Journal of Mobile Multimedia, Vol. 2, No. 2 (2006) 098–111

CONTENT ADAPTATION BASED APPROACH FOR UBIQUITOUS MULTIMEDIA

HAIPENG WANG[a]
School of Computer Science, Northwestern Polytechnical University
Xi'an, 710072 Shaanxi, P.R.China
whp_box@hotmail.com

ZHIWEN YU
Information Technology Center, Nagoya University
Nagoya, Japan
zhiwen@itc.nagoya-u.ac.jp

XINGSHE ZHOU, TAO ZHANG, DONG XIANG
Northwestern Polytechnical University
Xi'an, 710072 Shaanxi, P.R.China

Received March 1, 2006
Revised April 18, 2006

Content adaptation is playing an important role in ubiquitous multimedia, however it is a challenging work due to the high degree of dynamism and heterogeneity of the ubiquitous computing environments, where hundreds of devices provide information, and thousands of terminals access these information. Recently, some researchers proposed to address this issue by dynamically organizing services or components into customized applications at runtime. However, due to the maintenance of the dependencies between services or components, this kind of system becomes more complicated with the growth of the system scale. Programming for these systems is also error-prone. This paper discusses the issues of content adaptation based approach for ubiquitous multimedia, and presents a prototype system, called UbiCon system. By abstracting media streams into generic CONTENT entities, the system provides a simple and powerful means for services to operate media stream. The CONTENT is dynamically created by the system at runtime, and essentially has local association with related services. As a result, the CONTENT is also used as a loosely coupling method for cooperating associated services. By abstracting services with a T model, the services can effectively cooperate together with other services. As a result, a collection of sophisticated applications can be built with this simple and effective services model.

Keywords: ubiquitous multimedia, ubiquitous computing, content adaptation
Communicated by: I. Ibrahim

1 Introduction

There is an explosive growth in the number of ubiquitous multimedia sources, such as the powerful general-purpose servers providing HTTP streams or audio/video streams, the fixed sensors continuously generating traffic or weather information streams, and mobile devices

[a]P.O.Box 404, Northwestern Polytechnical Unversity, Xi'an 710072, Shaanxi, P.R.China.

reporting position. On the other hand, with the vision of ubiquitous computing became reality, people are living in an environment embodying a lot of networked computers, such as PDAs, cellular phones, and smart appliances. People wish to access their desired information anytime anywhere with their handy terminals [1]. However, with the difference in capabilities of terminals and preferences of users, it becomes more difficult and complicated to serve suitable media streams in server side [2, 3], or to provide adaptation supports in client side [27, 26], which will result in fat server and fat client. Fat client is obviously not allowed for resource-limited devices, such as PDAs, smart appliances, and laptops. Even for powerful resource-rich servers, it is also unreasonable to design such fat server with endless requirements.

Performing adaptation in intermediary [25, 24] is a promising approach, which is more suitable in ubiquitous computing environments. Several projects undertake this rationale [25, 23, 22], most of which adopt the approach of dynamically configuring components into an application to fulfill the adaptation task. These components usually are composed into components tree or components graph. With this structure, it is necessary for the system or the application to maintain the dependencies between them, which is a complicated work. Furthermore, with the growth of the system scale, it becomes more difficult to maintain these dependencies, which in turn limits system's flexibility and extensibility.

In another aspect, the dependencies between adaptive units, for example components mentioned above or services, are usually specified in terms of the interfaces provided by them. For instance, PCOM [22] specified dependencies in terms of the (Java) interfaces provided by the component and the (Java) events that the component might signal. Because the interaction between components is task-specific, the interfaces become component-specific, which means that the components tightly couple to each other by these specific or private interfaces. This characteristic of tightly coupling between adaptive units is not a good choice in ubiquitous computing systems.

To address such issues, the approach being pursued by some research groups is based on dynamic adaptation of the content to the specific terminals being used. Our approach follows this philosophy and presents the distinctive aspects described below.

- Eliminate the maintenance of dependencies. This is achieved with CONTENT entity, which is an abstraction of media stream flowing between services.

