Doctor Thesis

A Study On
Efficient and Trustful Dynamic Content Sharing System

Yuuichi Teranishi

Dec. 2003

内容梗概

近年，広帯域なネットワーク環境の普及や，モバイル環境の整備に伴ない，映像，音声等を含めたコンテンツの利用環境が多様化し，様々な形態で利用されるようになってきている．しかし，高品質なコンテンツの作成には，膨大な労力や高価な機器が必要となるため，作業コストを低減しつつ，様々な要求や環境に適応できるコンテンツ流通システムの実現が求められている．

本研究では，複数の状況や用途でコンテンツを再利用でき，作成コストを削減可能なコンテンツ流通システムの実現を目指してきた．特に，多様な利用者の要求や，環境の変化に柔軟に追従できるよう，流通の過程で動的に構造を変化させることが可能な「動的コンテンツ流通」の実現に焦点をあてた．動的コンテンツ流通では，流通の過程で変化するコンテンツの構造を分かり易く記述し，かつ，効率的に処理・転送することが課題となる．また，流通範囲やコンテンツ自体の構造が，流通の過程で利用者の要求や環境に応じて変化するため，コンテンツ提供者の意図を越えた不正な配布や加工をいかに防ぎ，安全性を保つかも重要な課題である．

本論文では，まず，典型的な動的コンテンツ流通のアプリケーションである，モバイル環境を想定した位置依存情報システムの実現法を提案する．位置依存情報システムでは，利用者の位置や状況の変化に応じてコンテンツの組み合わせを変化できることが要求される．提案手法では，場所に応じたコンテンツをオブジェクトとして扱い，それらを階層化された「スコープ」に登録することで，分かり易く位置等に応じて変化する構造を記述可能である．また，固定ネットワーク上にオブジェクト再構成機能を持つアーキテクチャを取ることで，CPU 処理能力及びネットワーク帯域が小さいモバイル環境においても，効率的な処理・転送を可能としている．提案手法はプロトタイプシステム'LODIS(LOcation Dependent Information System)'に実装して評価を行ない，有効性を確認している．

また，本論文では，動的コンテンツ流通において，コンテンツの不正な配布や加工を防止するための手法を提案する．提案手法は，保存されたコンテンツの構造に基づいて，利用手続きの制約を評価することで，従来の手法では扱えなかった，加工に関する制約管理を可能としている．また，ルールとして複数の同じ状況下における制約定義が容易に行なえ，かつ，メタ情報と構造情報のみを用いて制約評価を効率的に実行できるため，動的コンテンツ流通に適している．提案手法は，新たに定義した利用制約記述言語 'AMF(ASIA Metadata Format)'，及び，新規開発したソフトウェアエージェントを用いてプロトタイプシステム 'ASIA' に実装し，有効性を確認している．

Abstract

Increasing demands for providing high quality multimedia contents saving the costs have neccesiated the development of content sharing systems. In the content sharing system, content creators can share the content materials such as musics, videos and softwares for different purposes. In other words, creators can make multiple derived contents from one original content. Moreover, to treat up-to-date, flexible, and personalized service, the derived content should be able to change dynamically according to the user status such as access time, access location and system environment in the content sharing system. In such *dynamic content sharing* environment, it is difficult to keep the system efficient and trustful since the content structure changes dynamically.

In this paper, we made two proposals to realize the efficient and trustful dynamic content sharing.

The first proposal involves to realize the simple description and efficient processing of the dynamic content sharing. We have focused on one typical and significant application called 'location-dependent information service'. To treat frequently changing content structure, we proposed a new content management model and designed new content description language named 'MODS (Mobile Object Description Script)' according to the model. In MODS, users can define dynamic content by binding content objects to the hierarchical 'scope' which defines the various status of the user. We also proposed an architecture which locates the object recomposing mechanism on the fixed network. The architecture reduces the load of network transmission and the amount of CPU power which enables efficient processing of the MODS even in the mobile environment.

The second proposal is a new restriction management mechanism to realize the trustfulness in the dynamic content sharing. We proposed a new restriction definition language named 'AMF (ASIA Metadata Format)' based on RDF, a standard meta-data format. The advantage of the AMF is that it can define derived content

restrictions by specifying conditions for the operation methods. We also proposed a new implementation model of restriction management system suitable for AMF. In our implementation model, stored operation methods for a derived content are evaluated dynamically on the software agents. By this mechanism, contents are protected from illegal reuses by dishonest users in the dynamic content sharing.

To evaluate proposals above, we have implemented basic mechanisms on the prototype system. Behavior appropriate for the dynamic content sharing service has been confirmed and the effectiveness of our proposal has been proved by the applications.

Related Papers

Journal paper

1. 寺西 裕一, 豊城 かおり, 奥田 剛, 下條 真司, 宮原 秀夫, "ASIA: 派生コンテンツの利用制約管理が可能な情報提供システム", 電子情報通信学会論文誌, Vol.J86-B No.8,pp.1463-1475, August 2003

2. 寺西　裕一, 種茂 文之, 梅本　佳宏, 寺中　勝美, "移動体計算機環境におけるオブジェクトの動的再構成", 情報処理学会論文誌, 第39巻 第4号, pp.1077-1087, April 1998

Proceedings of international conferences

1. Yuuichi Teranishi, Fumiyuki Tanemo and Yoshihiro Umemoto, "Dynamic Object Recomposition for Active Information System on Mobile Environment", International Database Engineering and Applications Symposium '97, pp.220-227, August, 1997

Talks

1. 寺西 裕一, 長谷川 知洋, 梅本 佳宏, 佐藤 哲司, "マルチメディアコンテンツ流通における利用制約管理機構", マルチメディア, 分散, 協調とモーバイルシンポジウム論文集, pp.213-218, 1999

2. 寺西 裕一, 長谷川 知洋, 梅本 佳宏, 佐藤 哲司, "利用制約に基づくマルチメディアコンテンツ流通システムの設計", 情報処理学会研究報告 99-DPS-95, pp.31-36, 1999

3. 梅本 佳宏, 寺西 裕一, 長谷川 知洋, 佐藤 哲司, “流通管理機構を持つ複合コンテンツの管理方式”, 信学技報 (データ工学) DE99-42, Vol.99, No.202, pp.69-74, 1999

4. 山本 太郎, 寺西 裕一, 梅本 佳宏, “医療情報システムにおける情報開示制御方式”, 情報処理学会研究報告 2000-DPS-100, pp.19-24, 2000

5. 豊城かおり, 寺西 裕一, 奥田剛, 下條真司, 宮原秀夫, “コンテンツの編集を考慮した権利管理機構の提案と実現”, 情報処理学会研究報告, 2002-DPS-106, pp. 241-246, Febrary, 2002

6. 片上 修一, 奥田 剛, 寺西 裕一, 下條真司, 宮原秀夫, “遠隔会議における映像音声機器遠隔制御の動的アクセス制御方式の提案”, 情報処理学会研究報告, 2002-DPS-106, pp. 187-192, Febrary, 2002

7. 寺西 裕一, 種茂 文之, 梅本 佳宏, 寺中 勝美, “移動体計算機環境におけるオブジェクトの動的再構成”, マルチメディア通信と分散処理ワークショップ論文集, pp.173-178, Octover, 1996

8. 片上 修一, 寺西 裕一, 奥田 剛, 下條 真司, 宮原 秀夫, “コンテンツと宣伝情報との連携を実現する利用制約管理方式の提案”, 情報処理学会研究報告 2002-DPS-110, pp. 43-48, November 2002,

Contents

1 Introduction **11**

2 The Dynamic Content Sharing **14**
2.1 The content sharing processes 14
2.2 The content sharing model 15
2.3 System Requirements 16

3 Location-dependent Information System **18**
3.1 The Location-dependent Information Service 18
3.2 System Requirements for Location-dependent information service . . 20
3.3 The Mobile Object Model 20
3.3.1 The Dynamic Document Model 21
3.3.2 Basic Concepts of the Mobile Object Model 22
3.3.3 Connectivity Scope 23
3.3.4 Hierarchical Scope 24
3.4 Architecture 25
3.4.1 Required Modules 25
3.4.2 Location of Modules 26
3.4.3 Object Transfer Protocol 28
3.5 Implementation of the LODIS 29
3.5.1 MODS: Mobile Object Description Script 29
3.5.2 Detailed Architecture of LODIS 32
3.5.3 Example Application 33

4 Content Restriction Management System **38**
4.1 Restriction management model of 'ASIA' 38
4.1.1 Current restriction management model 39

4.1.2 The restriction management model in ASIA 41
4.1.3 An example of the derived content evaluation 44
4.2 The content sharing system ASIA 45
4.2.1 AMF: Restriction description language 45
4.2.2 Restriction management using agents 49
4.3 The prototype system and its evaluation 50
4.3.1 An example keyword searching service 52
4.3.2 Evaluation of the prototype system 55
4.4 Considerations 56
4.4.1 Restriction definitions for content modifications 56
4.4.2 Inheritance of the restrictions 56
4.4.3 Efficient evaluations 57
4.4.4 Security considerations 57

5 Concluding Remarks **59**

List of Figures

2.1 The Content Distribution Processes 14
2.2 static and dynamic content sharing 16

3.1 Location-dependent Information Service 19
3.2 The Dynamic Document Model 21
3.3 Mobile Object Model 22
3.4 Recomposition according to the scope 23
3.5 Hierarchical Scope 24
3.6 Required Modules 26
3.7 ObjectRecomposer is located in the mobile client 27
3.8 ObjectRecomposer is located in the fixed network 27
3.9 Differential Object Transfer Protocol 29
3.11 Scope hierarchy 30
3.10 An example description of MODS 31
3.12 LODIS architecture 32
3.13 User position generator 33
3.14 Experimental system 34
3.15 Transmitted data per hour 34
3.16 Object recomposition time 35
3.17 Example of the application 36
3.18 Detailed information for an attraction 36

4.1 User environment model in ASIA 42
4.2 Content restriction model in ASIA 43
4.3 The structure of derived content 43
4.4 An example of the derived content evaluation 45
4.5 RDF model for AMF 47

4.6 An example of AMF 48
4.7 Restriction management by agents 50
4.8 Display with 'template-A' 54
4.9 Display with 'template-B' 54
4.10 processing time for 'template-A' 55

List of Tables

3.1 Parameters available for scope definition 37
3.2 MobileObject methods (partial list) 37

4.1 Basic operations in the AMF 1.0 46

Chapter 1

Introduction

Multimedia contents like movies, musics, softwares have been created to fit for the each channel of the distribution, e.g. film theater playing, TV broadcasting, package productions. But recent digitalization of the contents and widespread use of broadband network made variety of alternative way of distribution like on-demand content distributions, broadcasting over Internet, personal content distributions, and so on. But it is very hard to create high quality content materials like movies and musics, since it requires much efforts, artistic senses, and expensive equipments. Hence the requirement for reusing outstanding high quality content in the multiple distribution channel, in other words, the realization of the objective "One Source, Multi Use", is increasing. In such content reusing environment, multiple derived contents are created for multiple different purposes from one original content. Namely, the original content is 'shared' by the derived contents in the content distribution. For such demands, many standards have been developed to realize the content sharing. For example, W3C consortium have developed a specification to define the structure of the distributed multimedia contents, called 'SMIL[1]'. TV-Anytime Forum[2] have been developing specifications to share audio-visual contents in the network environment and treat them in the electronic equipments. MPEG-21[3] have been developing specifications of the infrastructure for the delivery and consumption of the multimedia content.

