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

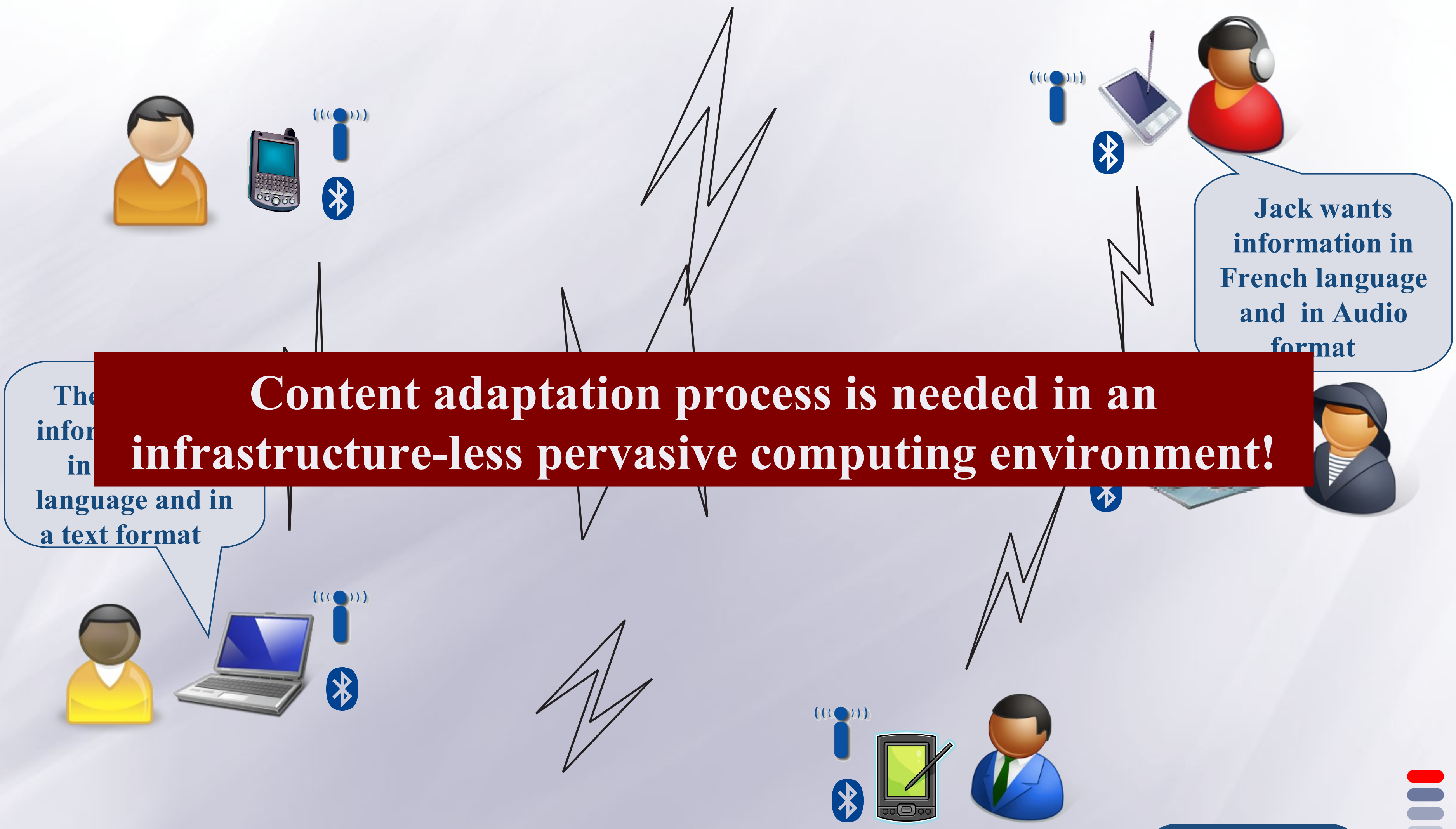
Outlin
e

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

Introductio
n

- **Mobile Ad-hoc Network (MANET):** can support pervasive computing (*Any Time, Any Where, Any Device*) effectively
- **Challenges an in 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 excuteable services
- Service composition is appropriate to perform content adaptation in a dynamic pervasive computing environment such as a MANET

Existing Service Composition Approaches

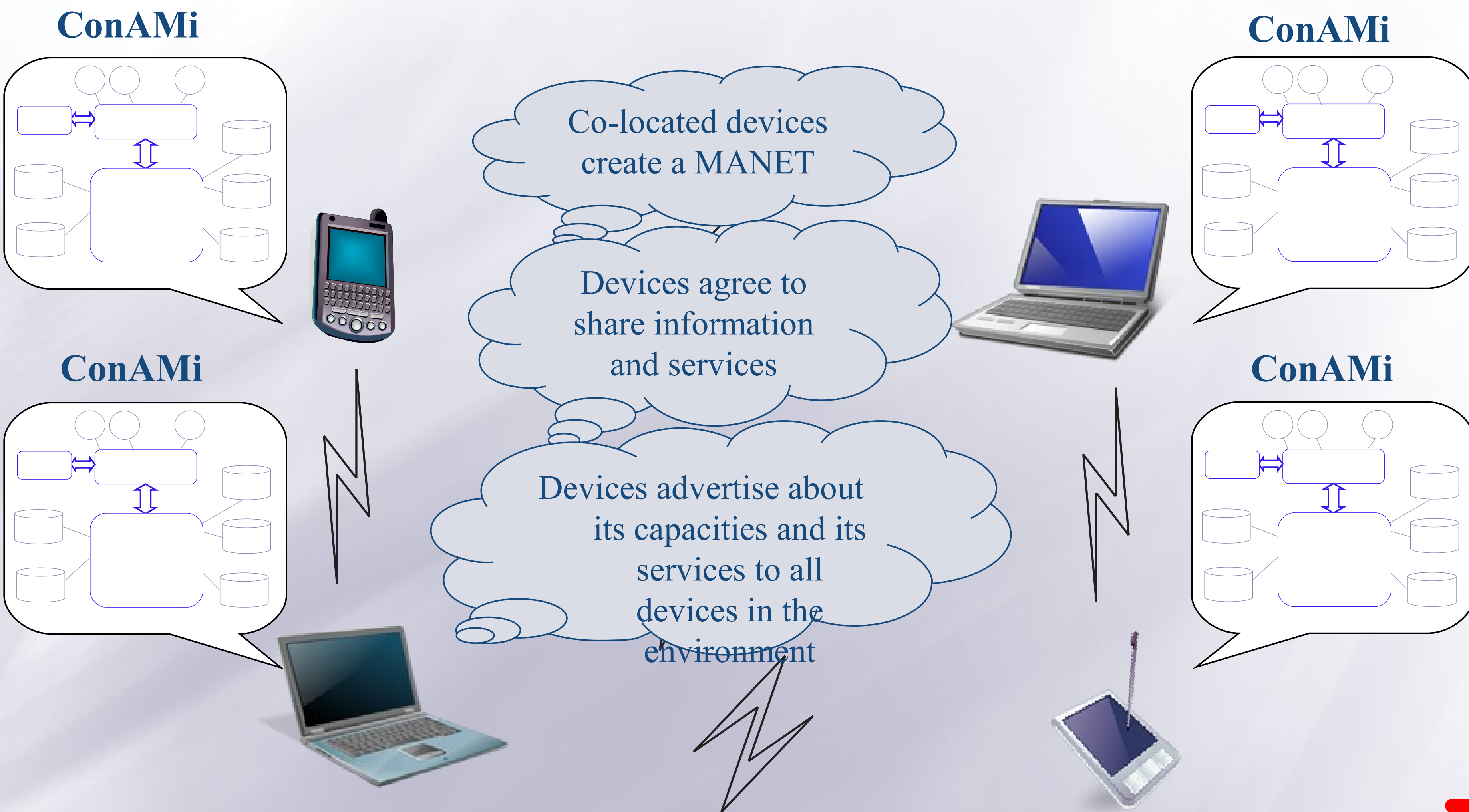
	eFlow	DCAF	HTGA	BDSCP	FTSCP	MoSCA	OUR APPROACH
Infrastructure-less	-	-	++	++	++	++	
Context-Awareness	-	+	-	-	-	-	
Optimization of execution time	-	++	-	-	-	-	
Reliability	+	-	+	+	+	+	
Decentralized Services Execution	-	-	-	-	-	-	
Fault Tolerance	+	-	++	++	+	-	

(++ Comprehensive   + Partial   - Limited or none)

Our Objective ???

