Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2008, Article ID 851628, 10 pages
doi:10.1155/2008/851628

Research Article

Efficient Execution of Service Composition for Content Adaptation in Pervasive Computing

Yaser Fawaz,[1] Girma Berhe,[2] Lionel Brunie,[1] Vasile-Marian Scuturici,[1] and David Coquil[3]

[1] *Laboratoire LIRIS-UMR-CNRS 5205, INSA de Lyon, 7 Avenue Jean Capelle, 69621 Villeurbanne Cedex, France*
[2] *Department of Computer Science and Communication (CSC), University of Luxembourg, Campus Kirchberg, 6 Rue Richard Coudenhove-Kalergi, 1359 Luxembourg, Luxembourg*
[3] *University of Passau, Innstraße 43, 94032 Passau, Germany*

Correspondence should be addressed to Yaser Fawaz, yaser.fawaz@insa-lyon.fr

Received 5 April 2008; Accepted 25 June 2008

Recommended by Harald Kosch

Multimedia content adaptation has been proved to be an effective mechanism to mitigate the problem of devices and networks heterogeneity and constraints in pervasive computing environments. Moreover, it enables to deliver data taking into consideration the user's preferences and the context of his/her environment. In this paper, we present an algorithm for service composition and protocols for executing service composition plan. Both the algorithm and the protocols are implemented in our distributed content adaptation framework (DCAF) which provides a service-based content adaptation architecture. Finally, a performance evaluation of the algorithm and the protocols is presented.

1. INTRODUCTION

There is a huge amount of multimedia information being captured and produced for different multimedia applications, and the speed of generation is constantly increasing. While providing the right information to a user is already difficult for structured information, it is much harder in the case of large volume of multimedia information. This is further complicated with the emergence of new computing environments. For example, a user may want to access content on the network through a handheld device connected by wireless link or from a high-end desktop machine connected by broadband network. The content that can be presented on one device might not be necessarily viewable on another device unless some content transformation operations are applied [1].

An efficient content delivery system must be able to adapt the delivered content for every client in every situation in order to address the wide range of clients, minimal bandwidth requirement, and fast real-time delivery [2]. As a consequence, content adaptation is one of the research topics that have attracted a number of multimedia research works. Here, we focus on issues related to content adaptation in pervasive computing systems.

Several adaptation approaches have been developed to perform content adaptation for pervasive computing. These approaches are generally classified into: server-based [3, 4], client-based [5, 6], and proxy-based [7–9]. As server-based adaptation approach degrades the performance of the server, and client-based adaptation approach is very difficult and sometimes impossible due to the limited processing power of pervasive devices (e.g., smartphones, PDAs), most of existing adaptation systems implement a proxy-based approach. Furthermore, to alleviate the overload problem of content adaptation processing, distributed approaches were proposed such as Ninja [10], MARCH [11], DANAE [12], and DCAF [13].

Jannach and Leopold [14] proposed a server-side multimedia content adaptation framework which performs the content adaptation by composing adaptation services resident on the server itself. This approach results in computational load and resource consumption on the server so it decreases the performance of content delivery. However, in our architecture (DCAF), content adaptation is performed externally using Internet accessible adaptation services which enhance the performance of the system in terms of content delivery time, resources consumption, and network overhead.

While the DCAF architecture [13] provides a distributed adaptation mechanism, its centralized control manner of adaptation services execution impacts the overall performance of the system in delivering the adapted data to the user. In order to enhance the performance of the system, a decentralized service execution protocol is incorporated to the DCAF architecture. In this paper, we present the centralized and decentralized service execution protocols, and the result of the experiments that have been done to compare their performances.

The rest of the paper is organized as follows. In Section 2, we discuss related works. Section 3 presents an overview of the distributed content adaptation framework (DCAF). The multimedia adaptation graph generator (MAGG) algorithm is presented in Section 4. In Section 5, we present the composite service execution protocols. The results of the experiments which have been done to evaluate the performance of the MAGG algorithm and the composite service execution protocols are presented in Section 6. In Section 7, we discuss fault tolerance issues in DCAF architecture. Finally, in Section 8, we conclude the paper and highlight some future works.

2. RELATED WORKS

As we have mentioned in Section 1, the existing adaptation frameworks are categorized into three groups: server-based approach, proxy-based approach, and client-side approach. In the following, we present each of these approaches.

In the case of server-based approach (e.g., [3, 4, 15]), the functionality of the traditional server is extended by adding content adaptation. In this approach, both static (offline) and dynamic (on-the-fly) content adaptations can be applied and better adaptation results could be achieved as it is close to the content; however clients experience performance degradation due to additional computational load and resource consumption on the server [16].

In proxy-based approach (e.g., [8, 16, 17]), a proxy, that is between the client and the server, acts as a transcoder for clients with similar network or device constraints. The proxy makes a request to the server on behalf of the client, intercepts the reply from the server, decides on and performs the adaptation, and then sends the transformed content back to the client. In this approach, there is no need of changing the existing clients and servers. The problem of proxy-based adaptation approaches is that most of them focus on a particular type of adaptation such as image transcoding, HTML to WML conversion, and so forth, and that they are application specific. In addition, if all adaptations are done at the proxy, it results in computational overload as some adaptations are computational intensive and this degrades the performance of information delivery like the server-based approach.

Client-based approach (e.g., [5, 6, 18]) can be done in two ways: transformation by the client device or selection of the best representation after receiving the response from the origin server. This approach provides a distributed solution for managing heterogeneity since all clients can locally decide and employ adaptations most appropriate to them. However, adaptations that can benefit a group of clients with similar request can be more efficiently implemented with server-based or proxy-based approaches. Furthermore, all of the clients may not be able to implement content adaptation techniques due to processor, memory resource constraint, and limited network bandwidth [19].