- Offer a loosely coupling mechanism between the services cooperating together to fulfill an adaptation task. This is also achieved with CONTENT abstraction, with a distinct feature of local view belong to related services (see Section 3).

- Provide a simple and powerful mechanism for composing multiple services into a sophisticated adaptation application. This is achieved with the service T model.

The rest of this paper is organized as follows. Section 2 presents our approach. Section 3 introduces the fundamental conceptual models. Section 4 describes the prototype system of UbiCon. Section 5 discusses the current status of our implementation. Section 6 discusses related work. Section 7 concludes the paper by discussing the contribution of the paper and future work.

2 Approach

The approach, presented in this paper, shares the philosophy of intermediary adaptation [25, 24]. Figure 1 shows the role of intermediary in adaptation systems. This model consists of three major components, producer and consumer, media stream, and intermediary. The producer and consumer are the origin and destination of media stream respectively. The media streams are used to abstract media contents (e.g., text stream, audio/video stream). The intermediaries are used to abstract the media operations (e.g., reducing the image resolution, reformatting HTML files).

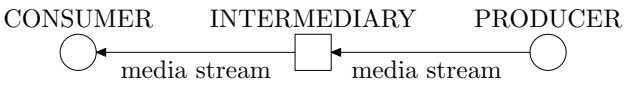

Fig. 1. The intermediary model.

More importantly, this model presents a separation between the media streams and the intermediaries. Based on this separation, the UbiCon, our prototype system, provides two major abstractions for describing media streams, and services located in intermediaries. The abstraction of media streams provides an *identical* interface for intermediaries to access these streams, through which a collection of services can operate together in a loosely coupling manner, which in turn reduces the overhead and complexity for maintaining the dependencies between these services. Another abstraction concerns the composition of services, which provides a mapping or binding from adaptation tasks or subtasks to services themselves. This mapping also simplifies the management of adaptation tasks with a set of TupleSpace-styled operations. For instance, whenever a service completes a content adaptation task delegated to it, the service *takes* this task from a TupleSpace [20, 19, 18] by itself. By using this mechanism, the UbiCon accomplishes some degree of automation of adaptation tasks management.

3 Conceptual Model

In this section, we describe several fundamental models underlying the content adaptation based approach.

3.1 Delivery Context

In the following description, the *service* refers to a combination of operations and data made available to users through the network. Services available on the network are delivered by service providers. Such services usually generate multimedia contents, e.g., text, images, audio or video streams. End users consume those services through accessing terminals. The *accessing device*, or terminal refers to a ubiquitous device having certain hardware and software characteristics. It is usually connected to a network. Accessing devices may be greatly differ in their characteristics, since they usually fulfill different requirements, either technology or usage driven.

Delivery Context refers to the set of parameters conditioning the way the user perceives information through his/her terminal. In general, terminals can be described in terms of the following characteristics:

- **Device capabilities**. They describe the accessing attributes of ubiquitous devices, which include the hardware attributes and software attributes. The hardware attributes describe the physical structure and the inner constraints of a device, e.g., the maximum screen resolution, the CPU speed, or the kind of input and output peripherals, and so on. The software attributes describe the software capabilities of the terminal as well as the programs installed on it, such as, the operating system, the availability of specific applications (e.g., a presenter in speech, or Web browser) and interpreters (e.g., Java Virtual Machine, or Macromedia Flash), and so on.

- **Network attributes**. They include parameters such as bandwidth, response time, faults management policies, or security support.

- **User preferences**. They describe the personal expectations of certain users about the way the content should be presented and, sometimes, about the content itself, e.g. preferences about the presence of presentation in visual media (text, or html), or in speech, as well as the subjects the user is most interested in (e.g., music, art, sports).