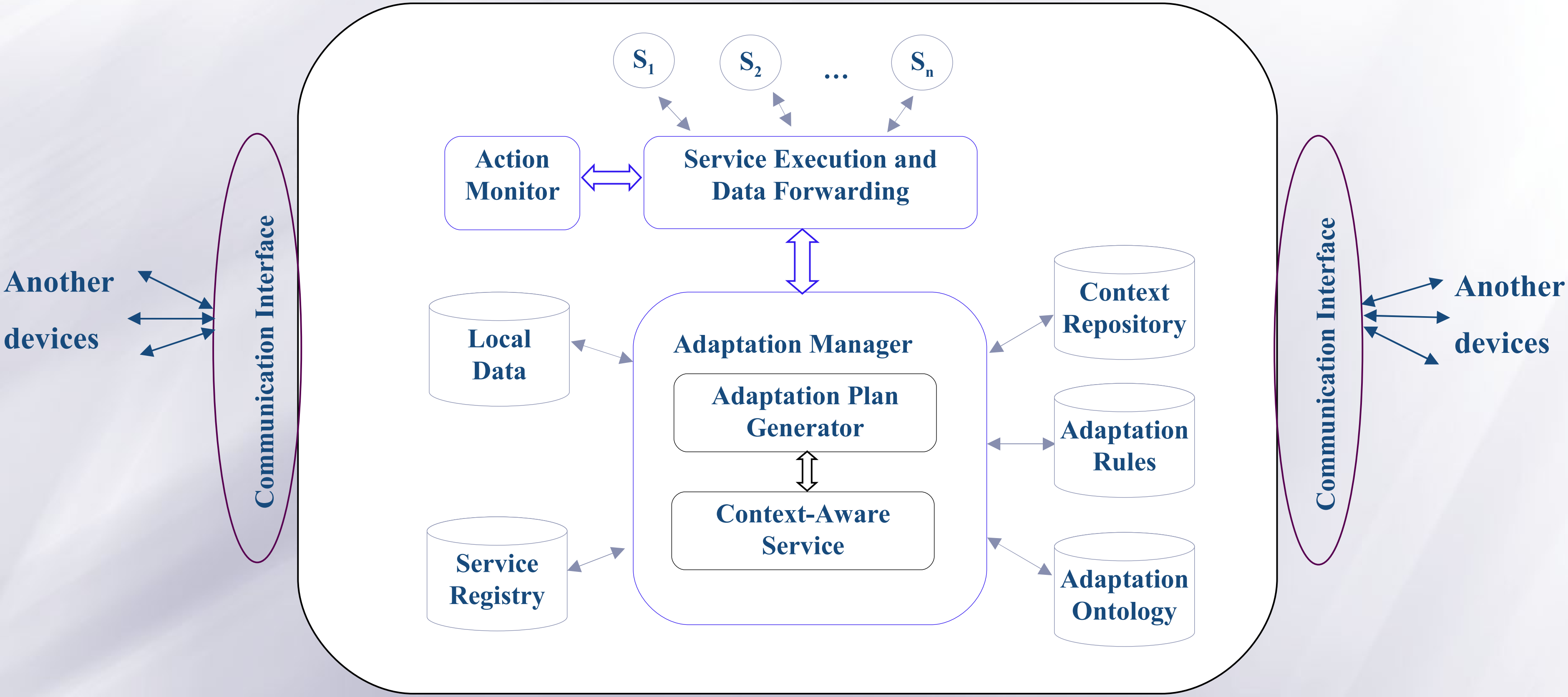
- Propose a service composition approach which
  - Supports the infrastructure-less environment
  - Identifies the components of the composite service at run time
  - Considers the dynamicity of services
  - Optimises the composite service in terms of total completion time
  - Implements a decentralized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