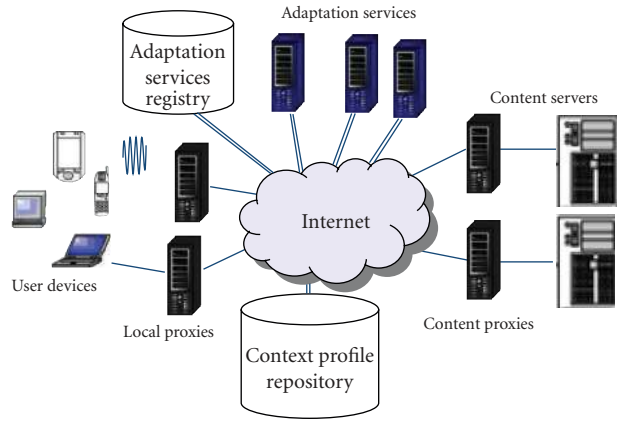

Figure 1: Distributed content adaptation framework (DCAF).

The above three approaches do not deal with the problem of content adaptation from a service perspective that can be commercialized and utilized by users, content providers, or other service providers (like Internet service providers). Introducing content adaptation as a service (service-based adaptation approach) distributes the activities and results in performance enhancement. It also opens new opportunities to service providers as additional revenue.

A service-based content adaptation approach is quite recent. There are very few works on distributed content adaptation mechanism. Nevertheless, we can cite: Ninja [10], MARCH [11], DANAE [12], and DCAF [13]. In our previous work [13], we proposed a service-based architecture (DCAF). In this architecture, the adaptation tools are developed externally by third party or service providers. While the architecture provides a distributed adaptation mechanism, the centralized control of the execution of these services impacts the overall performance of the system in delivering the adapted data to the user. In order to enhance the performance of the system, a decentralized service execution protocol is introduced. In this paper, we present this protocol and the result of the comparison done with the centralized service execution protocol.

3. OVERVIEW OF THE DCAF ARCHITECTURE

3.1. Components of the DCAF architecture

As displayed in Figure 1, the DCAF architecture is composed of six components. The description of these components is summarized as follows.

Content servers (CSs): they are standard data repositories such as web sites, databases, and media servers.

Content proxies (CPs): they provide access to content servers, formulate user request to source format, manage and provide content description (meta-data).

Context profile repository (CPR): it stores the users' preferences and the device characteristics. Users can update and modify their profiles at any time. Dynamic data such as the user location and the network conditions are determined at request execution.

Adaptation service registry (ASR): it is like Universal Description, Discovery and Integration (UDDI) registry; it stores multimedia adaptation services description including the semantic information (e.g., the type of data the service handles) and allows for service look up.

Adaptation services proxies (ASPs): they host adaptation tools. In our framework, ASPs are implemented as Web Services.

Local proxies (LPs): they are access points to information delivery systems. They are in charge of retrieving and processing context profile (user, device, and network), decide the type and number of adaptation processes, discover appropriate adaptation services, and plan execution of the services.

3.2. Context, content, and service descriptions

The decision of the adaptation process depends on the quality of information gathered from various sources. This information consists of context description, content description, and adaptation service descriptions.

(1) Context description (*context metadata*)

It includes the device profile, the user profile, the network profile, and other dynamic context data like location, sensor information, and so forth. We have used the CSCP [20] standard to represent the context information. Figure 2 shows an example of partial context profile for a sample device.

(2) Content description (*content metadata*)

The metadata contains both the information about the object like object's title, description, language, and so forth, and feature data about the media itself including the media type, the file format, the file size, the media dimensions, the color information, the location, and so forth. For content description, the XML form of MPEG-7 description is used. Figure 3 shows an example of the description for an image data.

(3) Adaptation service description

It contains information about the service such as name, identifier, function, processing rate, processing rate is the amount of data processed per second. It is expressed in Kbytes per second, cost, and so forth. In order to describe adaptation services, we developed multimedia adaptation service ontology [21]. This ontology facilitates describing adaptation services semantically so that they can be discovered, selected, composed, and invoked automatically. Figure 4 presents an example of a description of audio to text adaptation service using the described ontology.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContextProfile rdf:ID=”Context”>
  <device>
  <dev:DeviceProfile>
    <dev:Hardware>
      <dev:ScreenWidth>320</dev:ScreenWidth>
      <dev:ScreenHeight>200</dev:ScreenHeight>
    </dev:Hardware>
    <dev:Software>
      <dev:OperatingSystem>
        <dev:Name>EPOC </dev:Name>
        <dev:Version>2.0</dev:Version>
        <dev:Vendor>Symbian</dev:Vendor>
      </dev:OperatingSystem>
      <dev:UserAgent>
        <dev:Type>Browser</dev:Type>
        <dev:Name>Mozilla</dev:Name>
        <dev:Version>5.0</dev:Version>
        <dev:Vendor>Symbian</dev:Vendor>
      </dev:UserAgent>
    </dev:Software>
  </dev:DeviceProfile>
  </device>
</ContextProfile>
```

Figure 2: Partial example of context profile.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentDescription>
  <MultimediaContent type=“Image”>
    <MediaInformation ID=“x_ray_image”>
    <MediaUri> http:/localhost/imagefiles/gorge .jpeg
    </MediaUri>
    <MediaTitle>Examen de la gorge</MediaTitle>
    <MediaDescription> Patient atteint d'un cancer de la
gorge…  </MediaDescription>
    <MediaLanguage>French</MediaLanguage>
    </MediaInformation>
    <MediaProfile>
      <MediaFormat>JPEG</MediaFormat>
      <MediaSize>61.6 </MediaSize>
      <MediaWidth>660</MediaWidth>
      <MediaHeight>445</MediaHeight>
      <MediaColor>24</MediaColor>
    </MediaProfile>
  </MultimediaContent>
</ContentDescription>
```

Figure 3: Sample multimedia content description using MPEG-7 standard.