In the present content delivery service, content providers treats mainly 'static content sharing'. It means the derived contents have fixed structure and do not change during the distribution. The original content materials are embedded in the derived content. But such static content sharing is not enough for the current

content delivery network, in which enormous number of contents are created day by day and large number of users who have different requirements and environments exist. In such environment, to treat up-to-date information and to adapt to the user requirements, the structure of the derived content is required to be able to change dynamically according to the situations or the requirements. We call the content sharing with dynamic structure change as 'dynamic content sharing'.

The dynamic content sharing is suitable to provide up-to-date, flexible, and personalized content service. An example service of dynamic content sharing is 'news on demand', which changes the content day by day according to the provided news and user requirement. Another example application is 'location-dependent information system', which displays location-dependent contents on the mobile clients such as Handheld PC, PDA and mobile phone, according to the user's current position. The latter application requires more dynamic changes of the content structures.

In spite of the advantages, there are technical challenges to realize the dynamic content sharing as a practical service. Our goal in this research is to establish the basic technology to realize the efficient and trustful dynamic content sharing system to suit for the practical service. In particular, we have focused on the following two problems.

The first problem is how to realize the simple description and efficient processing of the dynamic content sharing. To develop a content service with dynamic content sharing, it is required to describe the derived content structure which follows to the user situations or requirements. It is difficult to describe such dynamic content in the current content distribution framework. Moreover, architecture for effective transfer and processing is required for dynamic modifications or compositions.

The second problem is how to keep the reuse policy in the dynamic content sharing. While reusing the contents, users must follow the restrictions which are given by the original content creators, to keep the meaning of the content and the rights of the creators. Many digital rights management (DRM) system have been developed to keep the usage limitations of the contents. But currently, no rights management framework is suitable to manage the restrictions for dynamic content sharing.

Firstly in this research, we focused on the 'location-dependent information service', as a typical dynamic content sharing. It requires a dynamic sharing of the location-dependent contents and efficient transferring of data in the mobile environment. We have developed a prototype location-dependent information system

called "LODIS". The detailed design and implementation of LODIS is described in this paper. We proposed a new content management model and designed content description language called 'MODS', according to the model. In 'MODS', service providers can define dynamic content sharing services by binding content objects to the hierarchical 'scopes', which define the various status of the user. We also proposed an architecture which locates the object recomposing mechanism on the fixed network. The architecture reduces the load of network transmission and the amount of CPU power used by the mobile client.

Secondly, we have developed an information delivery system called "ASIA", which can manage the restrictions of the dynamic modifications or compositions. To manage rights in the dynamic content sharing, we proposed a new restriction definition language 'AMF' based on the RDF, a standard meta-data format. AMF can define the derived content restrictions by specifying the rules for the operation methods. We also proposed a new mechanism to manage the AMF restrictions by applying the stored operation methods dynamically to the contents on the software agents.

By developing the prototype systems, we have confirmed the feasibility of our models. Some applications which are developed based on the models show the effectiveness of our proposals.

In the chapter 2, the content sharing model we treat here is described. In the chapter 3, the design and implementation of the location-dependent information system we have developed is described. In the chapter 4, our design of the restriction management system and its prototype is described.

Chapter 2

The Dynamic Content Sharing

2.1 The content sharing processes

The digital content sharing system we treat here is shown in figure 2.1. As shown in the figure, typical content distribution process consists of five processes: create, store, publish, retrieve and use (reuse).

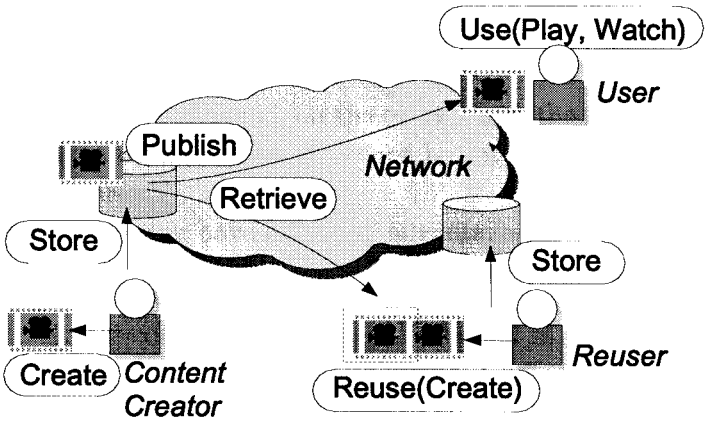

Figure 2.1: The Content Distribution Processes

Firstly, the content creator creates a content. It means creating a digital data which can be stored as the electronic media. For example, making data with format like MPEG, Motion-JPEG, AVI, WAV, MP3, etc. Converting the analog data into

the digital data, e.g. using an image scanner to scan the photograph, is also one of the content creation.

Next, created data is stored to the data storage systems such as file systems and databases. It requires indexing or binding meta-data for the effective retrieval. Then, the content is published to the network with unique identifier (e.g. URI). Published contents are retrieved by users using the identifier. The identifier of the content is obtained via link or as a search result. Finally, Content data is transferred to the user's client and used (displayed, played, executed, and so on). If 'reuse' of content is occurred, the reuser creates a derived content from the original content, modifying or composing with other content. In this case the same distribution processes are repeated.

2.2 The content sharing model

The content sharing model can be divided into two models, the 'static content sharing' and the 'dynamic content sharing'.

In the 'static content sharing', the structure of the content, i.e. the original contents used by derived content, is decided statically and does not change in the distribution processes. In other words, the original content is statically bound in the derived content.

In the 'dynamic content sharing', on the other hand, the structure of the derived content changes dynamically. The structure is not decided until the content is provided to the each user.

Dynamic content sharing have following advantages.

- Providing up-to-date information service
  Since the original content is not statically bound in the derived content, it is easy to provide up-to-date information service for frequently updated contents, like news content, serialized novel, and so on.

- Adaption to the user requirements
  Since the content structure can be changed dynamically, it can provide flexible content service which can adapt to the user's requirements or environments. Multiple kind of personalized content can be provided without preparing multiple candidate of the contents.

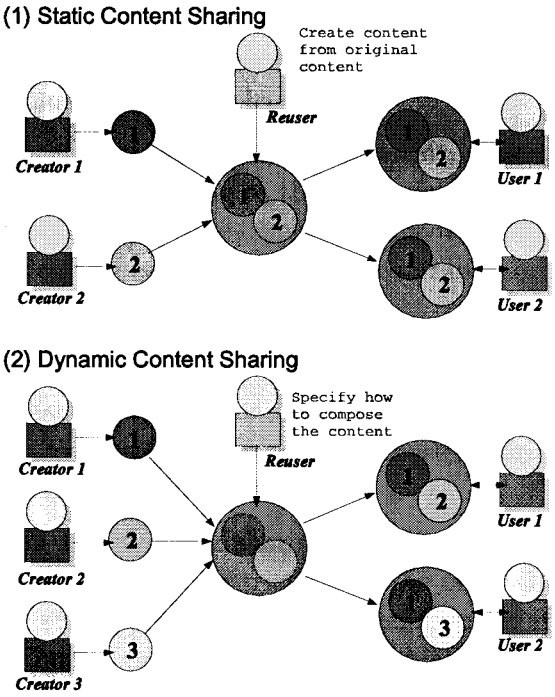

Figure 2.2: static and dynamic content sharing

2.3 System Requirements

To realize the dynamic content sharing, following features are required.

1. Reusable content creation
   During the 'create' and 'store' process in the content distribution, it is required that each content is created as a reusable unit and attributes or indices are attached to them. For example, it is useful for recomposition if each video content is created scene by scene and the meaning like 'interview of a certain person' or the attributes like 'category of the comedy' are appended as index or meta-data.

2. Simple description and efficient processing
   In the 'reuse' process in the content distribution, the reuser must describe the content structure that changes according to the user situations or require-

ments. It is difficult for content creators or reusers to grasp such dynamic structures. Hence a simple content description framework is required for the efficient development of the services.

The content data provided by the independent content creators have to be collected or modified according to the dynamically changing structure. However, it should not affect on the displaying, playing and executing the content. Hence the efficient processing architecture suitable for the dynamic content sharing is required.

3. Security and trustfulness
   To keep the meanings of the contents or the rights of the creators, it is required to give a restriction to the range of 'reuse'. During the content sharing processes, users must satisfy the restrictions.

   However, during the dynamic sharing process, obviously it is impossible to check manually whether the restrictions of the original content is satisfied or not. Therefore an automatic restriction management framework that covers the whole distribution processes is required. Also, the restriction management system should have an efficient architecture to evaluate the restrictions.

In our research so far, we mainly focused on the requirement 'reusable content creation'. We have been researching on it through the development of a "News on-demand system". We proposed a semi-automatic structuralization mechanism for the TV-news program with videos, sounds, narration texts, using existing scene detection algorithms. The 'scenario database system' is developed for it[4]. We realized some applications like 'Digest News Compositions', 'Related News Navigations' using the scenario database[5].

In this research, we mainly focused on the requirement '2. Simple description and effective processing' and '3. Security and trustfulness'. To treat the requirement 2, we focused on a typical application called 'location-dependent information system'. To treat the requirement 3, we designed a restriction management system with dynamic content sharing.

Chapter 3

Location-dependent Information System

Recent technological advances in portable computers and wireless communications have made it possible for mobile users to access a variety of network resources. The resulting computing environment, the so-called *mobile environment*, has been the backdrop for a number of research and development projects.

Our research has focused on one particular application for the mobile environment: *location-dependent information service*. The location-dependent information service is one of the typical dynamic content sharing application. It requires dynamic changes of the content structure which consists of location-dependent contents that are specific for the user's locations or situations.

We have developed a location-dependent information system called 'LODIS', which can define an active and dynamic contents easily and has efficient architecture suitable for the mobile environment. In this chapter, the design and the implementation of the system is described.