Proposed Approach



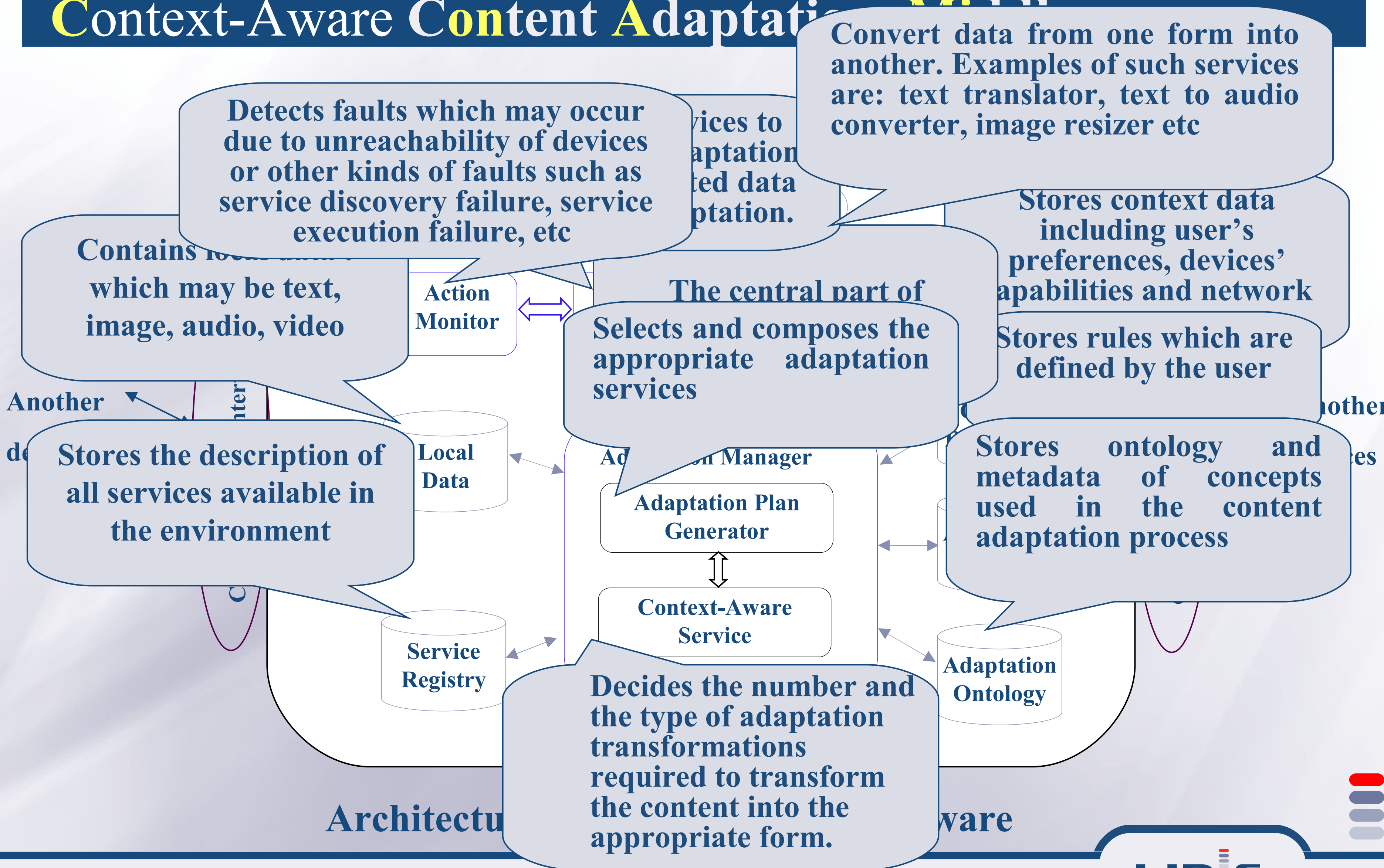
Deployment of ConAMi on a MANET

ConAMi:
Context-Aware Content Adaptation Middleware



Architecture of the ConAMi middleware

ConAMi:
Context-Aware Content Adaptati



Architectu ware

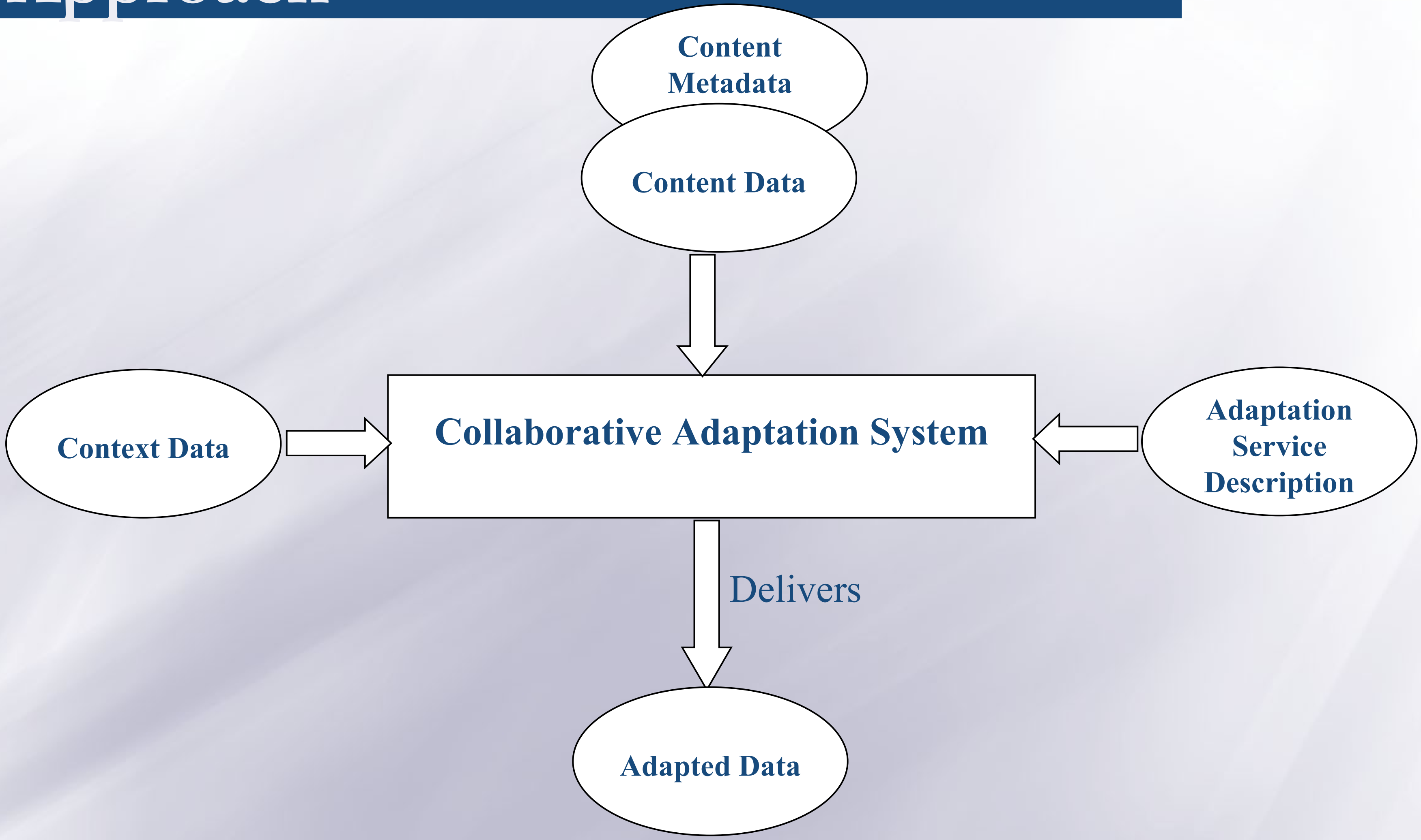
Important Terms

- **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 intial 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



Content Data and Metadata Description

- The content data : Text, Audio, Image, Video
- The cont … ch as Identifier, Type, Fo…
- The MP… be the content metadata…
- Sample c… image:

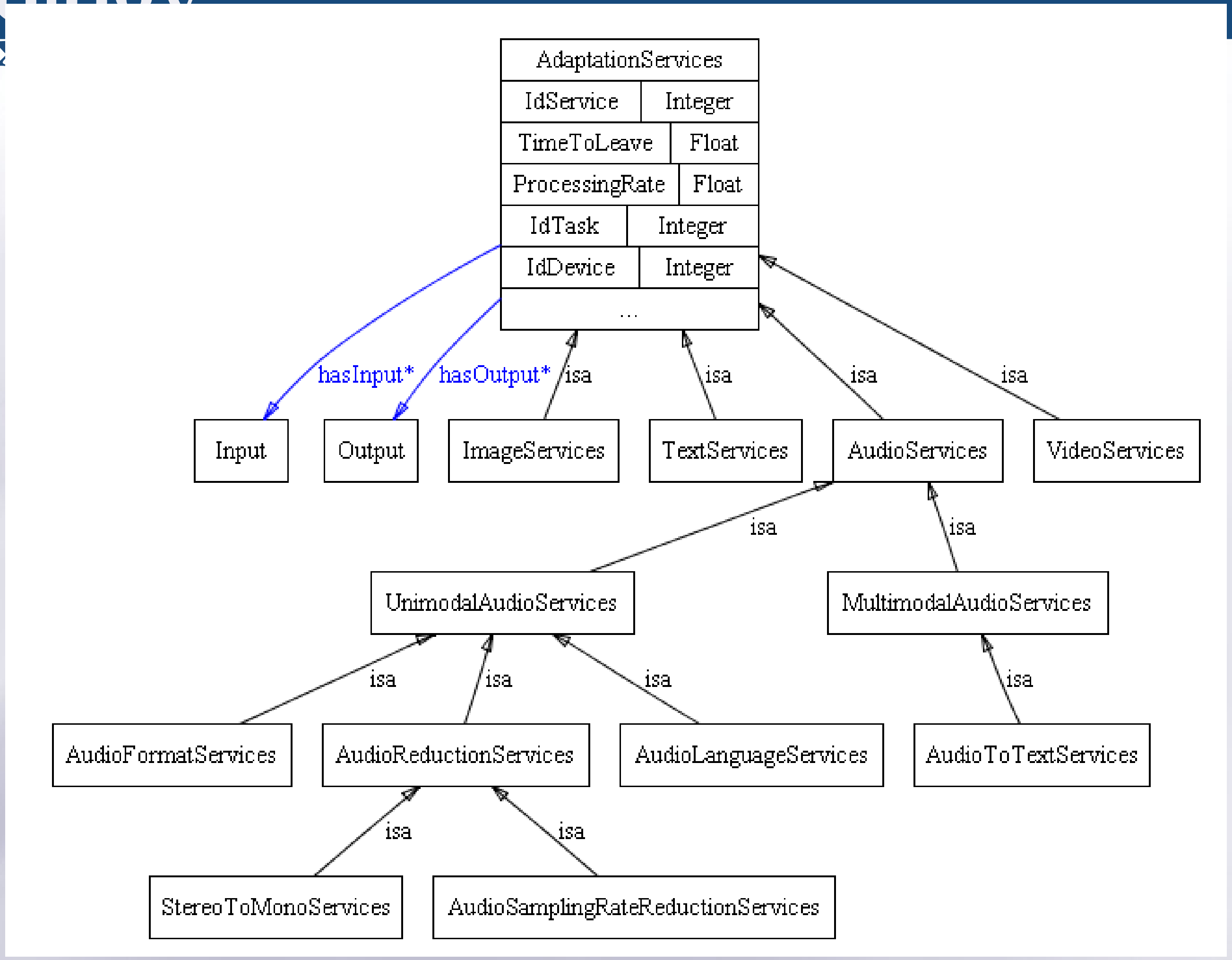
```
< ?xml version="1.0" encoding="UTF-8"?>
<ContentDescription>
   <Content type="Image">
         <ContentInformation ID ="Image_ID">
            <ContentTitle>Image_SIP</ContentTitle>
            <ContentDescription>The photo was taken
            on the birthday of Lionel
            </ContentDescription>
         </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