4. MULTIMEDIA ADAPTATION GRAPH GENERATOR (MAGG)

In a multimedia content adaptation framework like DCAF, the challenge is that there is no single complete software solution that can satisfy all adaptation needs. In order to solve this problem, adaptation services are composed to realize the required adaptation. Service composition is defined as the process of putting together atomic/basic services to perform

complex tasks. For example, to transform English text into audio in French, we need the composition of a language translation service and a text to audio conversion service. Since an adaptation process can be carried out in a number of adaptation steps (adaptation tasks) and there could be several adaptation services that execute each adaptation task that leads to different service composition possibilities and makes services composition difficult. In order to solve this problem, we have developed an algorithm called a multimedia adaptation graph generator (MAGG) that can compose distributed multimedia adaptation services.

4.1. Service composition modelling: definitions and notations

Definition 1 (media object). A media object is a multimedia data item which can be a text, an image, an audio, or a video represented as $M(m_1, m_2, \ldots, m_n)$ where $m_1, m_2, \ldots, m_n$ are media features or metadata.

Definition 2 (state). The state S of a media object M, denoted as $S(M)$, is described by the metadata values. For example, for an image object the state holds the values for the format, the color, the height, the width, and so forth.

For example, $S(M)$ = (bmp, 24 bits, 245 pixels, 300 pixels).

Definition 3 (adaptation task). An adaptation task is an expression of the form $t(a_1 a_2 \cdots a_n)$ where t is a transformation and $a_1, a_2, \ldots, a_n$ are parameters.

For example, ImageFormatConversion (imageIn, imageOut, oldFormat, newFormat), where

(i) imageIn: image input file,

(ii) imageOut: image output file,

(iii) oldFormat: old file format,

(iv) newFormat: new file format.

Definition 4 (adaptation service). An adaptation service is a service described in terms of inputs, outputs, preconditions, and effects. An adaptation service is represented as $s = (R\ I\ O\ Pre\ Eff\ Q)$, *where*

(i) *R:* an atomic process that realizes an adaptation task,

(ii) *I:* input parameters of the process,

(iii) *O:* output parameters of the process,

(iv) *Pre:* preconditions of the process,

(v) *Eff:* effects of the process,

(vi) *Q:* quality criteria of the service.

Definition 5 (operator (plan operator)). A plan operator is an expression of the form $o = (h(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m), Pre\ Eff\ Q)$, where

(i) h: an adaptation task realized by an adaptation service with input parameters $a_1, a_2, \ldots, a_n$ and output parameters $b_1, b_2, \ldots, b_m$. h is called the head of identification of the operator,

```
<AtomicProcess rdf:ID="AudioToTextAdapationService">
  <hasInput rdf:ID="#InputAudio">
    <parameterType datatype="&xsd;#anyURI">
  </hasInput>
  <hasOutput rdf:ID="#OutputText">
    <parameterType datatype="&xsd;#anyURI">
  </hasOutput>
  <hasPrecondition>
    <hasExpression rdf:ID="InputAudioFormat">
      <hasSubject rdf:resource="#InputAudio"/>
      <hasProperty rdf:resource="#Audio"/>
      <hasFormat rdf:datatype="&xsd; #string">#WAV
      </hasFormat>
    </hasExpression>
  </hasPrecondition>
  <hasServicePrice>
    <parameterType datatype="&xsd;#float">25
  </hasServicePrice>
  <hasServiceTime>
    <parameterType datatype="&xsd;#float">200
  </hasServiceTime>
</AtomicProcess>
```

Figure 4: An expert of service description for Audio to Text adaptation service.

(ii) *Pre*: represents the operator's preconditions,

(iii) *Eff*: represents the effect of executing the operator,

(v) $Q = \{q_1, q_2, \ldots, q_n\}$: represents quality attributes (e.g., cost, processing rate, etc.).

Let S be a state, t be an adaptation task, and M be a media object. Suppose that there is an operator o with head h that realizes t such that *Pre* of o is satisfied in S. Then, we say that o is applicable to t, and the new state is given by

$$S(M_o) = \text{Executing } (o, M, S). \tag{1}$$

Example: for the above given adaptation task, we can have an adaptation operator instance as follows.

Operator:ImageFormatConversionOperator (http://media-adaptation/imagefiles/image1, http://media-adaptation/imagefiles/image2, mpeg, bmp).

(i) Input: http://media-adaptation/imagefiles/image1.

(ii) Output:http://media-adaptation/imagefiles/image2.

(iii) Precondition: hasFormat (http://media-adaptation/imagefiles/image1, mpeg).

(iv) Effect: hasFormat (http://media-adaptation/image files/image2, bmp).

(v) Quality: (cost = 30 units, processing rate = 1500 kbyte/s).

Definition 6 (adaptation graph). An adaptation graph $G(V, E)$ is a directed acyclic graph (DAG), where

(i) V is the set of nodes that represent the adaptation operators,

(ii) E is the set of edges that represent the possible connections between the adaptation operators.

The start node $A \in V$ is a pseudo operator with effect (initial state) but no precondition.

The end node $Z \in V$ is a pseudo operator with precondition (goal state) but no effect.