In order to be properly presented to end users, ubiquitous multimedia must be adapted according to the current delivery context. By content adaptation we manipulates the structure of a multimedia resource, such as Web pages or images, the selection of its content, or the modification of the nature or the form of the content itself and its related resources. These selection and modification activities often deal with deciding whether specific content elements should be kept or discarded and, if they need to be adapted, what kind of adaptation should be performed, e.g., to which extent to resize an image, without loosing the message it transmits. Content adaptation is carried out by software entities, called adapting services, that are able to perform actions on the content or on its subparts. The overall adaptation of a multimedia is the result of basic adapting actions performed on each single content element of the whole multimedia, according to a proper adaptation plan.

3.2 DTG: The Task Model

The Delegatable Task Graph (DTG) model describes a set of tasks and the dependencies between them. These task are derived from an adaptation application. The delegatable task means that this task can be delegated to and fulfilled by a service. The service is provided by any services providers, and registers itself using UbiCon's register system service. To make this idea concrete, suppose a Web content adaptation application, where an HTML stream flows from an HTTP server to a resource-limited PDA client. We assume that the PDA is not able to deal with the table tags and the high quality images due to the restriction of client's capabilities and bandwidth. Now we can derive at least two delegated tasks. First is the table-to-list task, which reformats the table tags to alternative list tags. Second is color-to-gray task, which reduces the color space of image. Both of these tasks can be delegated to related services registered in the UbiCon system. In this example, there are not direct dependencies between these two tasks. However, if we derive two other tasks—HTML stream producing task, and HTML viewing task—from this application, these will introduce dependencies between the former two tasks and the later two tasks.

The DTG model is defined in a graph, which is composed of edges and nodes, where the nodes represent delegated tasks, and the edges stand for dependencies between these tasks.

The DTG graph, depicted in Figure 2, is specified by $G = (V, E)$ where G is the graph, V is the set of nodes, and E is the set of edges that connect nodes.

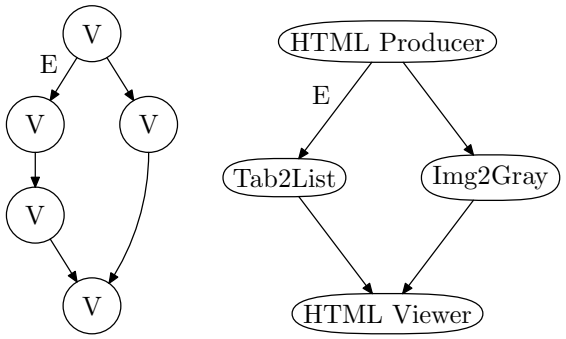

Fig. 2. The DTG Model.

UbiCon describes the DTG graph in a graph markup and modeling language of XGMML [17] with some modifications. We use four major elements to describe a DTG graph—*graph*, *node*, *edge*, and *att*. The *graph* element is the root element and it can contain *node*, *edge* and *att* elements. The *node* element describes a node of a graph, and the *edge* element describes an edge of a graph. Additional information for graphs, nodes and edges can be attached using the *att* element. An example is illustrated in Figure 3. A traversal of the nodes and edges makes a full DTG file used to specify the configuration for a collection of services operating together to fulfill an adaptation application.

```
<?xml version="1.0" encoding="UTF-8"?>
<graph directed="1" id="1">
  <node id="1" label="">
    <att name="" value=""/>
    <att name="" value=""/>
  </node>
  <node id="2" label=""/>
  <node id="3" label=""/>
  <node id="4" label=""/>

  <edge from="1" to="2"/>
  <edge from="1" to="3"/>
  <edge from="2" to="4"/>
  <edge from="3" to="4"/>
</graph>
```

Fig. 3. An example of DTG graph.

3.3 CONTENT: The Media Stream Model

The content model is used to describe the media stream flowing in UbiCon system from one service to another service, which provides a general view of a variety of media streams for services. The content model provides an abstraction of CONTENT, which has two interfaces,

outputstream, and *inputstream* as shown in Figure 4. The *outputstream* provides services with an output media stream, with which services can write data into it. The *inputstream* provides services with an input media stream, with which services can read data from it.

Fig. 4. The CONTENT Model.