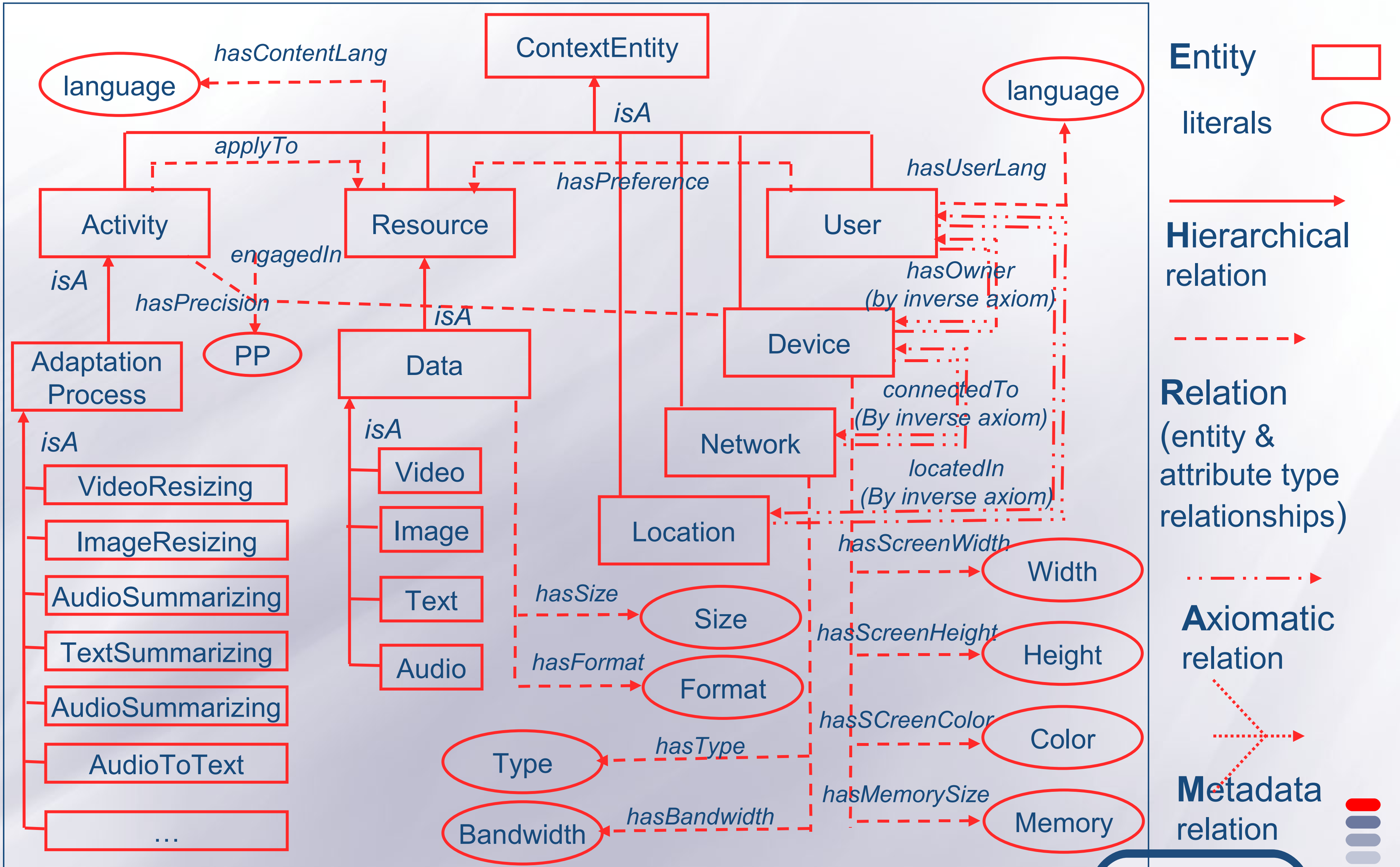
- Service description enables service discovery, service composition and service selection
- WSDL (Web Service Description Language) is not adequate for adaptation services since it does not include semantic information
- Adaptation service description should include semantic information such as input/output 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 ex

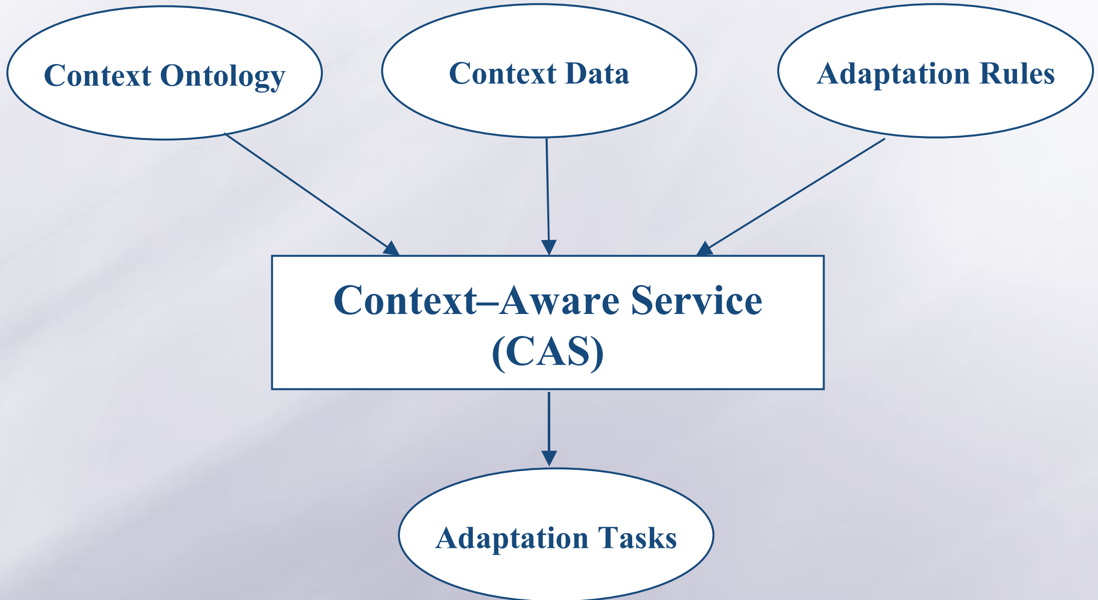


Context Data Representation / Cona model based on EHRAM [Ejigu et al 2007]



Context-Aware Service (CAS)

- The functional model of the CAS module:



- The CAS module consistes of two components [Ejigue et al]:
  - The HCoM (Hybrid Context Management) model : it uses ontology approach to manage context semantics and 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 the context data and then applies rules and axioms for reasoning and decision on the actions to be triggered

Context Ontology for Content Adaptation based on cona Model

An excerpt of the context ontology for content adaptation

