0
   - `InitializationTime`
   - `n_0.color = GREEN` // set color of n_0 green
   - `n_0.parent = null` // set parent of n_0 null
   - `n_0.numChild = 0` // set the number of children of n_0 zero
   - `newParents = { n_0 }` // set n_0 as a new parent

**End**
Example...

\[ \text{St}_0 = \{A\} \]

The list of adaptation tasks which are produced by the CAS module for the example scenario are:

\[ \text{St}_1 = \{S_1, S_2, S_3\} \]

\[ \text{St}_2 = \{S_4, S_5, S_6, S_7\} \]

\[ \text{St}_3 = \{S_8, S_9, S_{10}, S_{11}\} \]

\[ \text{St}_4 = \{Z\} \]
The APG Module...

1. The APG module selects the optimal service composition plan, in terms of execution time, during tree construction.
2. The APG creates a record message (RM) which contains:
   - The addresses of the services in the optimal path of the tree
   - The address of the requester peer
   - The address of the contractor peer
   - The list of the adaptation tasks
   - The data to be adapted

The RM message for the example scenario is:

<table>
<thead>
<tr>
<th>$S_{2}, S_{5}, S_{9}, S_{83}$</th>
<th>@ R.P</th>
<th>@ C.P</th>
<th>ListTasks</th>
<th>Data to be adapted</th>
</tr>
</thead>
</table>
Performance of tree construction algorithm

Tree Construction time versus number of tasks and number of services per task
Decentralized Composite Service Execution Protocol

The advantage of the SEDF module is to make the partially adapted data flow directly from one service to the next one without the need to an intermediator.

The decentralized service execution protocol over perform than the centralized one with respect to delivery time and the size of exchanged data.

The contractor peer sends the record message RM1 to the device which has the first service in RM1.

The SEDF Module functionality:
Exececutes the local service, modifies the record message and it forwards the modified record message to the next device.
Comparison of Composite Service Execution Protocols

Data delivery time versus number of services (Bandwidth= 54Mb/s, file size= 1 Mbytes)

Data delivery time versus file size (Bandwidth=5.5Mb/s, three services are considered)

Exchanged data size versus number of services (Bandwidth= 54Mb/s, file size= 1 Mbytes)
Content adaptation is accomplished successfully if there is no fault during the execution of the composite service.

Faults can be happen very easily in a MANET due to:
- Network disruption
- Service discovery failure
- Service execution failure
- …etc

To enable a reliable composite service, the middleware ConAMi considers the time to leave of service (TTL) during services composition.

TTL can’t give a guarantee for non occurrence of a fault

Fault detection and recovery mechanism is needed to enhance the performance of the ConAMi middleware
Fault Detection and Recovery Mechanism

To ensure highly reliable service composition, the devices exchange acknowledgement messages. Every device caches a copy of the record message until it receives an ACK message from its successor.

The action monitor (AM) module of the contractor peer uses acknowledgement messages to detect faults. Every device executes the (AM) module to detect faults. The requesting (R.P) peer requests data to the contractual (C.P) peer. The contractual (C.P) peer delivers the requested data to the owner of the task (S).

To ensure highly reliable service composition, the devices exchange acknowledgement messages. Every device caches a copy of the record message until it receives an ACK message from its successor.

The action monitor (AM) module of the contractor peer uses acknowledgement messages to detect faults. Every device executes the (AM) module to detect faults. The requesting (R.P) peer requests data to the contractual (C.P) peer. The contractual (C.P) peer delivers the requested data to the owner of the task (S).

To ensure highly reliable service composition, the devices exchange acknowledgement messages. Every device caches a copy of the record message until it receives an ACK message from its successor.

The action monitor (AM) module of the contractor peer uses acknowledgement messages to detect faults. Every device executes the (AM) module to detect faults. The requesting (R.P) peer requests data to the contractual (C.P) peer. The contractual (C.P) peer delivers the requested data to the owner of the task (S).

To ensure highly reliable service composition, the devices exchange acknowledgement messages. Every device caches a copy of the record message until it receives an ACK message from its successor.

The action monitor (AM) module of the contractor peer uses acknowledgement messages to detect faults. Every device executes the (AM) module to detect faults. The requesting (R.P) peer requests data to the contractual (C.P) peer. The contractual (C.P) peer delivers the requested data to the owner of the task (S).