Remark 1. Let $o_i \in V$ and $o_j \in V$, a link or an edge e_{ij} exists from o_i to o_j if the following condition is satisfied:

$$o_j \cdot Pre \subseteq o_i \cdot Eff, \tag{2}$$

where

(i) $o_j \cdot Pre$ denotes preconditions of o_j,

(ii) $o_i \cdot Eff$ denotes effects of o_i.

Definition 7 (adaptation planning problem). An adaptation planning problem is a four-tuple (S_A, S_Z, T, D), where S_A is the initial state of the media object, S_Z is the goal state of the media object, T is an adaptation task list, and D is the adaptation operators. The result is a graph $G = (V, E)$.

Definition 8 (adaptation path). An adaptation path is a path in the adaptation graph G that connects the start node to the end node. It is represented as a list of the form $p = (A, o_1, o_2, \ldots, o_n, Z)$, where A and Z are the start and the end nodes and o_i is an adaptation operator instance.

The MAGG algorithm consists of different procedures and functions. In Algorithm 1, we present only the structure of the algorithm. See [21] for complete listing of the algorithm. This algorithm is used to construct a multimedia adaptation graph which gives all service composition possibilities that satisfy the required adaptation needs.

4.2. Optimal adaptation path search

Since an adaptation task can be achieved by more than one service, the services are represented by operators in the graph, and each service has different QoS, choosing an appropriate service is an obvious requirement. Once the adaptation graph that consists of all possible compositions is generated, then the choice of the optimal adaptation path (also called the service composition plan) in the graph is done based on user specified QoS criteria [13]. The QoS is a multidimensional property which may include service response, service charge, quality of received data, and so forth. Here, the QoS represents only the service charge (cost) and waiting time. For a service s executing an adaptation task t, the QoS is defined as follows:

$$Q(s) = (s_{\text{cost}}(s,t), s_{\text{time}}(s,t)), \tag{3}$$

where

(i) $s_{\text{cost}}(s,t)$: the adaptation service execution cost,

(ii) $s_{\text{time}}(s,t)$: the adaptation service execution time and the data transmission time.

Let $p = (A, s_1, s_2, \ldots, s_n, Z)$ be a path in an adaptation graph, where n is the number of services in the adaptation path. We define the QoS of the path p, denoted as $Q(p)$, as follows:

$$Q(p) = (Q_{\text{cost}}(p), Q_{\text{time}}(p)), \tag{4}$$

where

$$\begin{aligned} Q_{\text{cost}}(p) &= \sum_{i=1}^{n} s_{\text{cost}}(s_i, t_i), \\ Q_{\text{time}}(p) &= \sum_{i=1}^{n} s_{\text{time}}(s_i, t_i). \end{aligned} \tag{5}$$

To aggregate the quality values, we define scaled qualities $Qs_{\text{cost}}(s_i)$ and $Qs_{\text{time}}(s_i)$ as

$$\begin{aligned} Qs_{\text{cost}}(s_i) &= \begin{cases} \dfrac{Q_{\text{cost}}^{\max} - Q_{\text{cost}}(s_i)}{Q_{\text{cost}}^{\max} - Q_{\text{cost}}^{\min}}, & \text{if } Q_{\text{cost}}^{\max} - Q_{\text{cost}}^{\min} \neq 0, \\ 1, & \text{if } Q_{\text{cost}}^{\max} - Q_{\text{cost}}^{\max} = 0, \end{cases} \\ Qs_{\text{time}}(s_i) &= \begin{cases} \dfrac{Q_{\text{time}}^{\max} - Q_{\text{time}}(s_i)}{Q_{\text{time}}^{\max} - Q_{\text{time}}^{\min}}, & \text{if } Q_{\text{time}}^{\max} - Q_{\text{time}}^{\min} \neq 0, \\ 1, & \text{if } Q_{\text{time}}^{\max} - Q_{\text{time}}^{\max} = 0, \end{cases} \end{aligned} \tag{6}$$

where $Q_{\text{cost}}^{\max}$ and $Q_{\text{cost}}^{\min}$ are the values of the maximum and the minimum costs, respectively. $Q_{\text{time}}^{\max}$ and $Q_{\text{time}}^{\min}$ are the values of the maximum and the minimum times, respectively.

Users can give their preferences on QoS by specifying weight values for each criterion. The score of a path with weighted values is calculated as in

$$\text{Score } (p) = \sum_{i=1}^{n} (Qs_{\text{cost}}(s_i) * w_{\text{cost}} + Qs_{\text{time}}(s_i) * w_{\text{time}}), \tag{7}$$

where $w_{\text{cost}} \in [0,1]$ and $w_{\text{time}} \in [0,1]$ represent the weight values assigned to the cost and the time, respectively, and $w_{\text{cost}} + w_{\text{time}} = 1$.

Let $P_{\text{set}} = \{p_1, p_2, \ldots, p_K\}$ be the set of all possible paths in an adaptation graph, then the optimal path is the path with the maximum score value $\text{score}_{\max}$, where $\text{score}_{\max}$, is defined as follows: $\text{Score} = \max\{\text{Score}(p_i)\}; i \in \{1, k\}$.

Dijkstra's algorithm [22] was used to find the optimal path, that is, the path in the graph with the maximum score value $\text{score}_{\max}$. More information about the optimal path selection using Dijkstra's algorithm is found in [21].

5. COMPOSITE SERVICE EXECUTION PROTOCOLS

The execution of the composite service (also called the service composition plan) can be done in centralized or decentralized approaches. In the centralized approach (also called a star-based approach), the exchange of data between the services is done through the use of a broker as an intermediator [23, 24]. In the decentralized approach (also called a mesh-based approach), however, the exchange of data is done directly from one service to another one without the need to an intermediator [25]. In the following, we incorporate the two approaches with our architecture DCAF.