```
<?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>
      … … …
```

Sample Rules for Content Adaptation

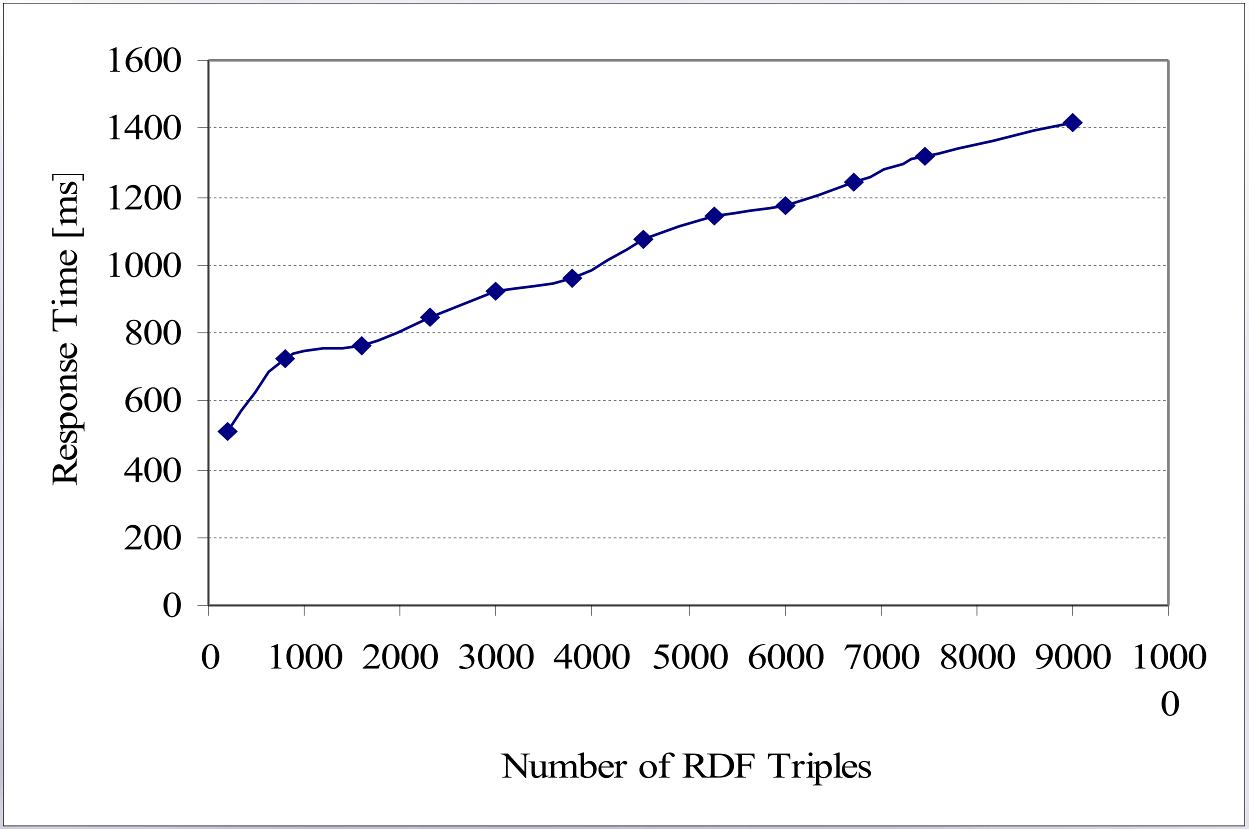
- We have defined transformation rules that are easily extensib… …lities and user, de…

```
@prefix cona: <http://www.cona.fr/ContentOntology.owl#>
#Text Rules
[DeviceLimitation:
      (?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)