To ensure highly reliable service composition, the devices exchange acknowledgement messages. Every device caches a copy of the record message until it receives an ACK message from its successor.

The action monitor (AM) module of the contractor peer uses acknowledgement messages to detect faults. Every device executes the (AM) module to detect faults. The requesting (R.P) peer requests data to the contractual (C.P) peer. The contractual (C.P) peer delivers the requested data to the owner of the task (S).
Fault detection and recovery mechanism...

AM Module

- Time estimation for receiving Ack message from the service $S_9$:
  \[ \text{EstTime} = \text{finshingTime}(S_9) + \text{Threshold} \]
- If Ack is not received within EstTime, The AM module assumes that the service $S_9$ is inaccessible

APG Module

- Inaccessible services are replaced in the record message by using the fault recovery algorithm
Example...

\[
\text{St}_0 = \{A\}
\]

\[
\text{St}_1 = \{S_1, S_2, S_3\}
\]

\[
\text{St}_2 = \{S_4, S_5, S_6\}
\]

\[
\text{St}_3 = \{S_8, S_9, S_{10}\}
\]

\[
\text{St}_4 = \{Z\}
\]

**Fault Recovery Algorithm**

**Input:**
- \(T\): an adaptation tree
- \(\text{OP}\): nodes found in the optimal path of \(T\)
- \(\text{N}\): nodes represent services of the record message which is prepared by the contractor peer
- \(i\): the index of the node that represents the broken service in \(\text{N}\)

**Output:**
- \(\text{newS}\): services in the new record message

**Begin**
- initialize \(\text{newS}\) to empty
- \(\text{newS} = \text{replaceBrokenService}(\text{N}[i-1], \text{N}[i+1].\text{parent}, \emptyset)\), where:
  - \(\text{N}[i+1].\text{parent}\): the parent node of the node \(\text{N}[i+1]\) in \(T\)
  - assign \(S_{\text{ref}}\) as the last service in \(\text{newS}\)
  - assign \(i\) as the index of the node that represents an equivalent service to \(S_{\text{ref}}\) in \(\text{OP}\)
  - assign next as the index of the non-neutralization node after the \(i^{th}\) node in \(\text{OP}\)
  - if \(\text{OP}[\text{next}]\) is not null:
    - \(\text{newS} = \text{replaceBrokenService}(\text{OP}[i], \text{OP}[\text{next}], \text{newS})\)
    - go to step 3

**End**

The APG module of the contractor peer implements a fault recovery algorithm to select another path in the tree.

The services \(S_9\) and \(S_{83}\) are inaccessible because the devices \(P_4\) and \(P_6\) are disconnected.

The services \(S_9, S_{83}\) are replaced by the services \(S_{10}, S_{84}\).
Fault Detection and Recovery Mechanism...

Replace $S_9, S_{83}$ by $S_{10}, S_{84}$

$S_5$ is executed

$S_{10}$ is executed

$S_9$ is not executed

$S_{84}$ is executed

We suppose the devices $P_4$ and $P_6$ are disconnected

The services $S_2$ and $S_5$ are exploited

Data are delivered

Contractor peer

Requester peer

Owner of $S_2$

Owner of $S_{83}$

Owner of $S_{84}$

Owner of $S_{10}$

Owner of $S_9$

RM33

RM3

RM44

ACK

ACK

NoACK

NoACK

ACK

ACK

ACK

ACK

ACK

ACK

ACK

ACK
Performance of the Fault Recovery Algorithm

Fault recovery time versus number of services per task
## Our Approach VS Existing Service Composition Approaches

<table>
<thead>
<tr>
<th></th>
<th>eFlow</th>
<th>DCAF</th>
<th>HTGA</th>
<th>BDSCP</th>
<th>FTSCP</th>
<th>MoSCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure-less</td>
<td>-</td>
<td>-</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Context-Awareness</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Optimization of execution time</td>
<td>-</td>
<td>++</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reliability</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Decentralized Services Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>+</td>
<td>-</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