Two distinct features of CONTENT are locality and dynamism. We illustrate them through an example. Suppose an HTML format transcoding case. There are two services. One is an HTML stream producer, which may be a proxy; the other is a table-to-list transcoder service (tab2list). Whenever upon receiving a request from the tab2list service, the UbiCon system creates a CONTENT entity dynamically at runtime, and returns it back to the caller service. This CONTENT entity is only valid and meaningful for both tab2list and HTML stream proxy; it is local to them, although there may exist other services in system, such as an image compressing service. In summary, this CONTENT is local to certain services, and is dynamically created at runtime. There are no constant and global CONTENT entities in the UbiCon system.

3.4 T: The Services Model

The T model describes the services running in the UbiCon system, which draws a skeleton for these services. As illustrated in Figure 5, the T model consists of three major interfaces, namely them *top* interface, *readstream* interface, and *writestream* interface. The *top* interface is responsible for the interaction with the DTG model, such as reading a delegatable task from the DTG graph at the initial phase, removing the delegatable task from the DTG graph after it is completed, and writing new extended tasks into the DTG graph. Both the *readstream* and *writestream* interfaces are responsible for the access to the CONTENT entity of the content model, such as reading table parts from an HTML stream through a CONTENT entity, and writing new or reformatted content into an HTML stream through a CONTENT entity. From the architecture point of view, the T model is the glue or the bridge between the DTG model and the content model.

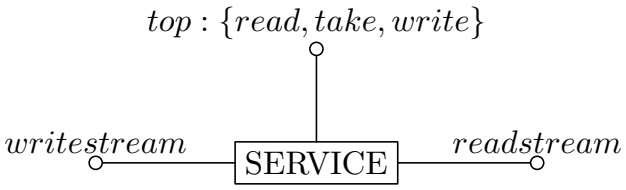

Fig. 5. The T Model.

The T model provides a simple and powerful mechanism for building a collection of sophisticated adaptation applications. This simplicity is important for developing large systems, including our UbiCon system.

3.5 Putting It All Together

Figure 6 shows that how these three models work together to fulfill a common adaptation task. The T model interacts with the DTG model through its *top* interface. For example, an instance of the T model may read, take, or write a task from the DTG. The *writestream* and *readstream*, interfaces of the T model, bind to the *outputstream* and *inputstream*, interfaces of the CONTENT model, respectively. The ubiquitous multimedia flows into or out the CONTENT entity through the *outputstream* or the *inputstream* interfaces, and accepts appropriate adaptation at various T service entities. Different T services make virtual binding through the CONTENT entity between them.

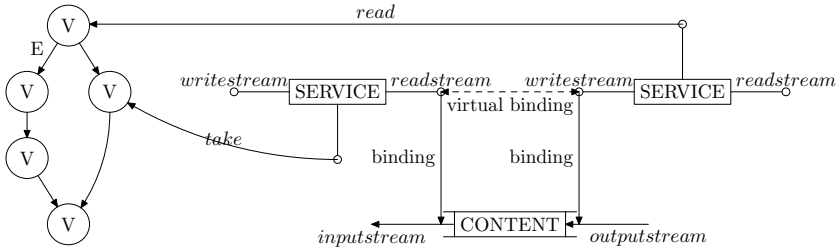

Fig. 6. Three Models Working Together.

4 UbiCon: The Prototype System

The UbiCon system is a flexible system, which adopts the content adaptation based approach to providing a ubiquitous multimedia environment. This system includes three major components, DTG, CONTENT, and session manager. The DTG component manages the adaptation tasks and the dependencies between them. The CONTENT component provides an abstraction of media stream, called CONTENT entity, and a set of operations on it. With these operations, the CONTENT provides a kind of cooperating mechanism for services. The session manager is responsible for local and transient managing. Figure 7 depicts the major functional components of the UbiCon system.