[UserPreference:
      (?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 enaged in activity)
->    (?X cona:hasAssociatedTask cona:TextToAudioConversion)]
```

Performance of CAS Module



Response time of context-aware service versus number of RDF triples

Example

- Th… list of adaptation tasks ca… Planning

Device limitation

User’s preference

User’s context « Activity »

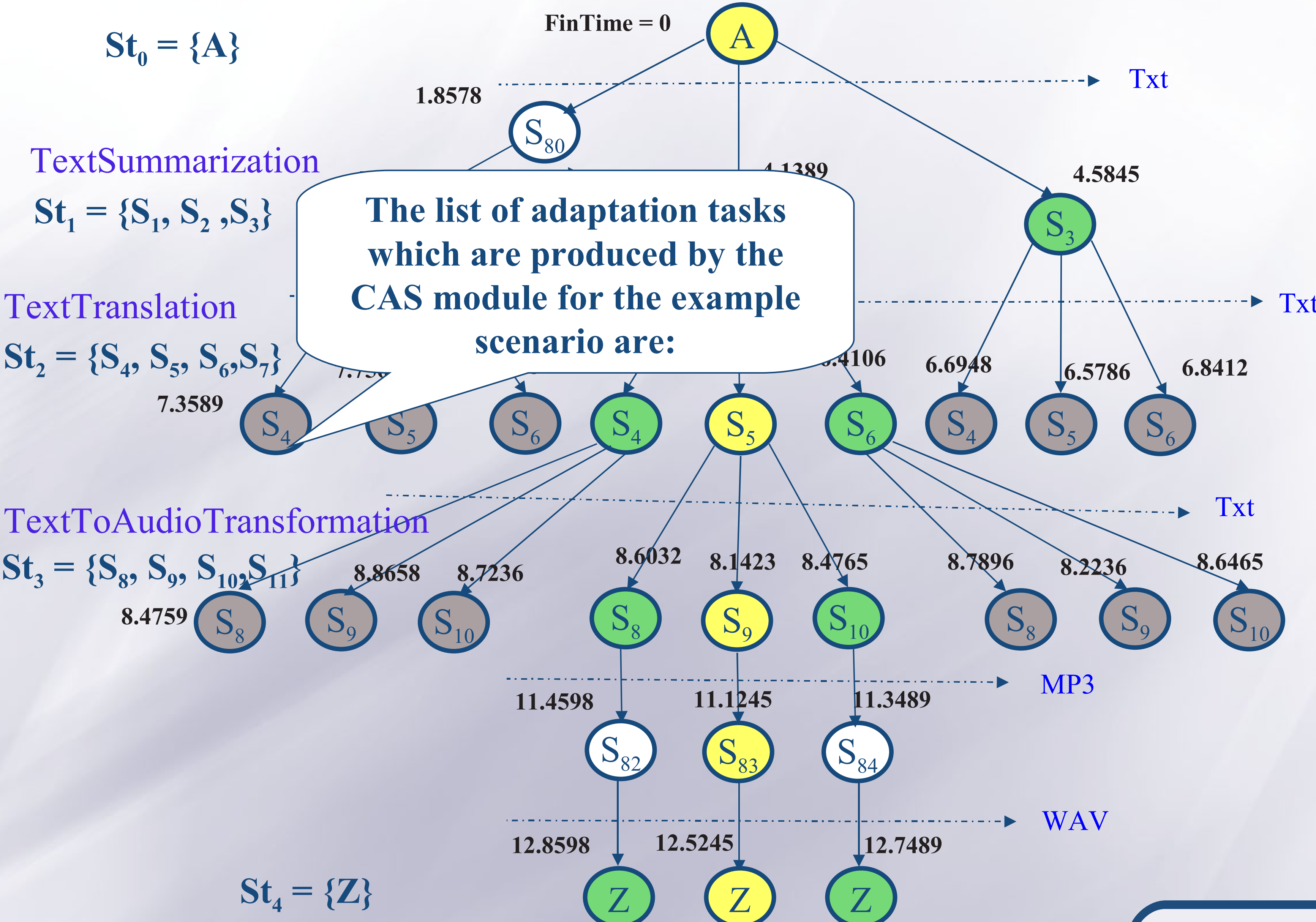
  - Text Summarization
  - Text Translation
  - Text to Audio Transformation
- Every adaptation task can be carried out using severals adaptation services.
- The challenges are how our system selects the appropriate services in the environment and how it initiates collaboration among peers to carry out 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
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

- **Service c… is composed of nodes an…**
  - **Every… …n time (TT) of a given…**
  - **Every… …here:**
    finT… …e,n_j.service)

Example…



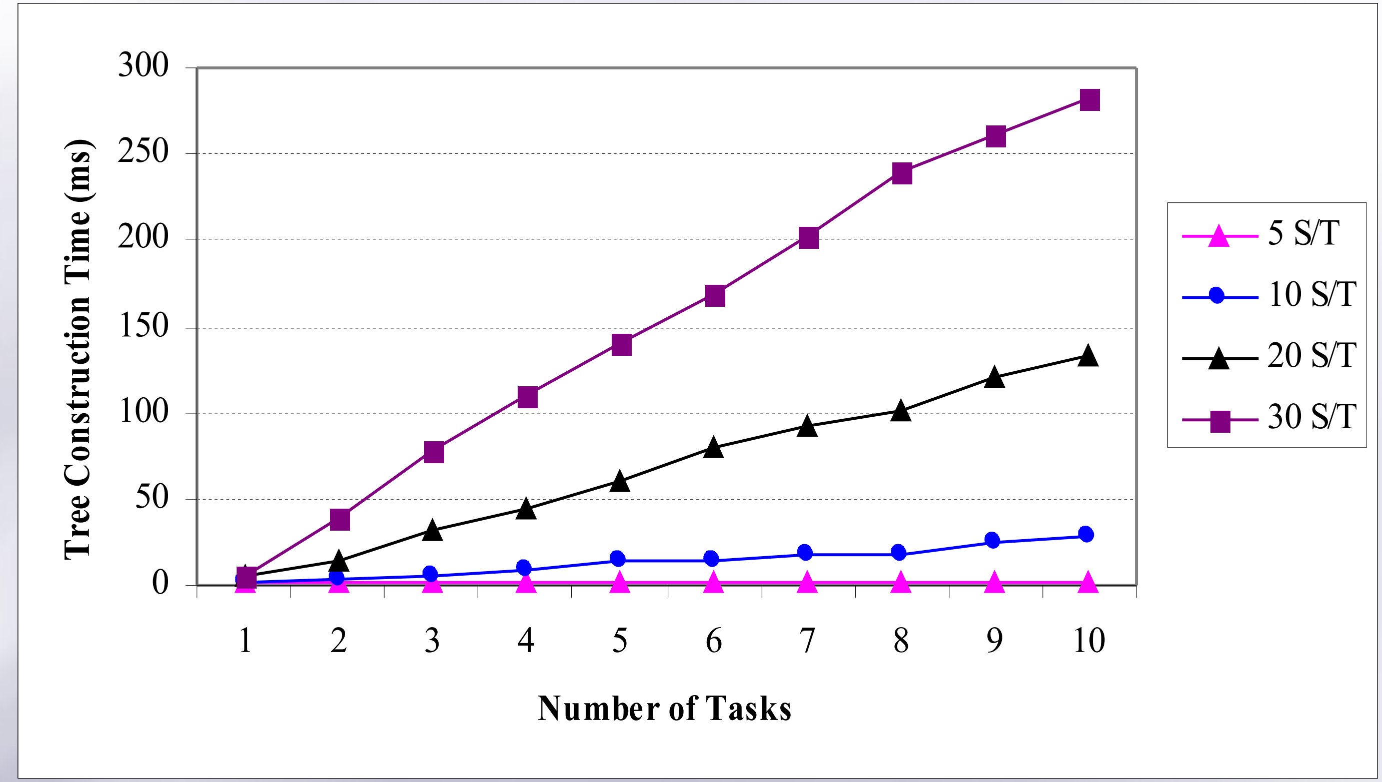
FinTime = 0
$St_0 = \{A\}$
TextSummarization
$St_1 = \{S_1, S_2, S_3\}$
TextTranslation
$St_2 = \{S_4, S_5, S_6, S_7\}$
TextToAudioTransformation
$St_3 = \{S_8, S_9, S_{10}, S_{11}\}$
$St_4 = \{Z\}$
Txt
Txt
Txt
MP3
WAV
The list of adaptation tasks which are produced by the CAS module for the example scenario are:
1.8578
4.5845
6.6948
6.5786
6.8412
7.3589
8.4759
8.8658
8.7236
8.6032
8.1423
8.4765
8.7896
8.2236
8.6465
11.4598
11.1245
11.3489
12.8598
12.5245
12.7489

The APG Module…

- The APG module selects the optimal service composition plan, in terms of execution time, during tree construction
- 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:

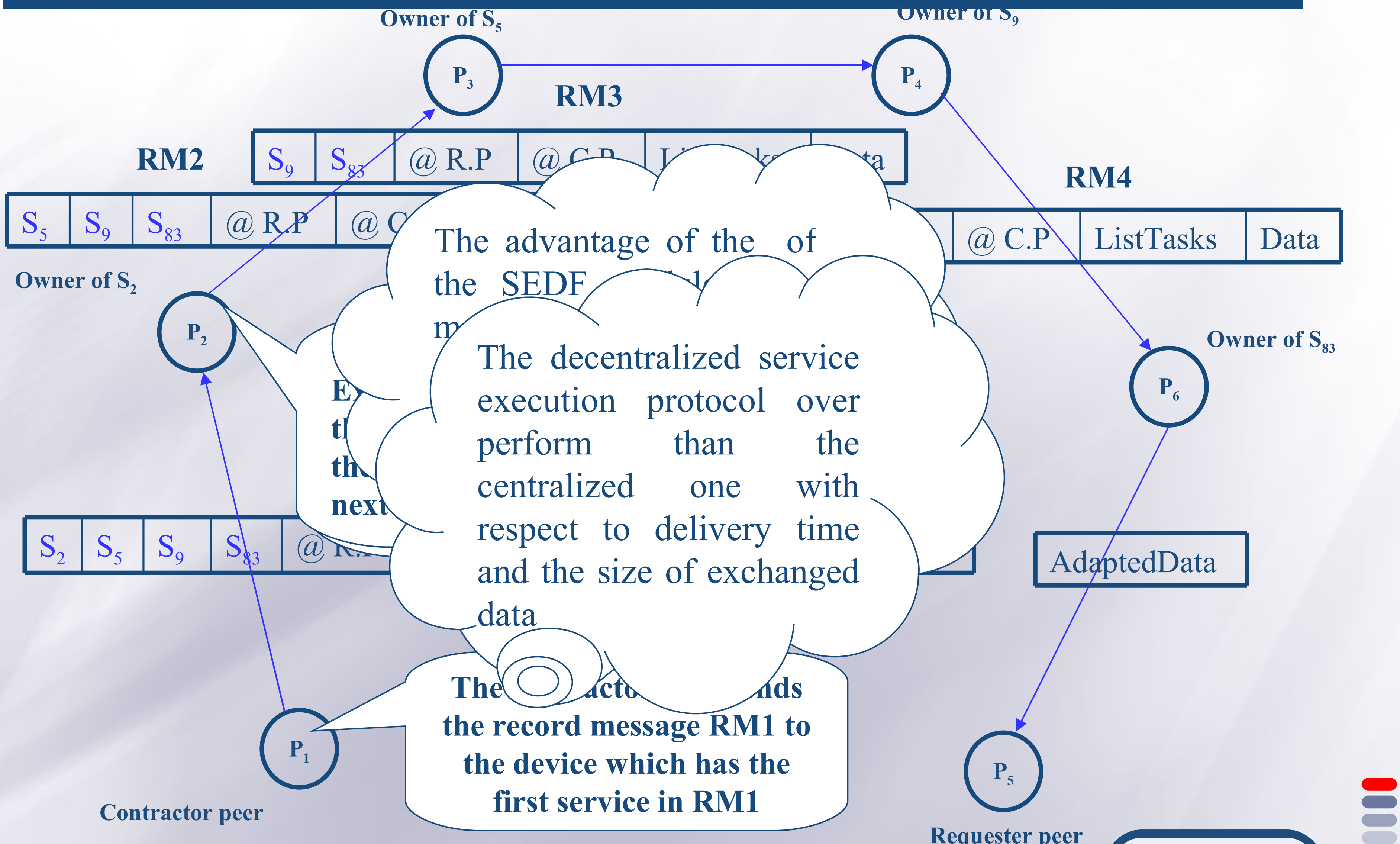