(++) Comprehensive     (+) Partial     (-) Limited or none)
Conclusions

Introduced a Context-Aware Content Adaptation Middleware (ConAMi) that

- Supports the infrastructure-less pervasive computing environment
- Recognizes the contextual information (device capabilities, user’s preferences and network condition)
- Identifies the components of the composite service at run time
- Considers the dynamicity of the services in the vicinity (TTL)
- Implements a decentralized (Mesh-based) service execution pattern
- Optimizes the composite service in terms of total completion time
- Discover faults and recover from them

Prototype is developed to evaluate the performance of the ConAMi middleware
Thanks for your attention!
Architectur of the Context-Aware Service (CAS)

Layer 5: Application Layer (Adaptation Plan Generator)

Layer 4: Reasoning and Decision Layer

Layer 3: Management Modeling Layer (HCoM Model)

Layer 2: Context Pre-processing Layer (EHRAM Model)

Layer 1: Context Capturing Layer

- Hardware and software capturing tools

Layer 1: Context Capturing Layer

- Context Capturing Layer
  - Hardware and software capturing tools

Layer 2: Context Pre-processing Layer (EHRAM Model)

- Entities
- Hierarchies
- Relations
- Axioms
- Metadata

Layer 3: Management Modeling Layer (HCoM Model)

- Context Manager
- Learned Rules
- Dispatch Rule
- Generic Ontology
- Context-Onto
- Dispatcher Context
- Select Relevant
- RAID-Action Engine
- Action Parameters
- Domain Ontology
- Repository
- Dynamic Context
- Context Filter
- Static Context

Layer 4: Reasoning and Decision Layer

- Reasoning
- Aggregation
- Interpretation
- Decision

Layer 5: Application Layer (Adaptation Plan Generator)
### Adapation Services

<table>
<thead>
<tr>
<th>TaskID</th>
<th>Service ID</th>
<th>Time to leave (s)</th>
<th>DeviceID</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>S₁</td>
<td>1200</td>
<td>P₆</td>
<td>Text Summarization</td>
</tr>
<tr>
<td></td>
<td>S₂</td>
<td>1000</td>
<td>P₂</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S₃</td>
<td>600</td>
<td>P₇</td>
<td></td>
</tr>
<tr>
<td>t₂</td>
<td>S₄</td>
<td>900</td>
<td>P₈</td>
<td>Text Translation</td>
</tr>
<tr>
<td></td>
<td>S₅</td>
<td>700</td>
<td>P₃</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S₆</td>
<td>800</td>
<td>P₅</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S₇</td>
<td>5</td>
<td>P₉</td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td>S₈</td>
<td>300</td>
<td>P₁</td>
<td>Text to Audio Transformation</td>
</tr>
<tr>
<td></td>
<td>S₉</td>
<td>600</td>
<td>P₄</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S₁₀</td>
<td>1500</td>
<td>P₇</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S₁₁</td>
<td>5</td>
<td>P₉</td>
<td></td>
</tr>
</tbody>
</table>

### Neutralisation Services

<table>
<thead>
<tr>
<th>Service ID</th>
<th>Time to leave (ms)</th>
<th>Device ID</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₈₀</td>
<td>500</td>
<td>P₇</td>
<td>Text Format Conversion</td>
</tr>
<tr>
<td>S₈₁</td>
<td>600</td>
<td>P₆</td>
<td></td>
</tr>
<tr>
<td>S₈₂</td>
<td>700</td>
<td>P₁</td>
<td>Audio Format Conversion</td>
</tr>
<tr>
<td>S₈₃</td>
<td>600</td>
<td>P₆</td>
<td></td>
</tr>
<tr>
<td>S₈₄</td>
<td>1100</td>
<td>P₈</td>
<td></td>
</tr>
</tbody>
</table>

Mapping required tasks to services:

- \( t₀ \rightarrow Sₜ₀ = \{A\} \)
- \( t₁ \rightarrow Sₜ₁ = \{S₁, S₂, S₃\} \)
- \( t₂ \rightarrow Sₜ₂ = \{S₄, S₅, S₆, S₇\} \)
- \( t₃ \rightarrow Sₜ₃ = \{S₈, S₉, S₁₀, S₁₁, S₁₂\} \)
- \( t₄ \rightarrow Sₜ₄ = \{Z\} \)