3.1 The Location-dependent Information Service

The location-dependent information service will bring us the possibility of realizing following applications, for example.

- **Nearest Target Search**
  Searching the nearest target from the user. For example, when a user wants

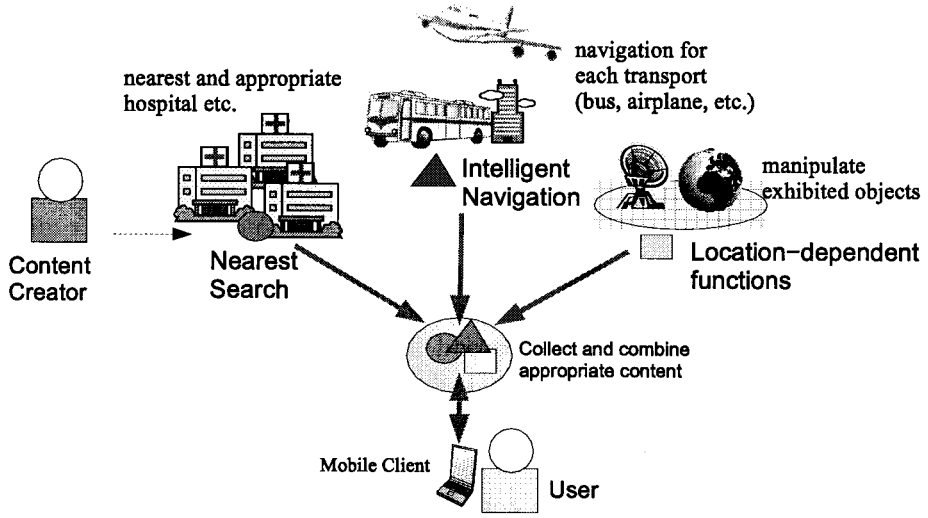

Figure 3.1: Location-dependent Information Service

to find a hospital with certain facilities, candidates of hospitals are listed in order of their closeness to the user. If the distance between the hospital and the user changes, the search results on the mobile client are updated.

- **Intelligent Navigation**
  The user's current position is displayed on a graphical map and directions to the destination from that position are provided.

  When one travels by car or on foot, the system only shows a course to the destination. In addition, at the train station, the mobile client will show the train schedule and upon arrival at the airport, it will show the boarding information.

- **Lactation dependent functions**
  In an exhibition hall or an amusement park, exhibited objects can be operated from a mobile client. For example, users can manipulate a robot at an amusement park from their mobile client.

3.2 System Requirements for Location-dependent information service

To realize the location-dependent information service, following features are required for the system.

- Simple description of the location-dependent contents
  The system must be able to treat dynamic content which changes its structure according to the user's positions. For example, in the 'nearest target search', if hospitals are searched, the result becomes a composed content of the hospital information list. It must change the list structure according to the closeness to the hospitals. Also, to provide 'intelligent navigations' or 'location dependent functions', the provided content must keep a context or a story of the service. Hence the content structure should be able to defined by a service provider.

  However, it is impossible to define the structure of the content for the user's all status, since it becomes combinations of all conditions of user and environment. Hence simple content description framework is required.

- Efficient architecture suitable for mobile environment. Though the dynamic recomposition requires complex process for providing the service, it should not affect on the performance of displaying, playing and executing the content. Hence the efficient architecture for dynamic recomposition is required. In the mobile environment, we also have to consider to reduce the load of network transmission and the amount of CPU power used by the mobile client.

Some researches of location-dependent services have been already done. But none of them satisfy the requirements mentioned above. DATAMAN [7], Mobisaic [11] use a *dynamic document model* to change HTML documents according to the user position and the access time. However, the unit of the information service is only a 'page' in this model, which means it cannot treat dynamic composition of the contents.

3.3 The Mobile Object Model

We proposed a new framework for an active location-dependent information service that can realize the dynamic compositions. The framework is based on a *mobile*

object model, extension of the *dynamic document model*. In this model, the service consists of *mobile objects* which correspond to the location-dependent contents and recomposed according to each user's status dynamically.

In this section, we describe the detail of the mobile object model. Firstly in the next sub-section, we summarize the *dynamic document model* which our model based on.

3.3.1 The Dynamic Document Model

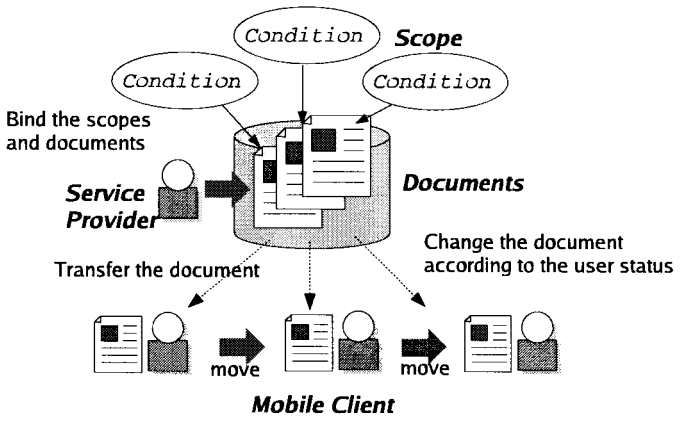

Figure 3.2: The Dynamic Document Model

In the dynamic document model, to describe the changes of the document displayed on the mobile client according to the user status, the content creator defines the conditions of the time and/or location for each document. These conditions are called *scopes*, which decide whether the document should be displayed or not.

When a service is accessed from a mobile client, every *scope* is evaluated using the user's location and the access time and the document having adequate scope is transported to the mobile client. When the user's current status changes (i.e. the user moves or time has passed), scopes are re-evaluated and a new document that satisfied the new conditions is transported (Fig. 3.2). This makes it possible for users to obtain a document of interest according to their movement or change of status.

3.3.2 Basic Concepts of the Mobile Object Model

In the dynamic document model, the unit of the service is a 'page'. Therefore, when the content changes according to the user status, the whole content is exchanged. It means that the content can treat limited combinations of the user status. Furthermore, it is difficult to provide service contexts like tracing of users position, navigation to the destination, and so on.

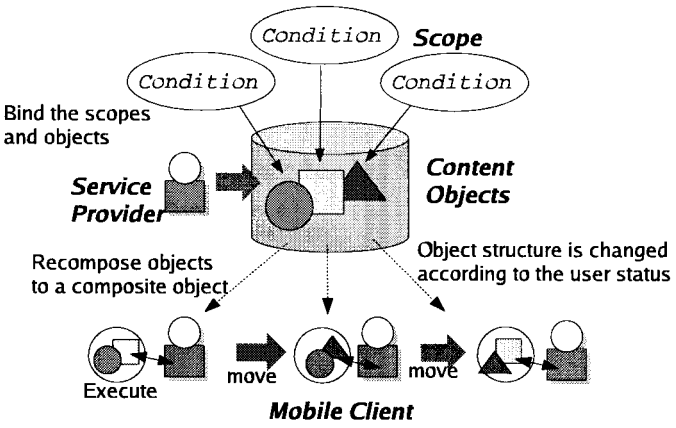

Figure 3.3: Mobile Object Model

In the *mobile object model*, on the other hand, a service consists of objects. These objects are called *mobile objects* and managed as *distributed objects* by multiple service providers in the mobile environment. Location dependent information service for mobile user is realized by recomposing multiple mobile objects to a composite object dynamically and executing it. This makes it possible to provide active location-dependent information services that realize not only dynamic change of the content according to the user status, but also the controlled context of the service.

Similar to the dynamic document model, our model requires *scopes* which may activate mobile objects. When a service accessed, the system evaluates every scope according to user status, obtains adequate objects, and then recomposes them to a composite object. When the user status changes, every scope is re-evaluated and newly obtained objects are recomposed and executed. At this time, service is exchanged (Fig. 3.3). If a same object exists before and after the status change,

the object continues the execution. This mechanism enables users to obtain suitable information without them being aware of changes in their status.

3.3.3 Connectivity Scope

In order for objects to be connected to one other, each object has to provide some kind of inter-object protocol. Accordingly, a mechanism for checking the connectivity of the objects is required because a message sender needs one or more message receivers. In other words, the system requires not only the scopes in the dynamic document model, but also *connectivity scopes.*

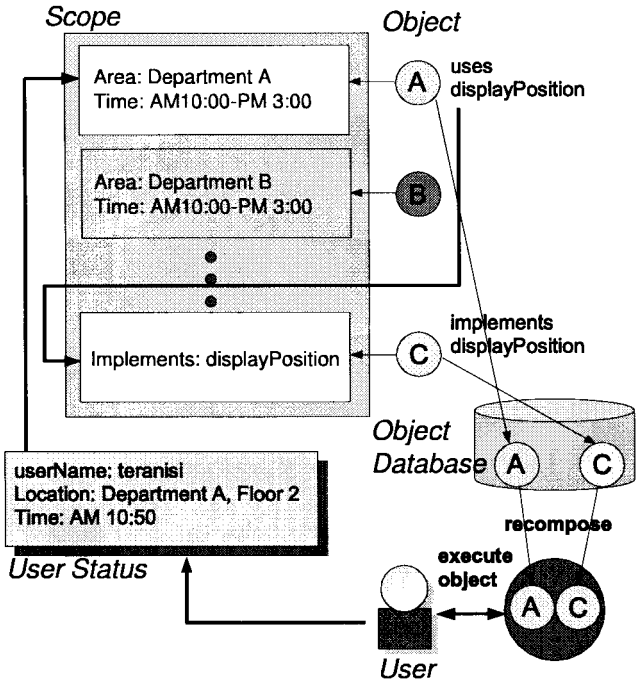

Figure 3.4: Recomposition according to the scope

Fig.3.4 shows an example of object recomposition according to the scope. Object "A" can be executed only if it can send a `displayPosition` message to another object. Object "C" can receive the `displayPosition` message because it implements

the behavior of this message. The connectivity scope of object "A" corresponds to the existence of the object that can interpret the message `displayPosition`, such as object "C".