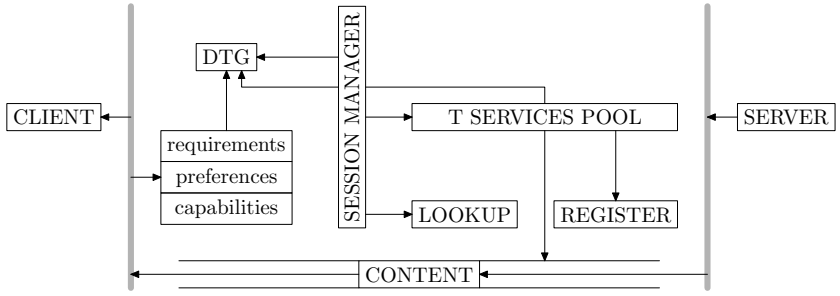

Fig. 7. The UbiCon Architecture.

4.1 The DTG

The DTG has two major functions, tasks and dependencies representation and tasks management. The DTG achieves them using an XGMML-based task representation and a TupleSpace-based management.

For tasks and dependencies representation, the DTG is realized as a data repository, which is made up of a collection of delegatable tasks. These tasks are derived from user's requirements based on user's preferences and client's capabilities. The DTG describes these tasks in an XGMML language, and arrage them into a graph structure, in which nodes and edges are the tasks and dependencies between nodes respectively. The tasks are specified in terms of interfaces, by which these tasks would be delegated to associated services implemented that interfaces. Figure 8 shows an example of DTG.

```
<?xml version="1.0" encoding="UTF-8"?>
<graph directed="1" id="1">
  <node id="1" label="HTMLProducer">
    <att name="interface"
      value="ubicon.HTMLProducerProxy"/>
    <att name="outputStream"
      value="ubicon.Content.HTMLStream"/>
  </node>
  <node id="2" label="Tab2List">
    <att name="interface"
      value="ubicon.Tab2List"/>
    <att name="inputStream"
      value="ubicon.Content.HTMLStream"/>
    <att name="outputStream"
      value="ubicon.Content.HTMLStream"/>
  </node>
  <node id="3" label="Img2Gray">
    <att name="interface"
      value="ubicon.Img2Gray"/>
    ...
  </node>
  <node id="4" label="HTMLViewer">
    <att name="interface"
      value="ubicon.HTMLViewerProxy"/>
    ...
  </node>

  <edge from="1" to="2" weight="0"/>
  <edge from="1" to="3" weight="1"/>
  <edge from="2" to="4"/>
  <edge from="3" to="4"/>
</graph>
```

Fig. 8. An example of DTG.

For the tasks management, the UbiCon realizes the DTG repository as a TupleSpace. With

such design, the session manager (subsection 4.3) and services are capable of performing a set of TupleSpace-styled read, take, and write operations on tasks. These TupleSpace-styled operations are very useful for the management of the DTG repository, since it does not demand a centralized manager all time. As a result, the UbiCon system produces better performance.

4.2 The CONTENT

The CONTENT component essentially provides a cooperating mechanism, which is a data-based mechanism in the UbiCon system. A collection of services work together by operating on an media stream. The operations can be divided into two categories. The first kind is access operations, including get an inputstream and get an outputstream from a CONTENT entity, and the second one is task-specific operations, such as format transcoding, image resizing, and text-to-speech translation. The first kind of operations are provided by the system—the CONTENT entity, and the second kind of operations are implemented by services themselves, which are out of control of the system.

The UbiCon realizes CONTENT using a TupleSpace. Upon receiving a request from the caller services, the CONTENT entity will provide them with an media stream, inputstream or outputstream, only if the system can create one at this moment. For a text-to-speech example, if and only if the text-to-speech service writes an outputstream into system's CONTENT TupleSpace, the speech player can only get a speech stream. Otherwise, the caller services would be blocked on this CONTENT entity, until a valid CONTENT entity is created successfully, or becomes active.

For services blocked on a CONTENT entity, there will raise a contention or conflict for this CONTENT entity resource. For example, both blocked services, HTML-to-text and table-to-list, will be notified, when a valid CONTENT entity appears. The UbiCon system resolves this conflict by checking the DTG with the weight attributes associated with the edge elements. The task and its corresponding services owing high weight would win the race.