S_2,S_5,S_9,S_{83}	@ R.P	@ C.P	ListTasks	Data to be adapted

Performance of tree construction algorithm

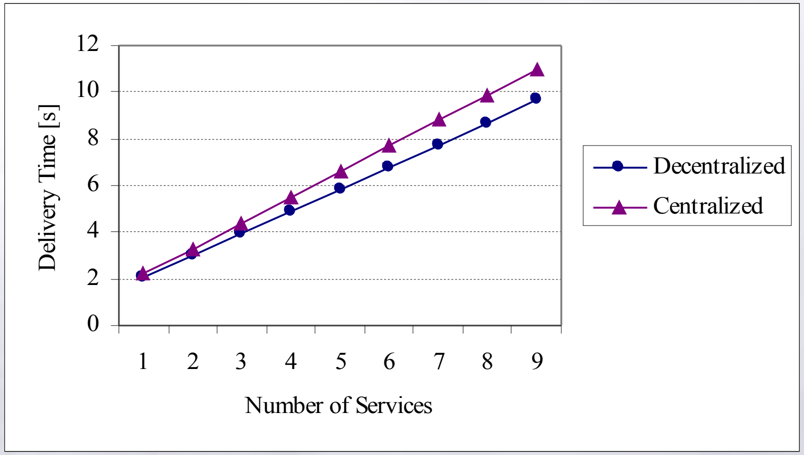


Tree Construction time versus number of tasks and number of services per task

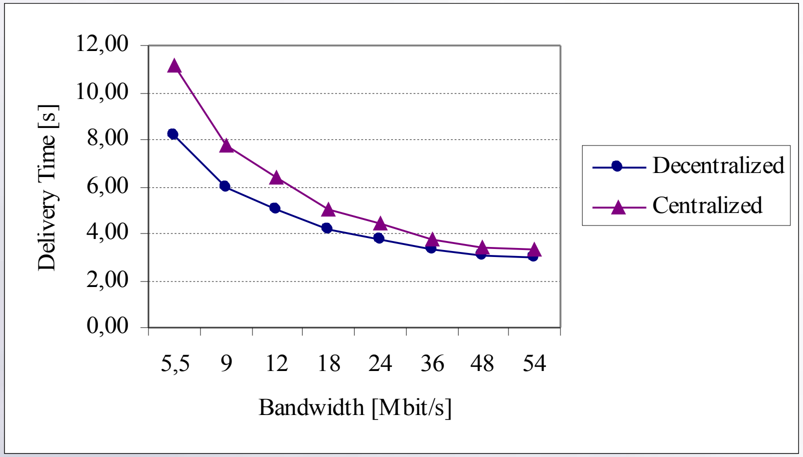
Decentralized Composite Service Execution Protocol



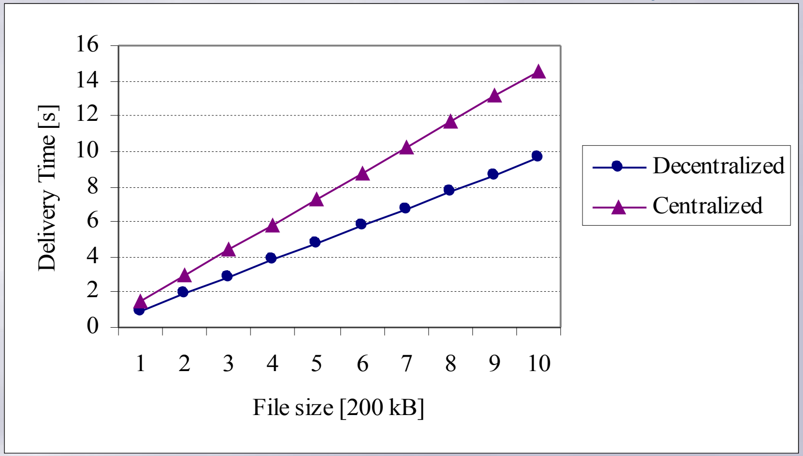
Comparison of Composite Service Execution Protocols



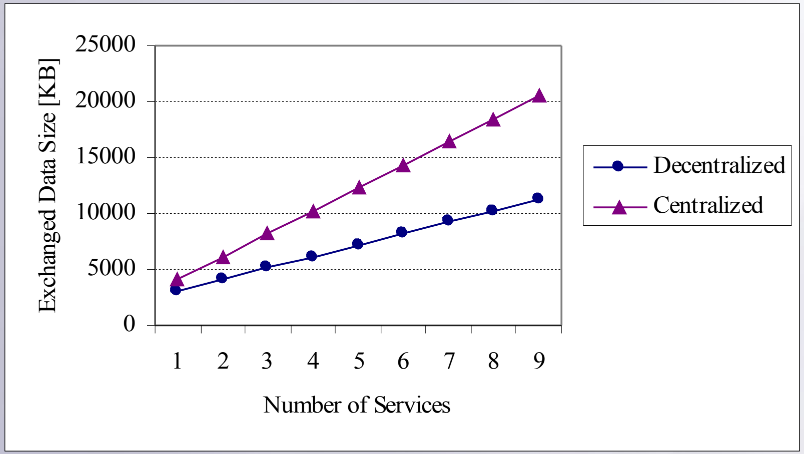
Data delivery time versus number of services (Bandwidth= 54Mb/s, file size= 1 Mbytes)



Data delivery time versus bandwidth (Filesize=1Mbytes, three services are considered)



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

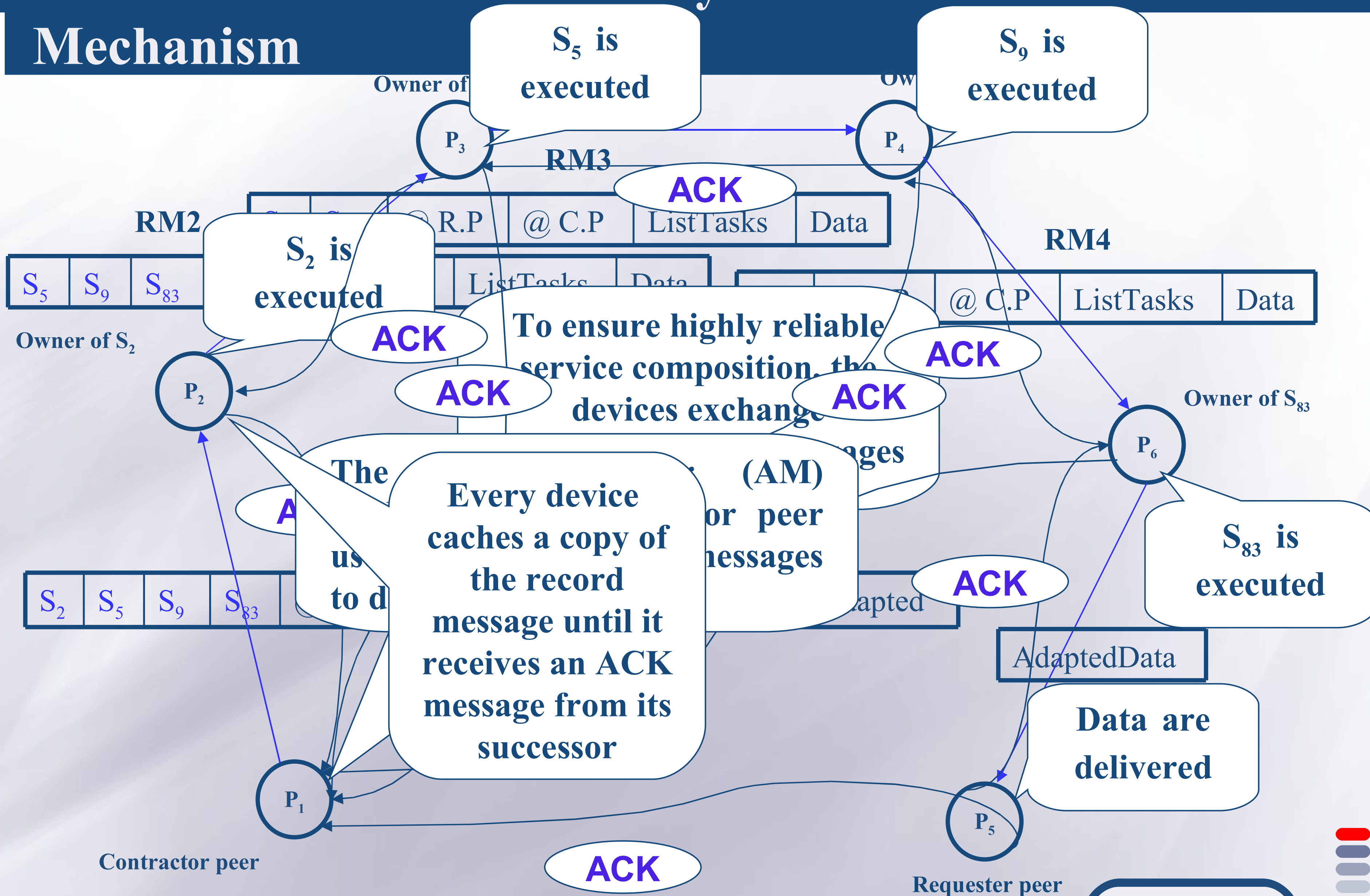


Exchanged data size versus number of services (Bandwidth= 54Mb/s, file size= 1 Mbytes)

The Need for a Fault Detection and Recovery Mechanism !!!

- 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



Fault detection and recovery mechanism…

- AM Module
  - Time estimation for receiving Ack message from the service S_9:

    $$EstTime = finshingTime(S_9) + Threshold$$

  - 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…

$St_0 = \{A\}$

FinTime = 0

A

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

TextSummari…

$St_1 = \{S_1, S_2, …$

TextTranslation…

$St_2 = \{S_4, S_5, S_6…$

7.3589

TextToAudioTr…

$St_3 = \{S_8, S_9, S_{10}…$

8.4759

Fault Recovery Algorithm

Input:

- T: an adaptation tree
- OP: nodes found in the optimal path of T
- 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 N

Output:

- newS: services in the new record message

Begin

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

…..

End

S_9 S_{10}

S_{83} S_{84}

S_3

Txt

…ces S_9 and S_{83} are …ible because the …s P_4 and P_6 are …connected.