5.1. Centralized (star-based)

In the DCAF architecture, the local proxy acts as a broker. As presented in Figure 5, the local proxy sends the data to each

```
Algorithm: graph (S_A, S_Z, T, D)
Input: Initial state S_A, final state S_Z, adaptation task list T and adaptation operators D
Output: an adaptation graph G(V, E)
// Global constant
// Limit  maximum number of neutral operators allowed in a connection
// Global variables
// V          a set of nodes in a graph
// E          a set of edges in a graph
// ao         start node
// zo         end node
// NO         a set of neutral operators available in the system
// Local variables
// T          a set of adaptation tasks
// t          an adaptation task element of T
// O          a set of nodes for adaptation operators realizing an adaptation task
// PO         a set of parent nodes
// P_z        a set containing the end node
Var
V, E, T, t, O, PO, ao, zo, Pz, NO
Begin
   ao = ConstructStartNode(S_A) // constructs the start node from the initial state
   zo = ConstructEndNode(S_Z) // constructs the end node from the goal state
   NO = ConstructNeutralOperators() // returns the list of the neutral operators available in // the system
   V = {ao}      // initialization
   E = Ø         // initialization
   PO = {ao}     // initialization
   for each t ∈ T
      Begin
        Construct nodes O from D with operators realizing t
        // several operators can realize a task
        Connect (O, PO)
        // after the process PO holds the value of O
      End // T is processed
   Pz = {zo}
   Connect (Pz, PO) // connects the end node
    Return G(V, E)
End // graph
```

Algorithm 1: The MAGG algorithm structure.

adaptation service proxy (ASP) in the service composition plan and gets the result from the ASPs.

The data that the local proxy gets from the ASPs are partially adapted before it is sent to the last ASP in the service composition plan. The number of communications between the local proxy and the ASPs increases due to the number of the ASPs involved in performing the content adaptation process. Hence, the forward and backward communications between the local proxy and the ASPs incur additional overhead on the overall performance of data delivering to the user.

5.2. Decentralized (mesh-based)

As shown in Figure 6, in the decentralized service execution protocol, the ASPs communicate with each other and with the local proxy by exchanging a record message (RM) [25]. The RM contains

(1) the addresses of the services that are in the optimal path of the graph,

(2) the address of the local proxy,

(3) the data to be adapted.

The exchange of records messages between the ASPs is done by using service execution and data forwarding (SEDF) modules. The SEDF module is in charge of executing local services to perform content adaptation and forwarding the record message (denoted RM_i) containing the partially adapted data for subsequent adaptation. When the ASP receives RM_i, its SEDF module does the following.

If the first service S in RM_i is found locally, the SEDF executes S by assigning the data in RM_i as input of S. Then, it removes S from RM_i and puts the output of S in the data field of RM_i.

It repeats step 1 until the first service of RM_i is not found locally or the last service in RM_i is executed.

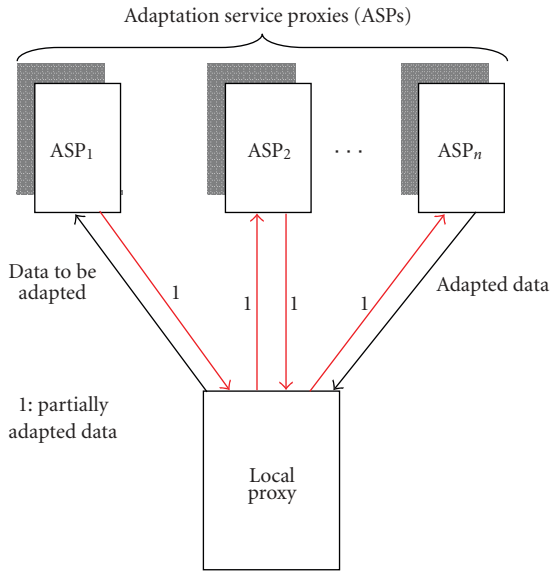

Figure 5: Centralized services execution pattern (star-based).

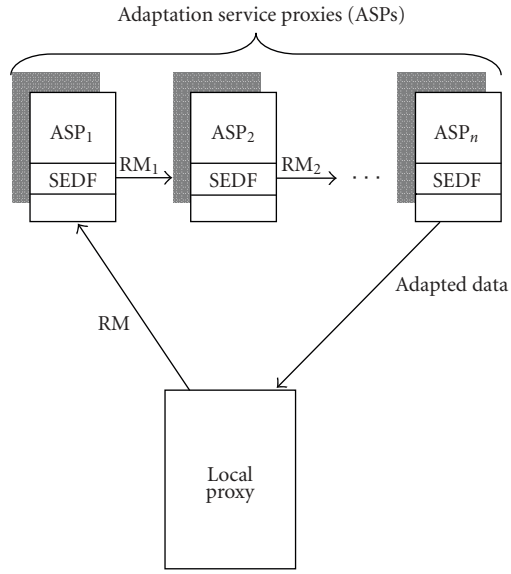

Figure 6: Decentralized services execution pattern (mesh-based).

The modified record message RM_i is forwarded to the ASP that owns the first service in RM_i. If RM_i does not contain any service which means that the adaptation process is finished, the adapted data are sent to the local proxy.

We have compared the centralized and decentralized service execution protocols using two metrics: data transmission time and size of exchanged data. The data transmission time as well as the size of exchanged data is less for the decentralized protocol since there are less number of communications. For a scenario with three ASPs, we need six and four communications for the centralized and decentralized protocols, respectively. Therefore, the decentralized protocol gives better performance than the centralized one especially when the bandwidth is small and the number of ASPs is big.