4.3 The Session Manager

The session manager component plays the role of local manager. It initializes a session, and provides local ID management, such as service ID, and CONTENT ID. The main purpose of local ID is to improve the performance of the UbiCon system. Another responsibility of session manager is to monitor the availability of services. When a service becomes unavailable, the session manager will perform a lookup procedure to find another substitute. The detecting of an unavailable service is realized using the soft-state mechanism, in which a heart-beat message sent by the runtime system.

4.4 Other System Components

The UbiCon system also contains a few system components and system services. We describe them briefly as follows.

- The T Services Pool. This pool contains a collection of services running in the system. These services are composed into an adaptation application by using the T model. Due to the supports provided by both the CONTENT model and the T model, these services are cooperating together in a loosely coupling manner.

- Lookup Service. This system service provides services lookup function by using a template-based matching mechanism. The template consists of interface, attributes, and service ID.

- Register Service. This system service provides services registering function. The registering information contains services' interface and additional optional attributes used to provide further detailed information.

4.5 The Programming Model

Briefly, we present the programming model concerning with CONTENT entity and DTG. The Figure 9 shows the programming model of the DTG, and the Figure 10 shows the programming model of the CONTENT entity.

```
public Interface Template {
    public ID id;
    public java.lang.Class[] types;
    public Entry[] attributeSetTemplates;

    Template(
        ID id,
    java.lang.Class[] types,
    Entry[] attrSetTemplates);
}

public ServiceTemplate implements Template;

getDTG(int id).read(ServiceTemplate tmpl);
getDTG(int id).take(ServiceTemplate tmpl);
getDTG(int id).write(ServiceTemplate tmpl);
```

Fig. 9. The Programming Model: DTG.

```
public ContentTemplate implements Template;
Content content;
content = getContent(ContentTemplate tmpl);
content.getInputStream();
content.getOutputStream();
```

Fig. 10. The Programming Model: CONTENT.

5 Implementation

Currently, two preliminary adaptation applications are conducted. One application is the Information browsing with PDA and large screen displayer (PDP: plasma display 61"), another is the presenting of HTML files using the transcoding from text to speech in special environments, where visual presenting is not allowed, such as car driving environment.

Our prototype is based on Jini V2.0 [15, 16], JavaSpaces V2.0 [14, 13], and J2SE V1.4. We use the JavaSpaces to construct and store both the DTG graph and CONTENT entities,

and use the Jini to build the system services of lookup and registration. We build a simple parser program which parses the DTG files and puts them into the DTG JavaSpaces.

The application of information browsing with PDA and PDP is used to verify the basic functions of the UbiCon system. The information, including environmental light and volume, object location, and background map, is captured by a sensor network from environment, which may be a battlefield or a district of city. Users may access these information with their appropriate ubiquitous devices, for instance a soldier with a handy PDA, or an army leader with the PDP in control center. Due to the differences between PDA and PDP in their capabilities, the content adaptation need to be carried out according to accessing terminals. The raw information, from the sensor network, is coded into XML, and then two services, xml2txt and xml2rtf, are used to perform content adaptation. Both of the services are implemented as Jini services. The xml2txt service transcodes the XML format into text format, which is appropriate to presented in the soldier's PDA (Figure 11). The xml2rtf service as many information as possible (location, light, and volume), and also the background map into the final result, which is presented in the army leader's PDP (Figure 12).

Fig. 11. PDA Screen Shot.

In this application, we deployed two services in two PCs. An IBM x-series-235 server runs a session manager and the system services of lookup service and register service. Both JavaSpaces-based DTG and CONTENT entities are also deployed in this server. After the xml2txt and xml2rtf services startup, they registered them in the system's register service. The session manager begins to work by checking the information stored in DTG, and then it lookups the relevant services, xml2txt and xml2rtf, and notifies them to work together to perform content adaptation. Both of these services get their valid CONTENT entities or block on them, until they complete successfully. After complete their works, the services take their tasks from DTG JavaSpaces without the participating of session manager. Finally, the session manager does some clear up work.