3.3.4 Hierarchical Scope

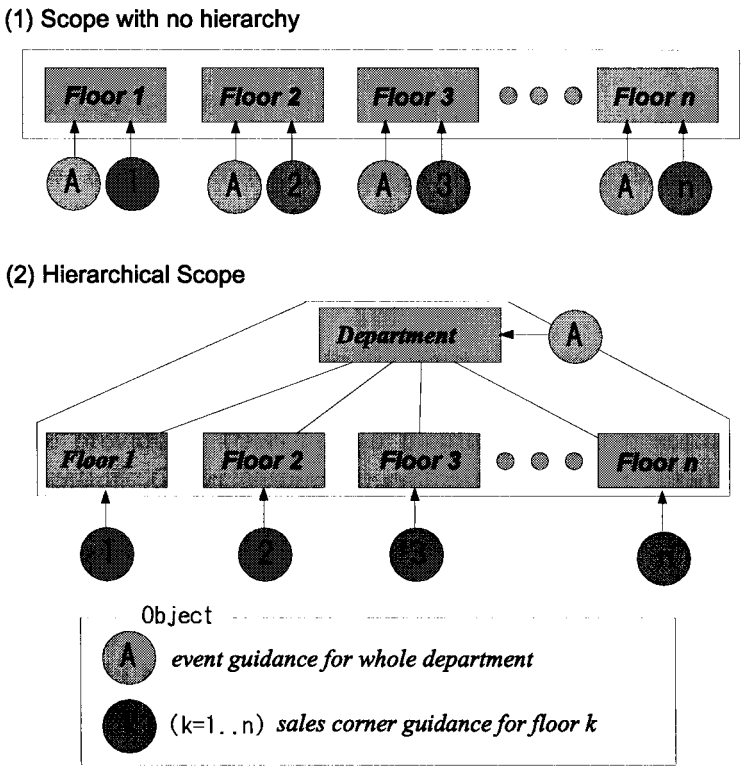

Figure 3.5: Hierarchical Scope

In location-dependent information service with the mobile object model, the same object may be used when user status changes.

Let us consider an example of an information service for a department store. The service provider prepares navigation services for sales corner on every floor and an event guide for the whole building. If the provider defines a respective scope for each floor, the same event guide object must be registered for every scope (Fig.

3.5(1)).

The mobile object model introduces *hierarchical scope* for reducing such redundant object registration. In hierarchical scope, when the user circumstances satisfies a higher level scope, it also satisfies the lower ones.

If the service provider registers a common object (the event guide object in this example) in a higher level scope, it does not have to be registered in every floor scope (Fig.3.5(2)).

3.4 Architecture

In this section, we discuss architectural issues involved in implementing the mobile object model in a mobile environment, and then we describe our architecture.

3.4.1 Required Modules

Fig. 3.6 shows the modules required to implement location-dependent information services using the mobile object model. *ObjectBase* manages mobile objects produced by service providers. *UserObserver* monitors each user status, such as the user's current location. *ObjectRecomposer* evaluates the scopes based on user status and recomposes obtained objects to a composite object. *ObjectExecuter* executes the composite object.

An important consideration is where to locate these modules in a mobile environment consisting of a fixed network and mobile clients.

ObjectBase should be located on the fixed network to make it easy for service providers to add/delete objects as necessary. Moreover, it is natural that this module is distributed to multiple servers in the fixed network because multiple service providers may manage mobile objects. UserObserver must be on the fixed network in order to collect user status. In the mobile environment, it is very important to reduce the computing load of mobile clients because their CPU power is limited. Furthermore, the frequency and amount of data transmission via wireless communications have to be reduced because of bandwidth limitations.

Therefore, we must carefully consider where we locate the ObjectRecomposer and ObjectExecutor in order to obtain reasonable performance. In the next subsection, we discuss the location of these modules in more detail.

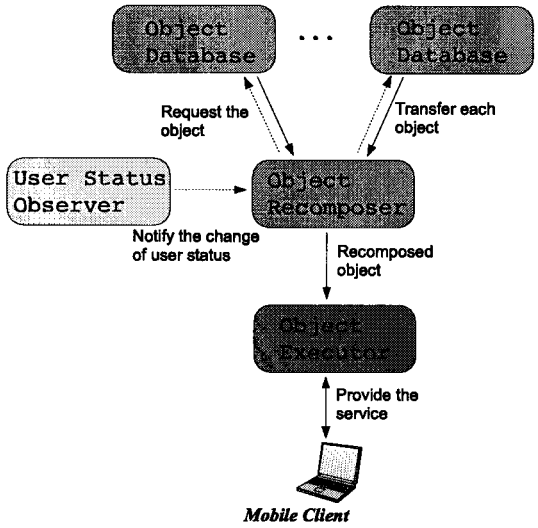

Figure 3.6: Required Modules

3.4.2 Location of Modules

If ObjectExecutor is located in the host server in the fixed network, the mobile client becomes the 'remote display'. It is not suitable because in the mobile environment, network connection is not stable enough to keep smooth remote displaying. It may cause hang-up displaying when the user entered to the tunnel. Therefore, we assume that the ObjectExecutor is located in the mobile client.

The immediate transfer of objects retrieved from ObjectBases is simple. Hence, it appears that ObjectRecomposer should also be located in the mobile client (Fig. 3.7). This architecture allows the system to be constructed based on the traditional client-server model. However, if many objects are used in recomposition, the frequency of connections is very large because a connection must be set up for each recomposed object. Furthermore, because recomposing and executing processes must run in the same mobile client, a large amount of CPU power is required. For that reason, this architecture is not adequate for the mobile environment.

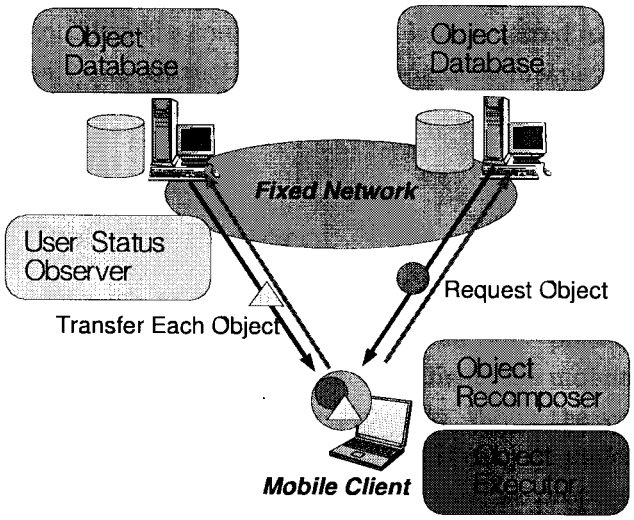

Figure 3.7: ObjectRecomposer is located in the mobile client

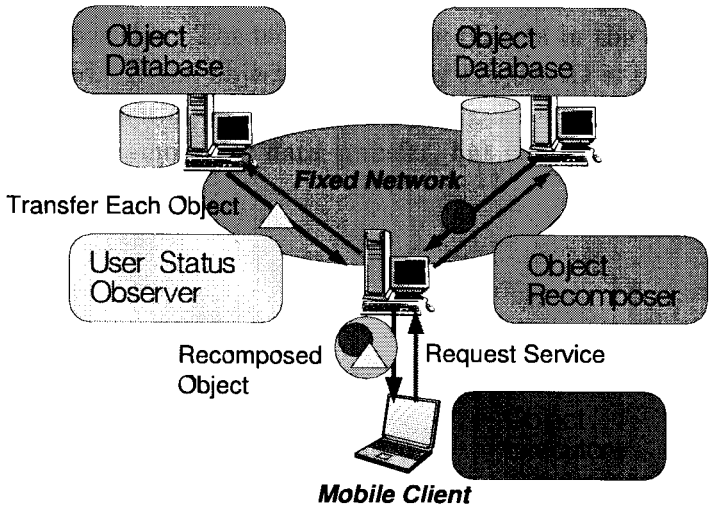

Figure 3.8: ObjectRecomposer is located in the fixed network

In our new architecture (Fig. 3.8), ObjectRecomposer is located at the host

in the fixed network. Mobile objects are recomposed to a composite object at the ObjectRecomposer, and then the composite object is transferred to the mobile client. In this architecture, only one connection has to be made no matter how many objects are used. Moreover, since only ObjectExecutor runs in mobile client, less CPU power is required.

3.4.3 Object Transfer Protocol

In a closed environment such as an amusement park, the same object may be frequently used because the user may return often to the same place. In such a case, when an object is stored in the cache of the mobile client, the mobile client does not have to obtain the same object through the network. When the system reuses objects that are currently running on the mobile client, it may be undesirable for these objects to be re-initialized and transferred. To avoid this, we propose *the differential object transfer protocol (DOTP)* for reusing objects in the cache or objects that are currently being executed in the mobile client. In DOTP, both objects in the cache and objects running in the ObjectExecutor are all registered in the ObjectRecomposer. That is, ObjectRecomposer knows which objects are in the mobile client. If the object already exists in the mobile client, not object itself but rather the Object-ID is used for recomposition. Therefore, only new objects in the composite object have to be transferred. If the object specified by Object-ID is currently running in the mobile client, it continues running (Fig. 3.9). This object transfer protocol can not only reduce the amount of data transfer, but can also avoid interruptions of service.

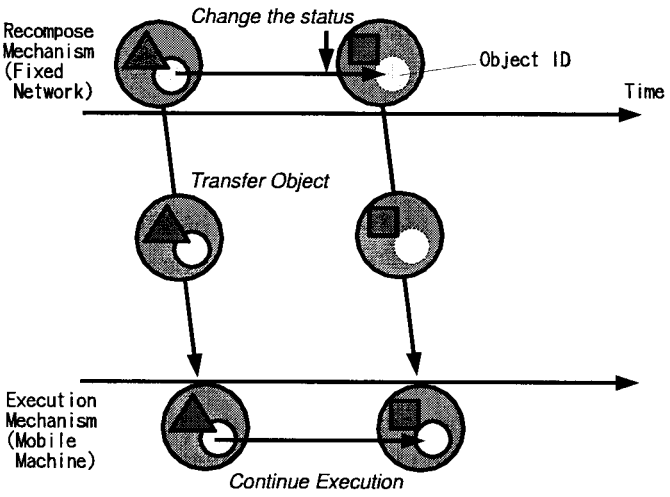

Figure 3.9: Differential Object Transfer Protocol

3.5 Implementation of the LODIS

To verify the effectiveness of the proposed functions, we implemented a prototype system called "LODIS" on a WWW/Java system.

3.5.1 MODS: Mobile Object Description Script

We have developed a description language called "MODS" for describing mobile objects in LODIS. In MODS, service description consists of a series of *scope definition statements* and *object definition statements*. Each scope definition statement defines user circumstances using parameters such as user positions, the relation among other user locations. Each object definition statement defines each mobile object used in the service. This statement includes specifications of object behavior, scope, a series of parameters, and so on. The behavior of the mobile object is defined in Java.

Fig. 3.10 is an example description of MODS. In this example, a part of theme park navigation service is described. There are some hierarchical scope definitions which corresponds to the areas of the theme park. Besides, there are object definitions which registers the navigation objects to the scopes.

`scope {}` is the scope definition statement (lines 4 to 36). This example includes 6 scopes. As described in Sec. 2, scopes can be defined hierarchically. The `is:` field specifies its parent scope in the hierarchy. The scope hierarchy of this example is shown in Fig. 3.11.