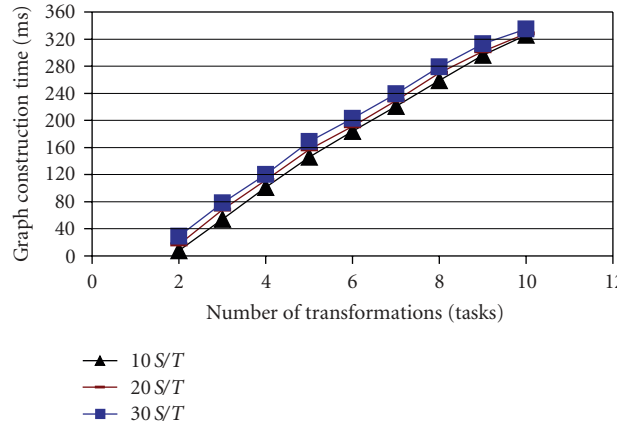

Figure 7: Graph construction time.

6. EXPERIMENTATIONS

Two major experiments were conducted. The first experiment is to study the behaviour of the graph generation algorithm with respect to the depth of the graph and the number of services per transformation. The second experiment is to measure the performance of the centralized and decentralized execution protocols. The experiments were performed on a 1.9 GHZ Pentium 4 with 256 MB RAM running Microsoft Windows 2000. In the following, we present the results of these experiments.

6.1. Graph construction

As depicted in Figure 7, the relationship between graph construction time and the number of the adaptation tasks (the adaptation transformations) is linear. Moreover, the construction time progresses slowly with the number of services per transformation (S/T). The graph construction time does not include the services execution time. It was also observed that the progress both for the depth (number of transformations) and the width (number of services per transformation) was almost constant with average increase of 40 milliseconds for each depth and 10 milliseconds for each 10 additional services. This implies that having several services per transformation does not affect much on the total construction time, while it provides the possibility to select the best service among the candidates.

The performance of the graph construction algorithm is reasonable even for the maximum depth of the graph (graph depth equals 10). For example, if we consider a graph with depth and width equal to 10 and 30, respectively, the construction time is only 335 milliseconds which is really acceptable. Nevertheless, most adaptation scenarios have 5 or 6 depth graph which is enough to realize any possible type of adaptation.

6.2. Service composition plan execution

A simulation has been made to compare the performance of the composite service execution protocols in terms of data delivery time and exchanged data size. The data delivery

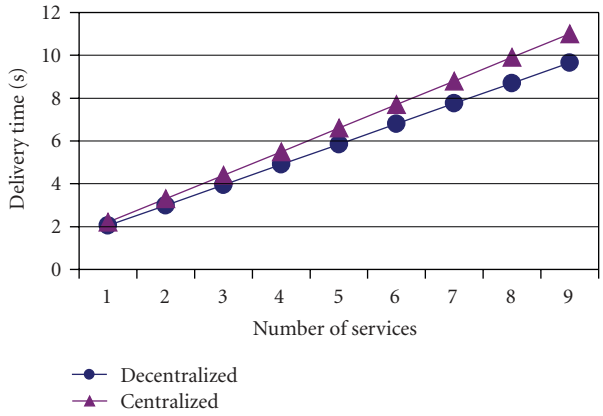

Figure 8: Data delivery time versus number of services. (Bandwidth = 54 Mbps, file size = 1 Mbyte.)

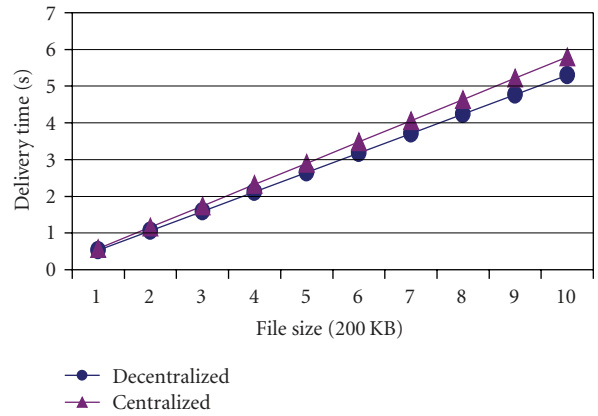

Figure 9: Data delivery time versus file size. (Bandwidth = 54 Mbps, three services are considered.)

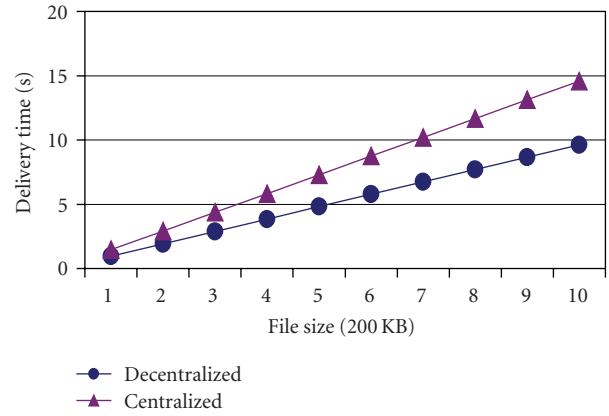

Figure 10: Data delivery time versus file size. (Bandwidth = 5.5 Mbps, three services are considered.)