The application of presenting HTML with speech is used to verify the ability of the UbiCon system to do dynamic content adaptation. This application is assumed to perform in special situations, such as in driving a car, where visual Web browsing is dangerous. By inserting these information into DTG JavaSpaces at run time, the session manager can enable

Fig. 12. PDP Screen Shot.

a new adaptation task dynamically. A text-to-speech service is started and begins to perform work, which is similar to xml2txt or xml2rtf. The text-to-speech service is built using the FreeTTS [12] with some modifications. As the first step of our research, the changes of adaptation context, such as the changes of environments, are input into the system in an interactive manner. We are in the progress to automate it.

The initial experience shows the effectiveness of the UbiCon system. However, it is also clear that more adaptation services are needed to be built, in order to provide sophisticated adaptation applications.

6 Related Works

A significant effort has been spent in the last few years on providing content adaptation, and many different solutions have been proposed. Odyssey [11, 10] is an early work in the adaptation field. In Odyssey, clients can guide applications in changing their behavior so that they use less of a scarce resource. This change usually reduces the user-perceived quality, or fidelity.

Puppeteer [9] is a system for adapting component-based applications in mobile environments, which implements adaptation by using the exposed APIs of component-based applications, enabling application-specific adaptation policies without requiring modifications to the application. With the restriction of component-based approaches by nature, Puppeteer is limited its usage to component-based applications with exposed APIs. It is also difficult to program with Puppeteer, because for adapting a new application, Puppeteer needs to implement associated drivers, policies, and transcoders, all of which are non-trivial.

PCOM [22] is a light-weight component system supporting strategy-based adaptation in spontaneous networked ubiquitous computing environments. It offers application programmers a high-level programming abstraction which captures the dependencies between components using contracts, by which PCOM supports automatic adaptation in cases where the execution environment changes to the better or to the worse.

The @Terminals [23] is an extensible infrastructure for universal access to Web contents

and services, which supports both dynamic adaptation of contents and services based on terminals capabilities. For content adaptation @Terminals is able to manipulate the structure of a Web resource, select its content, or modify the nature or the form of the content itself and its related resources. For service adaptation @Terminals supports the modification the services performed by the service providers.

The CAP [21] is a modularised and scalable architecture that provides content adaptation for both Web-related data and arbitrarily complex data. The architecture can be used as part of many client-server applications, not just Web browsers.

The Espial Escape Web browser [27] is a device-specific Web browser that is executed directly on the user terminal and carries out basic resources adaptations, such as rescaling of images or shrinking of documents. A similar approach can also be found in the Device Mosaic Web browser [26] by OpenTV.

7 Conclusions

This paper presents a content adaptation based approach to ubiquitous multimedia, and a prototype system, called UbiCon. Two distinct features of the UbiCon system are adaptation tasks management and dependencies-maintaining-free mechanism. They are implemented by abstracting media streams into generic CONTENT entity, and by abstracting services with the T model. Based on these two models, the UbiCon provides a simple and powerful means for services to operate media stream, and the services can effectively cooperate together with other services. The prototype implementation and two experiments verified the effectiveness of the UbiCon system. As a result, a collection of sophisticated adaptation applications can be built with the UbiCon system.

Future work can move towards a further extension of the prototype system for supporting the management of quality of service. Another area of interest can be the adaptation and composition of Web services, for the dynamical provision of high value services. Finally, higher quality adaptation could be achieved leveraging on emerging technologies and standards for content description, such as the semantic Web and ontologies [6, 5, 4].

References

1. Z. W. Yu and D. Q. Zhang (2006), *Middleware Support for Context-Aware Ubiquitous Multimedia Services*, chapter 32 in the book of Handbook of Research on Mobile Multimedia, edited by I. K. Ibrahim, Idea Group Inc., pp. 476-490.
2. V. Korolev and A. Joshi (2001), *An End End Approach to Wireless Web Access*, Proceedings of the International Workshop on Wireless Networks and Mobile Computing.
3. D. Q. Zhang and C. Y. Chin and M. Gurusamy (2005), *Supporting context-aware mobile service adaptation with scalable context discovery platform*, Proceedings of the IEEE 61st Vehicular Technology Conference (VTC2005-Spring), pp. 2859-2863.
4. M. C. Daconta, L. J. Obrst and K. T. Smith (2003), *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, Wiley Publishing, Inc.
5. G. Antoniou and F. V. Harmelen (2004), *A Semantic Web Primer*, MIT Press, Cambridge, Massachusetts.
6. U. Srinivasann and S. Nepal (2005), *Managing Multimedia Semantics*, IRM Press, USA, chapter 1, pp. 1-30.
7. C. Becker and G. Schiele (2003), *Middleware and Application Adaptation Requirements and their Support in Pervasive Computing*, Proceedings of the 3rd International Workshop on Distributed Auto-adaptive and Reconfigurable Systems at ICDCS, USA, pp. 98-103.