Txt

8.7896 8.2236 8.6465

S_8 S_9 S_{10}

MP3

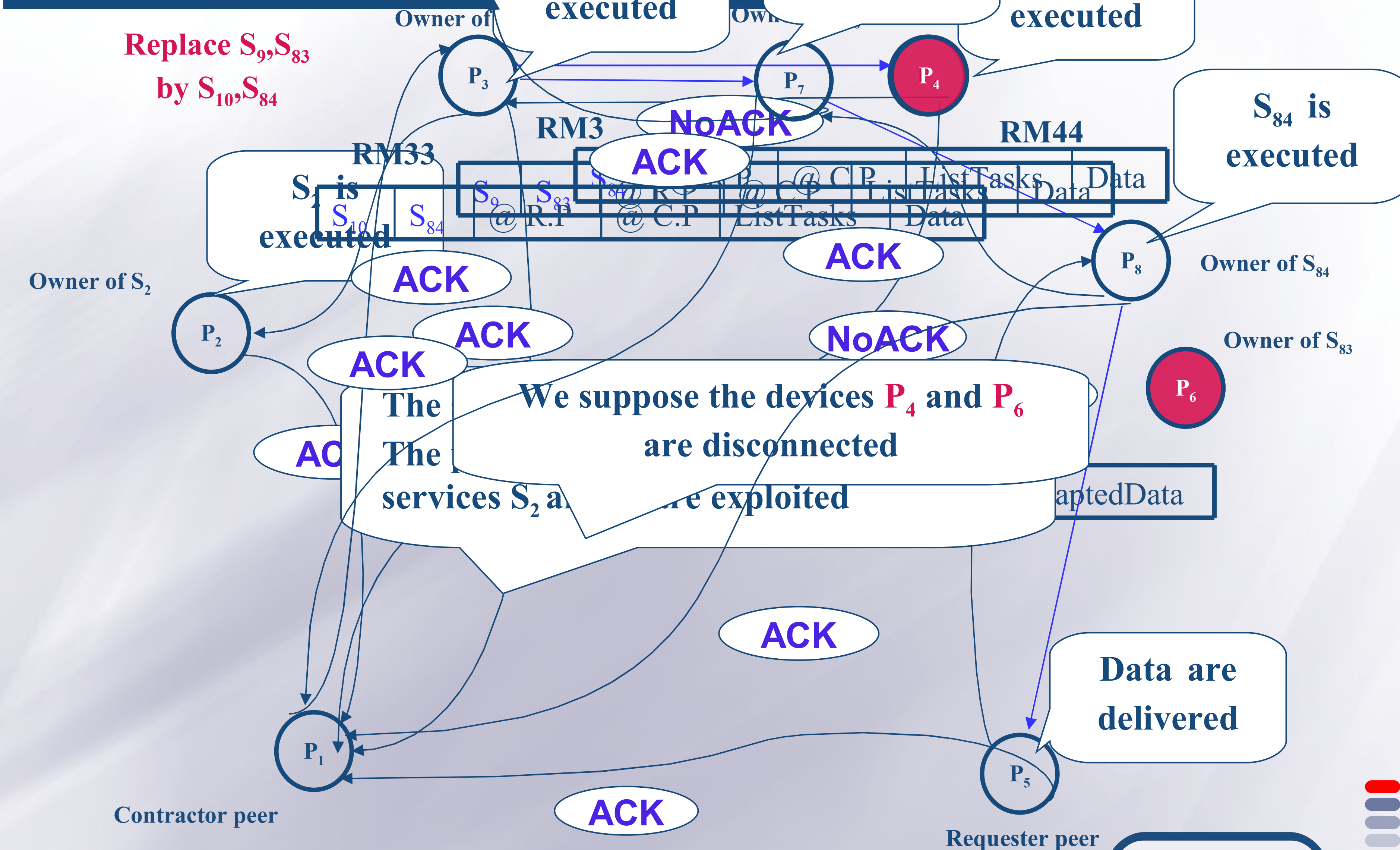
The services S_9 ,S_{83} are replaced by the services S_{10},S_{84}

12.8598 12.5245 12.7489

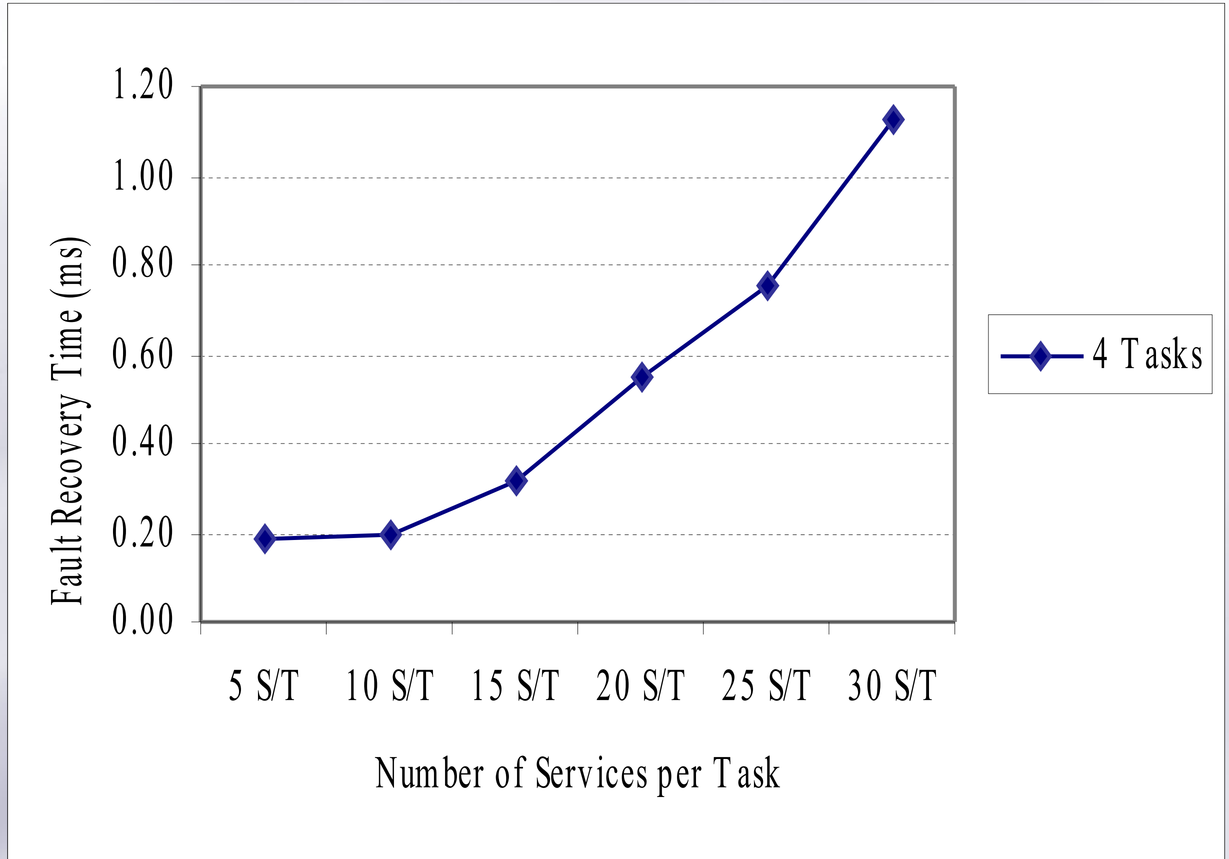
$St_4 = \{Z\}$

Z Z Z

Fault Detection and Recovery Mechanism…



Performance of the Fault Recovery Algorithm



Fault recovery time versus number of services per task

Our Approach VS Existing Service Composition Approaches

	eFlow	DCAF	HTGA	BDSCP	FTSCP	MoSCA	OUR APPROACH
Infrastructure-less	-	-	++	++	++	++	
Context-Awareness	-	+	-	-	-	-	
Optimization of execution time	-	++	-	-	-	-	
Reliability	+	-	+	+	+	+	
Decentralized Services Execution	-	-	-	-	-	-	
Fault Tolerance	+	-	++	++	+	-	

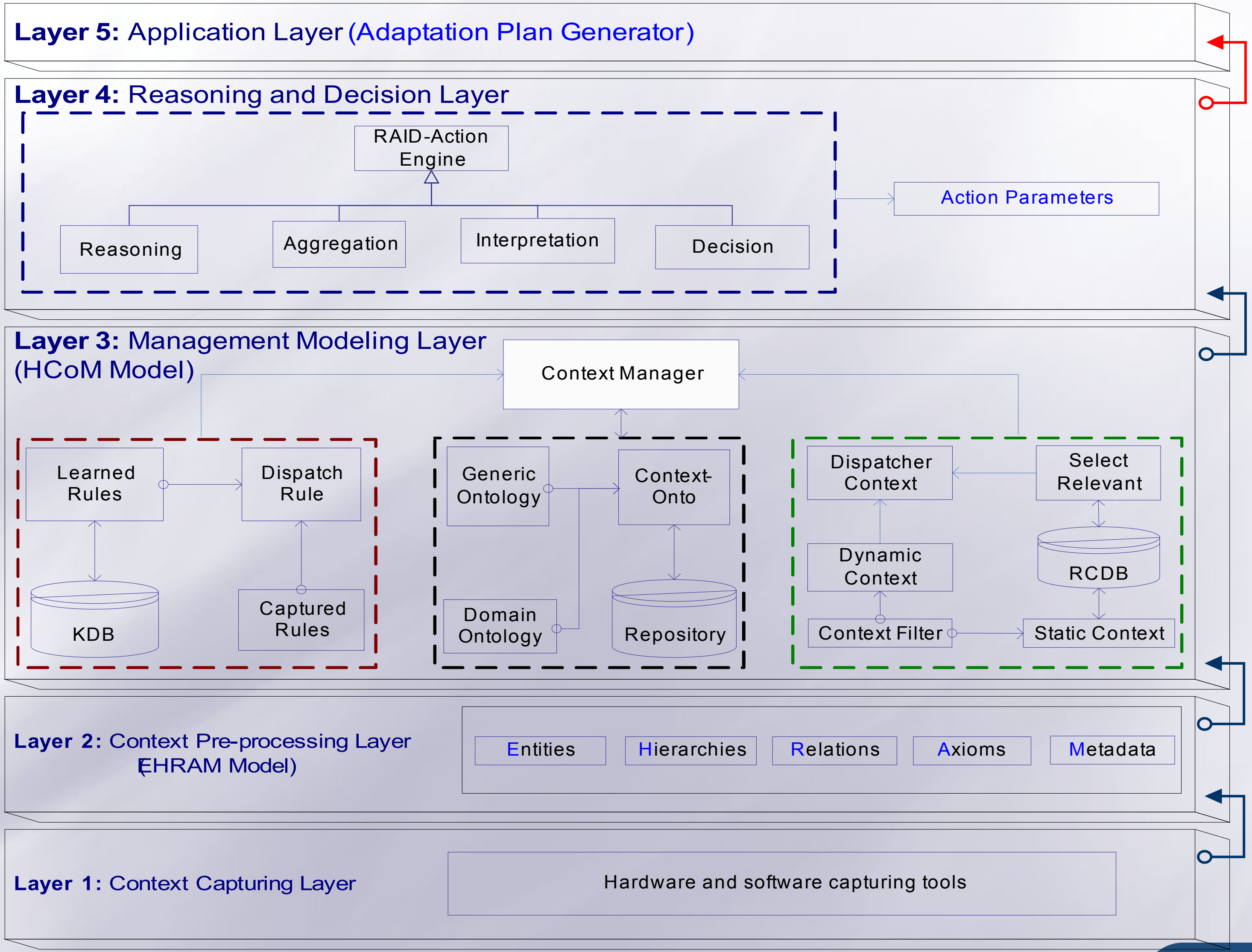
(++ Comprehensive + Partial - Limited or none)

Conclusion

- **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
  - Optimises 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)



Example

…

Adaptation Services

TaskID	Service ID	Time to leave (s)	DeviceID	Functionality
t_1	S_1	1200	P_6	Text Summarization
	S_2	1000	P_2	
	S_3	600	P_7	
t_2	S_4	900	P_8	Text Translation
	S_5	700	P_3	
	S_6	800	P_5	
	S_7	5	P_9	
t_3	S_8	300	P_1	Text to Audio Transformation
	S_9	600	P_4	
	S_{10}	1500	P_7	
	S_{11}	5	P_9	

Mapping required tasks to services

$t_0 \rightarrow St_0 = \{A\}$

$t_1 \rightarrow St_1 = \{S_1, S_2, S_3\}$

$t_2 \rightarrow St_2 = \{S_4, S_5, S_6, S_7\}$

$t_3 \rightarrow St_3 = \{S_8, S_9, S_{10}, S_{11}, S_{12}\}$

$t_4 \rightarrow St_4 = \{Z\}$

Neutralisation Services

Service ID	Time to leave (ms)	Device ID	Functionality
S_{80}	500	P7	Text Format Conversion
S_{81}	600	P6	
S_{82}	700	P1	Audio Format Conversion
S_{83}	600	P6	
S_{84}	1100	P8	