User circumstances are determined by condition parameters. Table 3.1 lists the parameters currently available in MODS. When user status changes, all scopes in the hierarchy are evaluated, and objects that are registered with the satisfied scopes are recomposed to one composite object.

`object {}` is the object definition statement (lines 37 to 58). In this example, two objects are defined. Each object is registered with its scope. The object defined from lines 37 to 44 is registered with the scope `"world"`, which is defined in the scope definition statement at line 4. This means that this object can be used in every scope that is a descendant of the scope `"world"` in the scope hierarchy.

The set of the `codebase:` field and `code:` field in the object definition statement specifies the location of the Java class. We use URL to specify the Java class location.

The `parameter:` field defines the parameters used for instantiating the specified Java class. At line 54, an image file is specified as a parameter for instantiating `mods.EasyNavigation` class.

The `implements:` field specifies the interface which is implemented in the object. This field is used for the connectivity scope described in Sec. 2.

The `MobileObject` class is prepared to provide the basic behavior of the object, i.e. migration, recomposition, execution, etc. Every Java class specified in the object definition statement must be a subclass of the `MobileObject`.

Much of the behaviors of mobile objects can be defined by overriding methods of the `MobileObject`. Table 3.2 lists the methods provided by `MobileObject`.

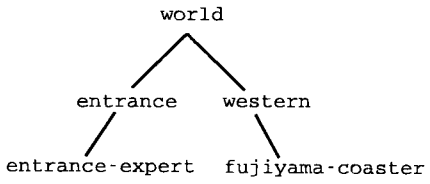

Figure 3.11: Scope hierarchy

```
 1: # amusement.mods
 2: # statements after '#' are comments.
 3: scope "world" {
 4:   condition:
 5:     [mylocation() == "world"];
 6: }
 7: scope "entrance" {
 8:   is:  "world";
 9:   condition:
10:     [mylocation() == "entrance"];
11: }
12: scope "entrance-expert"{
13:   is:  "entrance";
14:   condition:
15:     [mystatus("level") != "novice"];
16: }
17: scope "western" {
18:   is:  "world";
19:   condition:
20:     [mylocation() == "western"];
21: }
22: scope "fujiyama-coaster" {
23:   is:  "western";
24:   condition:
25:      [distance("fujiyama") <= 20];
26: }
27: scope "meet-teddy" {
28:   is:  "world";
29:   condition:
30:      [distance("teddy") <= 2];
31: }
32: object {
33: # uses interface "displayNavigation";
34: #       and      "setTarget";
35:    scope:     "world";
36:    codebase:  "http://server/objs";
37:    code:      "mods.Navigator";
38: }
39: object {
40:   scope:     "entrance";
41:   codebase:  "http://wildboy/objs";
42:   code:      "mods.EasyNavigation";
43:   implements: "displayNavigation";
44:   implements: "setTarget";
45:   parameters:  {
46:      mapimg="http://server/enter.gif";
47:      mapname="entrance";
48:      archive="http://server/ent.jar";
49:   }
50: }
```

Figure 3.10: An example description of MODS

3.5.2 Detailed Architecture of LODIS

Fig.3.12 shows the LODIS architecture. A mobile client of the LODIS only requires Java executor.

LODIS uses HORB [9] for network functionality, e.g. object migration, message passing, and so on. First, an object-execution core, which is implemented as a Java applet, is transferred to the mobile client via HTTP. Next, this applet runs as the HORB client object, and communicates with the ObjectRecomposer which is running as HORB server.

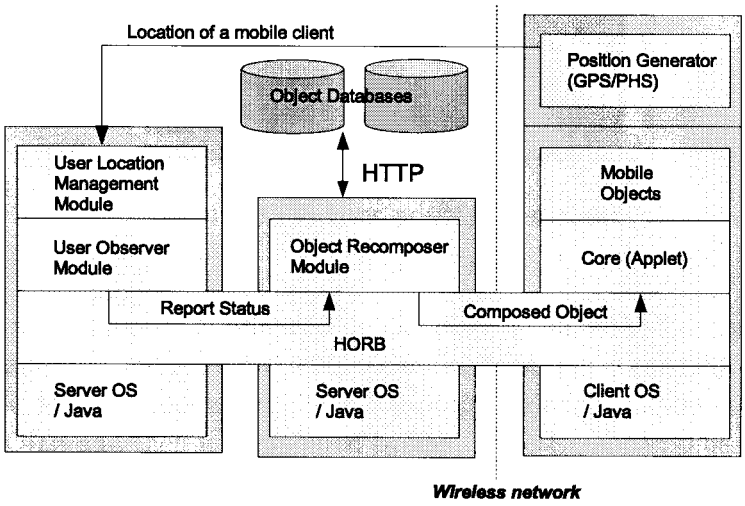

Figure 3.12: LODIS architecture

To detect user positions, we used GPS (Global Positioning System) PC card for the laptop client. Also we are considering to use PHS node searching to detect user position. For confirmation of the behavior of LODIS, we prepared a position generator(Fig.3.13) that enables users to trace their positions by mouse dragging.

The user position information is managed by the UserObserver and it reports user positions to the ObjectRecomposer.

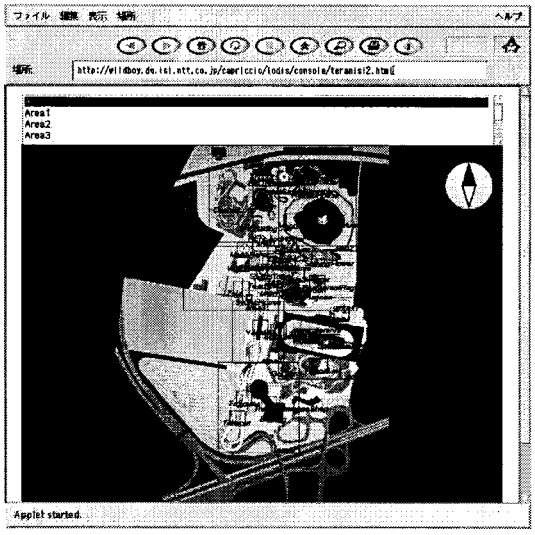

Figure 3.13: User position generator

The Java binary code is provided by the distributed WWW servers. Each Java code is instantiated and recomposed to a composite object in the ObjectRecomposer according to MODS description. This composite object is migrated to the mobile client by HORB and then it is executed by the ObjectExecutor. The migration and execution is done by the DOTP described in Sec. 3.

3.5.3 Example Application

The example application of LODIS works in the network environment shown in Fig.3.14. ObjectBase, UserObserver and ObjectRecomposer are run on the server workstations and ObjectExecutor runs on portable computers. The servers are connected to the 100 Mbps LAN and portable computer is connected to it via a PHS data packet connection. In this prototype system, two clients are connected to the network, therefore, two services are provided on the server at the same time.

Fig.3.17 and Fig.3.18 show the interfaces of this theme park navigator application. In Fig.3.17, there are four mobile objects composed: a user position display object, a navigation object, an attraction guidance object, and a crowdedness display object. In Fig.3.18, detailed information of an attraction is displayed after an

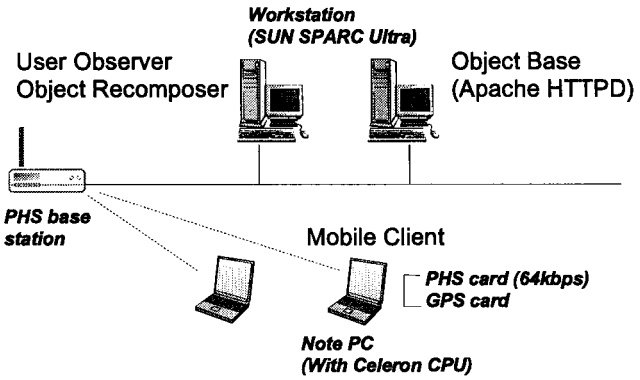

Figure 3.14: Experimental system

anchor point in Fig.3.17 is clicked.

This application consists of 27 objects. The amount of code is about 1200 lines (about 900 lines for Java class definition and about 300 for MODS definition). If the same application is written by the dynamic document model using HTML, more than 3000 pages of documents are required to follow each status of the user.

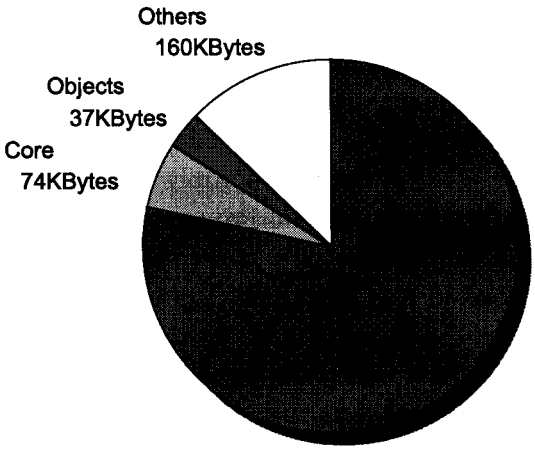

Figure 3.15: Transmitted data per hour

Fig.3.15 is a pie graph showing the various amounts of transferred data for a one-hour execution of this application. "Core" is the LODIS client core which runs

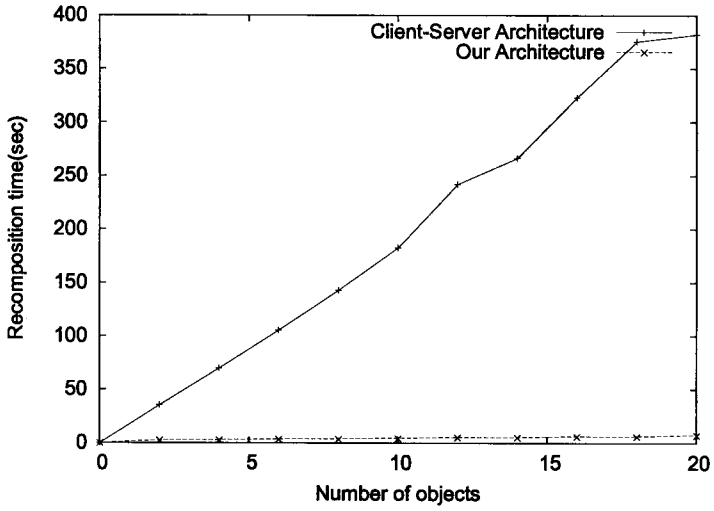

Figure 3.16: Object recomposition time

as the Java applet. This applet is transferred via HTTP at the service start time. "Objects" are mobile objects. "Others" is the application specific data used by mobile objects such as the crowdedness information reported by the server. The figure shows that almost 75% of transferred data is "Image Data". Thirty images (GIF images) were transferred in one hour. The max among these images was 170 Kbytes. To transfer such an image within a reasonable period, 2 seconds for example, more than 1 Mbps bandwidth is required.