8. C. Becker, G. Schiele, et al. (2003), *BASE — A Micro-broker-based Middleware For Pervasive Computing*, Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communication (PerCom 2003), USA, pp. 443-451.
9. E. D. Lara, D. S. Wallach and W. Zwaenepoel (2001), *Puppeteer: Component-Based Adaptation for Mobile Computing*, Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS), California.
10. J. Flinn and M. Satyanarayanan (1999), *Energy-Aware Adaptation for Mobile Applications*, Proceedings of the 17th ACM symposium on Operating systems principles, USA, pp. 48-63.
11. B. D. Noble, M. Satyanarayanan, et al. (1997), *Agile Application-Aware Adaptation for Mobility*, Proceedings of the sixteenth ACM symposium on Operating systems principles, France, pp. 276-287.
12. *FreeTTS*, http://freetts.sourceforge.net.
13. P. Bishop and N. Warren (2002), *JavaSpaces in Practice*, Addison Wesley.
14. E. Freeman, S. Hupfer and K. Arnold (1999), *JavaSpaces: Principles, Patterns, and Practice*, Addison Wesley.
15. Jini (1999), http://www.jini.org/.
16. Sun Microsystems, Inc. (2003), *Jini Specifications v2.0*.
17. *XGMML (eXtensible Graph Markup and Modeling Language) XGMML 1.0 Draft Specification*, 2001.
18. B. Johanson and A. Fox (2004), *Extending tuplespaces for coordination in interactive workspaces*, Journal of Systems and Software, Special issue: Ubiquitous computing, Vol.69, No.3, pp. 243-266.
19. S. P. Wade (1999), *An Investigation into the Use of the Tuple Space Paradigm in Mobile Computing Environments*, Master Thesis, Lancaster University.
20. D. Gelernter (1985), *Generative Communication in Linda*, ACM Transactions on Programming Languages and Systems (TOPLAS), Vol.7, No.1, pp. 80-112.
21. T. Phan, G. Zorpas and R. Bagrodia (2002), *An Extensible and Scalable Content Adaptation Pipeline Architecture to Support Heterogeneous Clients*, Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), pp. 507-516.
22. C. Becker, M. Handte, G. Schiele and K. Rothermel (2004), *PCOM — A Component System for Pervasive Computing*, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2004), pp. 67-76.
23. E. D. Nitto, G. Sassaroli1 and M. Zuccalà (2003), *Adaptation of Web Contents and Services to Terminals Capabilities: the @Terminals Approach*, Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communication (PerCom 2003), pp. 433-440.
24. P. Maglio and R. Barrett (2000), *Intermediaries Personalize Information Streams*, Communications of the ACM, Vol.43, No.8, pp. 96-101.
25. R. Barrett and P. Maglio (1999), *Intermediaries: An Approach to Manipulating Information Streams*, IBM Systems Journal, Vol.38, pp. 629-641.
26. *Device Mosaic Web Browser*, http://www.opentv.com/dm.
27. *Espial Escape Web Browser*, http://www.espial.com.
28. Anupam Joshi (2000), *On Proxy Agents, Mobility, and Web Access*, Mobile Networks and Applications, Vol.5, No.4, pp. 233-241.
29. R. Mohan, J. R. Smith and Chung-Sheng Li (1999), *Adapting Multimedia Internet Content for Universal Access*, IEEE Transactions on Multimedia, Vol.1, No.1, pp. 104-114.