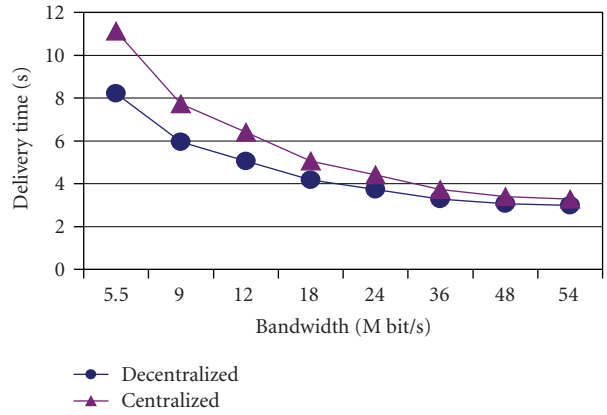

Figure 11: Delivery time versus bandwidth. (File size = 1 Mbyte, three services are considered.)

time is calculated as the sum of services execution time and the data transmission time. The data delivery time and the exchanged data size are measured with respect to the file size, the bandwidth which is considered as uniform, and the number of services involved in the content adaptation process. Figure 8 presents a relationship between the data delivery time and the number of services for the two protocols. The relationship is perceived to be linear for both protocols. As the number of services increases the performance gap between the protocols gets wide. This means, the decentralized protocols perform better with increasing the number of services in the service composition plan.

The data delivery time versus file size presented in Figure 9 and Figure 10 behaves in the same way as in Figure 8. The advantage of the decentralized service execution protocol becomes more visible when the data delivery time is analyzed with respect to the bandwidth. As illustrated in Figure 11, the performance gap between the two protocols is very significant when the bandwidth changes from 5.5 Mbits per second to 54 Mbits per second.

In addition, the analysis of the exchanged data size versus the number of services and the file size (see Figures 12 and 13) reflects similar behaviour to the data delivery time for the two protocols. Hence, the decentralized protocol performs better than the centralized one with respect to the data delivery time and the size of exchanged data.

To summarize, the decentralized execution protocol can enhance the performance of the DCAF architecture by decreasing the network load and the data delivery time and make it more scalable.

7. FAULT TOLERANCE IN DCAF ARCHITECTURE

Content adaptation process is accomplished successfully if there is no fault during services execution. However, the execution of the composite service can fail due to different causes such as network disruption and service discovery failure.

To tackle this problem, the local proxy can replace the failed service with an equivalent one. Identifying the failed service is straightforward for the centralized protocol as compared to the decentralized one since the local proxy controls the execution of each service in the centralized protocol. However, in the decentralized protocol, as discussed in

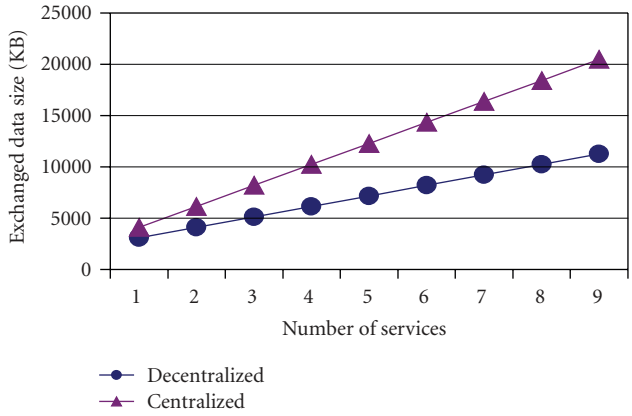

Figure 12: Exchanged data size versus number of services. (Bandwidth = 54 Mbps, file size = 1 Mbyte.)

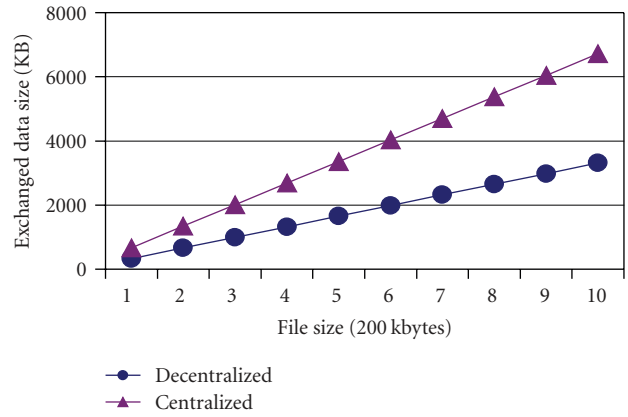

Figure 13: Exchanged data size versus file size. (Bandwidth = 54 Mbps, three services are considered.)

[26], the failed service can be identified through the use of acknowledgment messages (ACKs). When an ASP executes an adaptation service S and forwards the partially adapted data to the next ASP, it sends an ACK message to the local proxy to inform that the service S has been executed. If the local proxy does not receive an ACK message from such ASP, it concludes that the composite service execution is failed. The implementation of the fault detection and recovery mechanism within DCAF architecture is in progress.

8. CONCLUSION AND FUTURE WORK

In this paper, we have presented a multimedia adaptation graph generator (MAGG) algorithm and service composition plan execution protocols, that is, centralized and decentralized protocols. The algorithm constructs an adaptation graph which gives all service composition possibilities that satisfy the required adaptation needs. The selection of the optimal path (service composition plan) from the graph is done based on the QoS model which consists of the adaptation service execution cost, the adaptation service execution time, and the data transmission time.

The MAGG algorithm and the service execution protocols have been experimented in the DCAF architecture. The experiments on the graph construction algorithm (service composition algorithm) show that the graph construction time increases linearly as the number of adaptation tasks increases. The number of service per task, however, has not an important impact on the graph construction time. In addition, the experiments on the decentralized execution protocol show better performance than the centralized one, especially when the bandwidth is small, which is actually not surprising.