Fig.3.16 shows the object recomposition time (includes object transfer time) using client-server architecture (shown in Fig.3.7) and our architecture (shown in Fig.3.8) on the experimental system. The data size of each object used in this evaluation is approximately 0.5 KBytes.

According to the figure, though the object recomposition time grows in proportion to the number of objects, apparently our architecture requires short execution time. For example, when 20 objects are used, the client-server architecture took more than 360 seconds to recompose while our architecture took only 7 seconds.

In general, narrow bandwidth and large latencies of mobile network environment increase the overhead to establish TCP connections between computers. In this evaluation, most of the recomposition time is consumed by the TCP connection

establishments in the client-server architecture since the connection from mobile client is newly required for each object. But in our architecture, by contrast, there are no overhead of TCP connection establishments because the TCP connection between client and 'ObjectRecomposer' host is permanently kept and reused for object transfers. It can be said that our architecture is very efficient for the object recomposition in the mobile environment.

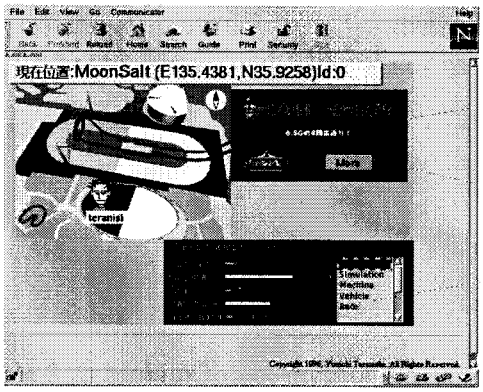

Figure 3.17: Example of the application

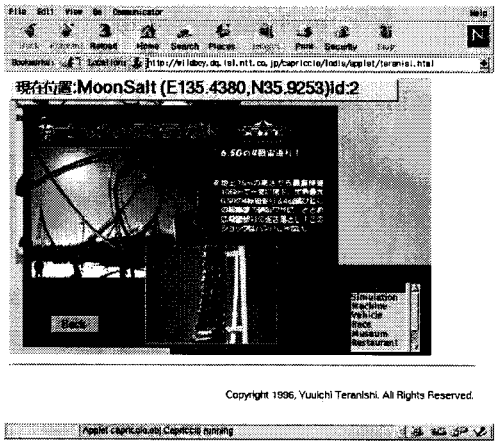

Figure 3.18: Detailed information for an attraction

Table 3.1: Parameters available for scope definition

Name	Parameter	Return value
`mylocation`	none	current location of user
`currenttime`	none	current time
`otherlocation`	user name	location of other user
`distance`	target name	distance from user
`mystatus`	status name	status of user
`otherstatus`	user name, status name	status of other user

Table 3.2: MobileObject methods (partial list)

Name	Meaning
`init`	called when migrated to mobile client
`start`	called when registered scope is valid
`stop`	called when registered scope become invalid
`paint`	drawing function
`repaint`	redrawing function
`getParameter`	get `parameter` field value in the MODS
`mouseDown`	called when mouse is clicked
`getStatus`	get current status of user
`setStatus`	set current status of user
`clearStatus`	clear current status of user
`setReportGeometry`	get geometry information of user
`changeGeometry`	called when user geometry is changed
`getImage`	get image in the URL
`getImplObject`	get objects which have specified interface

Chapter 4

Content Restriction Management System

To realize security and trustfulness in the dynamic content sharing, the digital rights management (DRM) is required.

We have developed a content sharing system called 'ASIA', which has a restriction management system for the derived contents as a DRM feature. In this chapter, the design and implementation of the content restriction management system of ASIA is described.

4.1 Restriction management model of 'ASIA'

The requirements for the trustful dynamic content sharing can be modeled as follows.

- Restriction definitions for content modifications
  To keep the meaning of the content and the right of the creator, the content creator must be able to define the restriction not only the range of the distribution, but also the range of content modifications.

  For example, if there is an intension of the composer that one phrase of the song should not be cut, the song composer should be able to define it as a restriction. The system must protect the distribution of the song which does not contain the phrase.

  If a content is aimed for merchandise, it should be avoided to composed with an advertisement content which is for a commercially confronted product. For

instance, composing a commercial movie for a certain personal computer with a movie of another personal computer should be avoided. In this case, system must protect the compositions of these contents.

As described above, it is required to describe the restrictions including the content modification operations.

- Inheritance of the restrictions
  The restriction defined for original content have to be inherited to the derived content.

  For example, suppose a video content has a restriction that the scene starts from 2 minute to 3 minute is not allowed to cut. Suppose the derived video content which contains this video at the tail of another video which have 3 minutes long. In this case, cutting the scene starts from 5 minute to 6 minute is not allowed.

  If a content can be reused only for the member of a certain group, the derived content should also have the same distribution policy. If a user of the derived content is a non-member, he or she must not create a derived content from it.

- Efficient evaluations of the restrictions
  In the dynamic content sharing, automatic restriction checking is required during the distribution processes. But it should not affect on the performance of distribution processes. Therefore, the restriction management system should have an efficient architecture to evaluate the restrictions.

4.1.1 Current restriction management model

According to the paper [16], current content protection model can be classified into following three types in terms of protection.

- sharing type
- download type
- observation type

In the 'sharing type', the content data only exists on one server which is under the control of the content creator and all users share it. The content data is provided to

the user's client on-demand. In this model, user's permission to view the content is evaluated just before the content data transferring. For example, streaming service of the videos using Real Player(tm) is one of this. Also, the "transpublishing[15]" proposed in the Xanadu project can be classified to this type. In the transpublishing, quotation (in transpublishing, it is called as transquotation) treats derived content. But there is no restriction to quote original content. Hence it cannot treat the modification restrictions.

In the 'download type', the operations to the content is protected after the data is downloaded to the client. For example, the content capsulization technique is the typical model. It encodes the content data and protect the access form users who don't have the decode key. There are many content capsulization techniques. Matryoshka[17], RightsEdge[18], MetaTrust[19], RightsShell[20], Access Tickets[21] are based on the content capsulization model.

In the 'download type', basically the creation of the derived content is not allowed for users. All of the content capsulization systems above don't allow content creations to the users. Only the special content provider can create the content. Furthermore, the created contents are treated as new contents. The derived content cannot inherit the restrictions regardless of the restrictions of the original content. A standard content restriction definition language XrML[13] mainly treats the content capsulization systems. In XrML, derived contents are treated as new contents and does not inherit the original restrictions. Furthermore, in the download type, permission for the operations to the content cannot be judged until the whole content data is downloaded and the operation is applied. Therefore, the raw content data must be downloaded just to judge whether the operation can be applied or not.

Precisely, the 'observation type', does not protect the content but it acts as a deterrent. In this type or restriction management, the operations are evaluated after the derived content is published. However, this model is inapplicable to the valuable contents because it does not take effect while the illegal content is not found. Generally, this model is considered as a protection support mechanism. cIDf[14] can be categorized to this type. It uses watermark technique to embed unique content identifier to the content. If the identifier is detected as a watermark in the distributed content, the content is tested whether it is proper reuse or not according to the meta-data which is bind to the identifier.

In the meta-data definition of the cIDf, there are modification permission attributes. But these attributes has a format of free text (human readable). So these

attributes have to be interpreted by a person and cannot be treated by computers.

ASIA basically belongs to the 'sharing type'. The difference between ASIA and existing 'sharing type' systems is that ASIA can treat the derived content with modifications while others cannot.

4.1.2 The restriction management model in ASIA

As described above, current protection techniques cannot treat the restrictions for the derived content. So they cannot satisfy the requirements described in the beginning of this section. To satisfy the requirements, we proposed a new restriction management model for derived content in ASIA.

Our model have following new features.

- Restrictions can be defined by rules for applying the modification operations.
- The restrictions of the original contents are inherited by saving the operation sequences applied to the derived contents.
- The evaluation of the restriction can be done only by the structure of the content, not the raw content data.

In ASIA, user environment shown in fig.4.1 is assumed. All ASIA users are assigned their own host server in the content delivery network. Each user connects to the host server from their client and access to the contents. The contents consist of texts, images, videos, and sounds. Users can also store and publish the newly created content on the host server. Host servers are connected to each other and users can access to the contents published on the other host servers. Each user can publish a derived content made by the original content from the other host server. At this time, all modification operations are executed on the host server.

The restriction definitions in ASIA is defined as a set of rules which consists of **Condition** and **Restriction** to the operations (fig. 4.2). Each ASIA user define the rules to the each contents when publishing them on the host server.

The **Condition** of the each restriction rule is evaluated according to the user's current status. It contains the condition of user's attribute or environment like age, organizations, current time, client machine type, etc.

If a restriction is applied to the users who belong to the 'groupA', the content creator have to define 'Condition' as "Attribute `Group` equals to `group-A`".

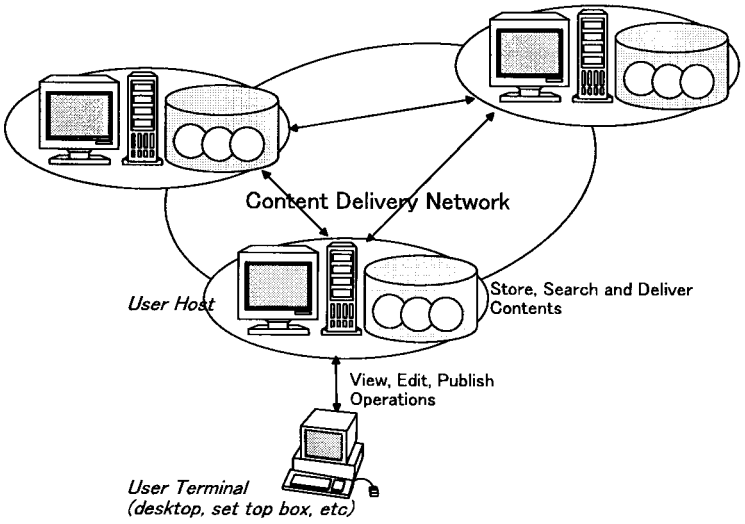

Figure 4.1: User environment model in ASIA

After evaluating 'Condition', the corresponding 'Restriction' is applied. The **Restriction** defines the limitation of operations. For example, the limitation for cutting specified rectangle from an image, combining a video with sound can be defined. The operation to the content succeeds only when the user's status satisfies 'Condition' and the operation is within the limitation of 'Restriction'. The result of the operation is displayed on the user terminal, if the operation succeeds. The operations which can be treated in the system depends on the capability of 'Restriction'. In the AMF 1.0 described in the next section, basic operations for the images, videos, sounds, texts are covered.

The structure of the derived content consists of the reference to the original content and the sequence of the operations (fig. 4.3). These derived content structure can be treated uniformed way with the original content structures. The restriction rules can be bound to the derived content, too. The derived content also can be referred by the other derived content. Fig. 4.3 is an example of the multiply derived content.

The operation restrictions to the derived content is evaluated from the restrictions bound to the original contents. The evaluation proceeds according to the structure of the derived content. At the evaluation, the operation to the derived

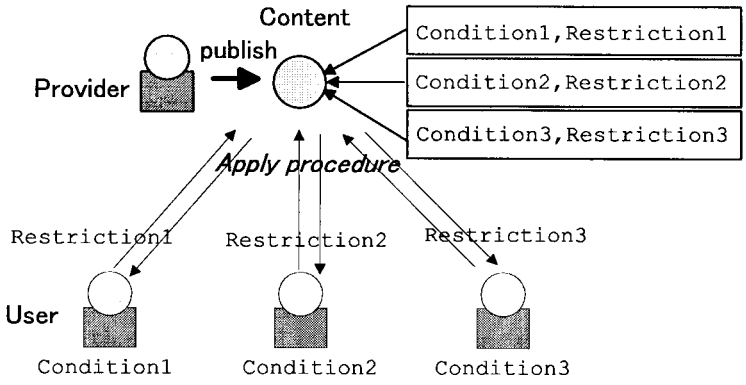

Figure 4.2: Content restriction model in ASIA

content is converted to the operation to the original content. For example, assume a movie C consists of 3 minutes movie A and 10 minutes movie B. When C is cut from 5 minutes to 6 minutes, the cut operation is treated as a cut from 2 minutes to 3 minutes to the movie B at the evaluation.

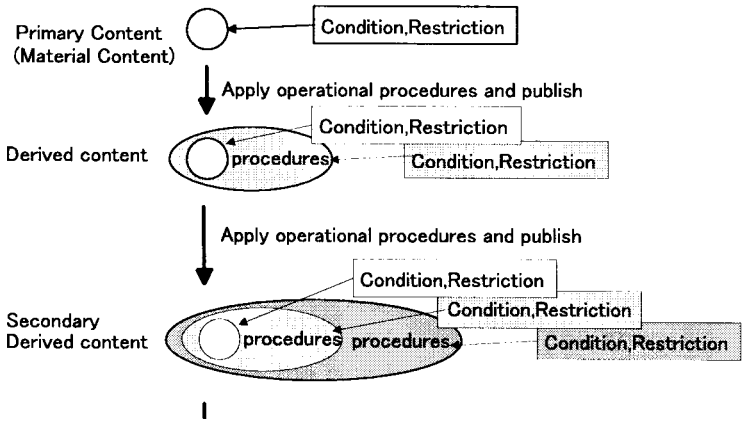

Figure 4.3: The structure of derived content

If a user accesses to the derived content, the operation sequences which are saved as the structure are dynamically evaluated. The restrictions are evaluated with publisher's authority. A user can access to the derived content only when all

operations are granted, i.e. the re-composition of the content is succeeded.

As described above, the operations are stored as the structures and evaluated dynamically at user's access. By this mechanism, the restrictions defined for the original content is inherited to the derived content adequately.

Moreover, in our model, the operation sequences and restrictions are separated from the content data itself. Therefore, it is not needed to retrieve the whole content data to evaluate the operations. This realizes efficient evaluation even when the original content is a huge video data.

It can be said that our model is based on the 'sharing type' protection model because the original content data only exists in the creator's host server and derived content is created dynamically referring the original content.

We didn't mention the security aspect of our model. Of course it is very important to keep the security through the content sharing process. But in our model, all the restrictions and the contents are kept in the host server and all the operations are assumed to be executed on it. So the secureness of the entire system depends on the host server.

4.1.3 An example of the derived content evaluation

Figure 4.4 shows an example of a derived content evaluation.

In this example, a derived content D is created and published by User1. User1 belongs to the groupA and groupB. The content D consists of content B and content C. The content D has a restriction which inhibits the cut operation. The content B is also a derived content, a cut from the beginning to the 100 second of video content A. B has a restriction which inhibits the composition with content C after the year 2004. The content A have a restriction which inhibits the cut operation from 10 seconds to 50 seconds if the user belongs to the groupA. The content C have a restriction which inhibits the compose operation with content E if the user belongs to the groupB.

Let us consider a User2 access to the content D in the year 2003. Firstly, the content structure and restrictions of the content D is retrieved from User1 to User2.

Secondly, according to the content structure, operations are evaluated dynamically with the authority of User1. The User1 belongs to the groupA and groupB and the time is before the year 2004, all the operations to create content D is granted.

Thirdly, the view operations to the content D is evaluated with the authority of User2. As a result, User2 can view the content D since there is no restrictions for

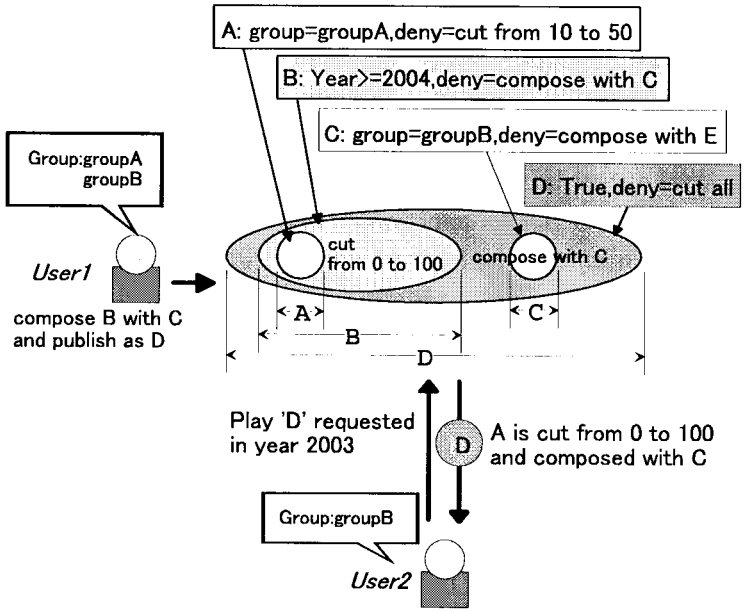

Figure 4.4: An example of the derived content evaluation

view operations.

As shown in this example, if the view operation is executed to the derived content, the content is created dynamically with authority of the publisher and then viewed as the authority of the user.

4.2 The content sharing system ASIA

The content sharing system ASIA is based on the restriction management model described in the former section. In this section, the design of the ASIA system is explained. Mainly the restriction description language AMF that we propose and our implementation model using the software agent is described.

4.2.1 AMF: Restriction description language

We proposed a new restriction description language AMF (ASIA Metadata Format), which can be defined as a meta-data of a content. In the AMF (Version 1.0), texts,

images, videos, sounds and their combinations can be treated. The operations for the content in the AMF Version 1.0 is shown in the table 4.1. Basic operations like play (view), cut, compose, transform(modify the content itself) are covered. 'cut' and 'compose' have spatial and temporal ranges. 'transform' contains size, color, quality modifications. In the prototype system described in the section 4.3 have a plug-in mechanism to append the new operations. AMF is based on the standard RDF[22] framework. Therefore, AMF can be used with other meta-data framework using RDF as long as it does not conflict the meaning.

AMF follows the RDF model shown in the figure 4.5.

Table 4.1: Basic operations in the AMF 1.0

operation	property	description
`play`	—	Play and view the contents. If this operation is denied, any other operations are denied.

`cut`	`from`
	`to`
	`start`
	`end`
		Cut and obtain the partial content. `from` and `to` specifies the temporal range. For example, if 'from' is set to 0.0 and 'end' is set to 100.0, it means the cut from the first 100 seconds of the content. `start` and `end` specifies the spatial range. For example, if 'start' contains the geometry (0,0) and 'end' contains the geometry (640,480), it means the cut for the VGA display.

`compose`	`type` `with`	Compose with the other content. `type` is one of the `serial`, `parallel`, `vertical` and `horizontal`. 'serial' and 'parallel' means temporal composition and 'vertical' and 'horizontal' means spatial composition. `with` attribute can specify the condition of the composed content.

`transform`	`resize`
	`scale`
	`rotate`
	`depth`
	`etc.`
		Modify the content itself. `resize` specifies the scale of the spatial zooming and shrinking. `scale` specifies the scale of the temporal zooming and shrinking, i.e. 'slow' and 'fast' playing. `rotate` specifies the degree of the rotation. `depth` specifies the changing the depth of images.

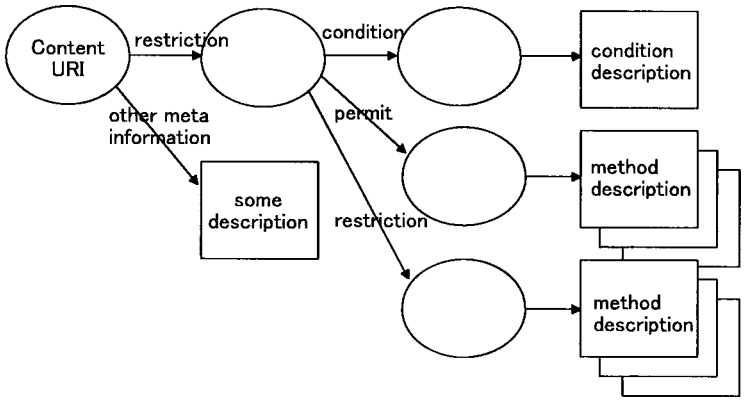

Figure 4.5: RDF model for AMF

The top level description of AMF is a property named `restriction`. `restriction` contains `permit` or `forbid`. Each of them contains the operation method description. If `permit` is specified, the operation method is granted. If `forbid`, the operation method is inhibited. `permit` and `forbid` can contain RDF's 'Bag' attribute, which enables the multiple operation definitions.

RDF's alternative definition 'Alt' can be used for `restriction`. One of the 'Alt' property is used for the restriction. If the `condition` attribute under the 'Alt' property is matched to the current user status, the content of the 'Alt' property is used. Each 'Alt' property corresponds to the each rule of the restriction model described in the former section. The `condition` property corresponds to the 'Condition' of the each rule. It can contain condition like user's property, expiration time, and so on. By the RDF feature, the definition of `restriction` can be referred by multiple content restrictions. It means the restriction definition is reusable.

In the AMF, applied operations to the contents can be treated as a `applied` property. Content creators can specify the derived content which is the result of a certain operation using this property. It provides easy way to specify the derived content since the creator does not have to conscious about the order of the operations.