Successful content delivery does not only depend on effective execution of adaptation services but also on detection and recovery of failed adaptation services during services execution. For this purpose, we are planning to implement a fault detection and recovery mechanism within the DACH architecture in order to develop a fault tolerant content adaptation system.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments that helped them improving the paper.

REFERENCES

[1] A. Held, S. Buchholz, and A. Schill, "Modeling of context information for pervasive computing applications," in *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI '02)*, Orlando, Fla, USA, July 2002.

[2] C.-H. Chi, Y. Cao, and T. Luo, "Scalable multimedia content delivery on Internet," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '02)*, vol. 1, pp. 1–4, Lusanne, Switzerland, August 2002.

[3] M. Margaritidis and G. C. Polyzos, "Adaptation techniques for ubiquitous Internet multimedia," *Wireless Communications and Mobile Computing*, vol. 1, no. 2, pp. 141–163, 2001.

[4] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting multimedia Internet content for universal access," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 104–114, 1999.

[5] K. Marriott, B. Meyer, and L. Tardif, "Fast and efficient client-side adaptivity for SVG," in *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, pp. 496–507, ACM Press, Honolulu, Hawaii, USA, May 2002.

[6] Z. Lei and N. D. Georganas, "Context-based media adaptation in pervasive computing," in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '01)*, vol. 2, pp. 913–918, Toronto, Canada, May 2001.

[7] A. Singh, A. Trivedi, K. Ramamritham, and P. Shenoy, "PTC: proxies that transcode and cache in heterogeneous web client environments," *World Wide Web*, vol. 7, no. 1, pp. 7–28, 2004.

[8] J.-G. Kim, Y. Wang, and S.-F. Chang, "Content-adaptive utility-based video adaptation," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '03)*, vol. 3, pp. 281–284, Baltimore, Md, USA, July 2003.

[9] S. Wee and J. Apostolopoulos, "Secure scalable streaming and secure transcoding with JPEG-2000," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '03)*, vol. 1, pp. 205–208, Barcelona, Spain, September 2003.

[10] S. D. Gribble, M. Welsh, R. von Behren, et al., "The Ninja architecture for robust Internet-scale systems and services," *Computer Networks*, vol. 35, no. 4, pp. 473–497, 2001.

[11] S. Ardon, P. Gunningberg, B. Landfeldt, Y. Ismailov, M. Portmann, and A. Seneviratne, "MARCH: a distributed content adaptation architecture," *International Journal of Communication Systems*, vol. 16, no. 1, pp. 97–115, 2003.

[12] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, and H. Hellwagner, "Automatic adaptation of streaming multimedia content in a dynamic and distributed environment," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '05)*, vol. 3, pp. 716–719, Genova, Italy, September 2005.

[13] G. Berhe, L. Brunie, and J.-M. Pierson, "Content adaptation in distributed multimedia systems," *Journal of Digital Information Management*, vol. 3, no. 2, pp. 95–100, 2005.

[14] D. Jannach and K. Leopold, "Knowledge-based multimedia adaptation for ubiquitous multimedia consumption," *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 958–982, 2007.

[15] B. D. Noble, M. Price, and M. Satyanarayanan, "A programming interface for application-aware adaptation in mobile computing," in *Proceedings of the 2nd USENIX Symposium on Mobile and Location-Independent Computing (MLICS '95)*, pp. 57–66, Ann Arbor, Mich, USA, April 1995.

[16] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas, "Dynamic adaptation in an image transcoding proxy for mobile web browsing," *IEEE Personal Communications*, vol. 5, no. 6, pp. 8–17, 1998.

[17] W. Y. Lum and F. C. M. Lau, "A context-aware decision engine for content adaptation," *IEEE Pervasive Computing*, vol. 1, no. 3, pp. 41–49, 2002.

[18] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, "Using smart clients to build scalable services," in *Proceedings of the USENIX Annual Technical Conference*, pp. 105–117, Anaheim, Calif, USA, January 1997.

[19] V. Cardellini, P. S. Yu, and Y.-W. Huang, "Collaborative proxy system for distributed Web content transcoding," in *Proceedings of the 9th International ACM Conference on Information and Knowledge Management (CIKM '00)*, pp. 520–527, McLean, Va, USA, November 2000.

[20] S. Buchholz, T. Hamann, and G. Hübsch, "Comprehensive structured context profiles (CSCP): design and experiences," in *Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PerCom '04)*, pp. 43–47, Orlando, Fla, USA, March 2004.

[21] G. Berhe, *Access and adaptation of multimedia content for pervasive systems*, Ph.D. thesis, INSA de Lyon, Lyon, France, September 2006, http://docinsa.insa-lyon.fr/these/2006/girma/these.pdf.

[22] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[23] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service composition for mobile environments," *Mobile Networks & Applications*, vol. 10, no. 4, pp. 435–451, 2005.

[24] G. Berhe, L. Brunie, and J.-M. Pierson, "Service-based architectural framework of multimedia content adaptation for pervasive computing environment," in *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS '04)*, San Diego, Calif, USA, January 2004.

[25] Y. Fawaz, A. Negash, L. Brunie, and V.-M. Scuturici, "Service composition-based content adaptation for pervasive computing environment," in *Proceedings of the International Conference Wireless Applications and Computing (IADIS '07)*, Lisbon, Portugal, July 2007.

[26] Y. Fawaz, C. Bognanni, V.-M. Scuturici, and L. Brunie, "Fault tolerant content adaptation for a dynamic pervasive computing environment," in *Proceedings of the 3rd International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA '08)*, Damascus, Syria, April 2008.