An example of AMF meta-data description is shown in the figure 4.6. In this example, default namespace (no prefix) is for the AMF. Tags beginning with `rdf:` are RDF properties and tags beginning with `dc:` are Dublin Core[28] properties. In this example, `restriction` is one of the `right-1` or `right-2`. One of them is

```
<rdf:RDF xmlns="http://../amf"
         xmlns:rdf="http://../rdf-syntax-ns#"
         xmlns:dc="http://../dc/elements/1.1/">
 <rdf:Description rdf:about="asia://System/content-x">
  <restriction>
   <rdf:Alt>
    <rdf:li rdf:resource="#right-1"/>
    <rdf:li rdf:resource="#right-2"/>
   </rdf:Alt>
  </restriction>
 </rdf:Description>

 <rdf:Description rdf:ID="right-1">
  <condition rdf:parseType="resource">
   <rdf:type rdf:resource="http://../amf/match">
   <group>group-A</group>
   <applied>cut</applied>
  </condition>
  <forbid rdf:parseType="resource">
   <method>compose</method>
   <with rdf:parseType="resource">
     <rdf:type rdf:resource="http://../amf/match">
     <dc:creator>company-1</dc:creator>
   </with>
  </forbid>
 </rdf:Description>
       :
```

Figure 4.6: An example of AMF

selected according to the result of the evaluation of `condition`. `right-1` consists of `condition` and `forbid`. The `condition` specifies that the user's `group` attribute is equals to `group-A` and the target content is a derived content that is a result of `cut` operation. The `forbid` specifies the `compose` operation with a content which have a `dc:creator` attribute and its value equals to `company-1`.

Therefore, the meaning of `right-1` is "If the content is a derived content by cut and the user belongs to the group `group-A`, composition with the content made by `company-1` is inhibited".

4.2.2 Restriction management using agents

We have developed an agent based restriction management mechanism in ASIA. In our implementation, Content Agent (CA) and User Agent (UA) exists on the user's host server (Figure 4.7). CA manages the published contents and AMF restrictions. CA acts as a content creator in the system. UA authenticates user, accepts the operation to the contents and returns the result of the operations to the user. Users can access to the content only by the UA interface. To publish the content, a user registers the content data and the AMF definition to the CA, via UA interface.

UA contains user's basic personal attributes. If the user's attribute is specified in the `condition` tag of the AMF description, the value in the UA is used for evaluation. In other words, UA acts as a user in the system.

If an user ordered to play the content published in the other host server, the content data is transferred from the CA to the user's UA and played. If a user operates the content through UA interface, the permission to execute the operation is evaluated in the UA, according to the AMF description. If the operation is granted, the operation is executed. In the UA, operations to the content is recorded. If the modified content is published to the CA with AMF description, the reference to the original content and the recorded operations are saved along with the AMF descriptions.

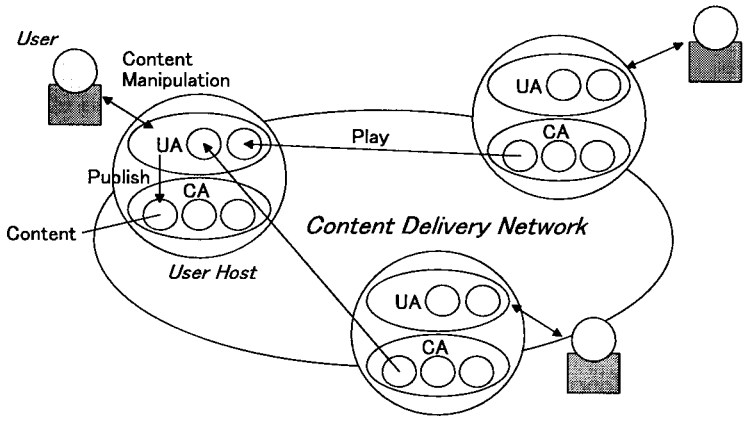

Figure 4.7: Restriction management by agents

4.3 The prototype system and its evaluation

We have developed a prototype system of ASIA using Java [1][32] as a developing language.

In our restriction management model, we have to store the applied operations in the derived content as a recoverable format. Object serialization methods of Java or saving operation log as a text format can be used for saving operations. But in this implementation, we adopted a S-expression (Lisp script) as its saving format for simple evaluation of the operations. Therefore in this prototype, each agent is made as a Lisp interpreter. To implement lisp interpreter using Java, we used kawa[24] framework. Also, Jena[25] is used for RDF parser and JXTA[26] is used for the communication framework for UA and CA. In this implementation, SMIL player and HTML player are embedded in the UA. If a user ordered 'play' the contents, the S-expression structure is converted to the SMIL or HTML and corresponding player is invoked. To implement SMIL player, Xsmiles[27] is used. For HTML player, HTML viewer class of the Java Media Framework (JMF) [29] is used.

This prototype has direct S-expression interface to accept user's operation in the UA. For example, user can register an original content by typing following message in the UA.

[1]Java is the trademark of the Sun Microsystems, Inc.

```
(publish "asia://Host1/kyoto.mpg"
       '(make-content :content-type "video/mpeg"
                      :url "http://server/demo/kyoto.mpg")
       "/home/demo/video.rdf")
```

publish is a function to register a content to the CA and start sharing in the ASIA's content delivery network. In the ASIA system, each content has its unique identifier specified by URI. It is the first argument of the function 'publish'. In this example, the content is published in the server host named 'Host1' with path name 'kyoto.mpeg'. In the second argument, the content data is specified. In this example, an MPEG content data provided by HTTP protocol is specified. The third argument is the file name of the AMF definitions. In this example, the file named 'video.rdf' is specified.

Derived contents can be created if an user sends a message like following to the UA.

```
(publish "asia://Host1/kyoto"
    '(make-content :content-type "application/x-asia"
                   :content (compose
                             parallel
                             (list
                              (open-content
                               "asia://Host1/kyoto.mpg")
                              (open-content
                               "asia://Host2/kyoto.mp3"))))
    "/home/demo/kyoto.rdf")
```

Basically the arguments to the function 'publish' is same as former example but the content of the second argument is the operation function and the reference to the original content. The derived content consists of kyoto.mpg in the Host1 and kyoto.mp3 in the Host2. It is published in the Host1 with path name 'kyoto' with AMF restrictions defined in the file 'kyoto.rdf'.

To play the content, a user has to send following message to the UA.

```
(play (open-content "asia://Host1/kyoto"))
```

open-content is a function to use the content with specified URI. In this example, the playing process follows the following steps.

1. In the Host1 specified by the URI, S-expression and the AMF data is retrieved to the UA.

2. The UA in the Host1 'eval's the S-expression on the permission evaluation environment. In the permission evaluation environment, the S-expression is converted to the applicable one which is within the range of the restriction. For example, the granted range of the cut is narrower than specified, the cut range expression is converted to the granted range. If the content is not granted to compose, the content is removed from the argument of 'compose' operation.

3. The result is transferred to the UA which executed `open-content`.

4. `play` operation is applied to the S-expression obtained by `open-content`. In the playing process, the S-expression is converted to the SMIL or HTML and an appropriate player displays it. At this moment, the content data (videos and sounds) are transfered from the CA to the UA.

4.3.1 An example keyword searching service

To prove the effectiveness of the proposed model, we implemented an example keyword searching service. This service accepts keywords from user, and returns the contents which are matched to the keyword. The results are automatically composed to one content.

This service is executed on the service provider's UA. The service provider can register the "template" which contains the functions to compose the search result. For example, following message registers a template on the CA.

```
(register-template
 "template-A" (lambda (results)
                (compose
                 'vertical
                 (append
                  (list (make-content
                         :content-type "text/plain"
                         :content "検索結果一覧"))
                         results
                         (list (open-content
                                "asia://Host1/CM.jpg"))))))
```

`register-template` is a function to register a template to the CA. In this example, the search result is composed vertically and banner advertisement (CM.jpg) is displayed at the bottom of the content. This template is registered with name 'template-A' in this example.

If the user specifies the keyword and the template, the template is evaluated in the provider's UA and matched contents are composed and displayed. If the user searches with keyword 'england' with template 'template-A', following message is sent to the user's UA.

```
(play (open-content
       "asia://Host1/?keyword=england&template=template-A"))
```

When the service provider's CA accept the request, then it searches the neighbor CAs with the specified keyword. If the keyword matches to the content (or metadata), the S-expression and the AMF is collected. After the search, the UA applies the template to the collected contents. The result of the evaluation is sent to the user's UA.

An example of this service is shown in figure 4.8 and 4.9. In this example, there are two contents which matched to the keyword 'england' and the first one have a restriction to inhibit composing with advertisement content. Although the keyword is same, the result differs because these two are the result of different template. In the figure 4.8, a template named 'template-A' defined above is used. It appends an advertisement content at the bottom of the result. In figure 4.9, a template named 'template-B' is used. It simply composes all the result contents. In the figure 4.8, only the second content is displayed since the first content denies the compose operation with the advertisement content while both of the content is displayed in the figure 4.9.

Figure 4.8: Display with 'template-A'

Figure 4.9: Display with 'template-B'

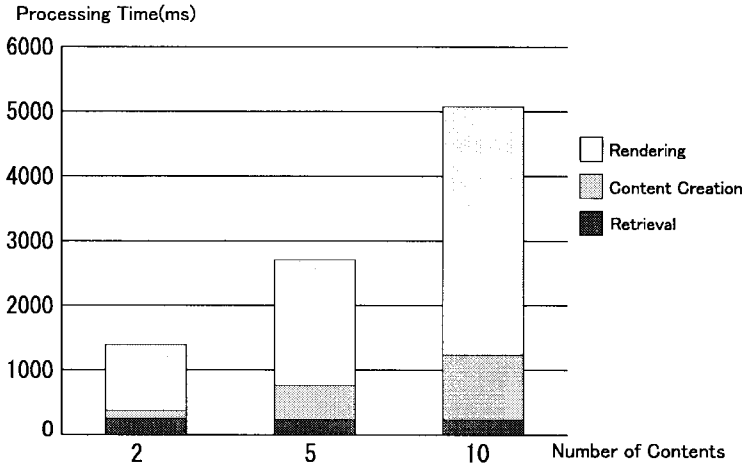

Figure 4.10: processing time for 'template-A'

4.3.2 Evaluation of the prototype system

The graph in the figure 4.10 shows the execution time (from the request issued time to the result displayed time) to retrieve 2, 5, and 10 contents as a result of the keyword searching from the 20 target contents. All of the target 20 contents are derived content. Each derived content consists of two image material and 1.5KBytes AMF description. The AMF restricts the compose operation with advertisement content but all the compose operation during this evaluation is granted.

The environment consists of the Laptop PC (Mobile Pentium III 800 MHz) using Linux (RedHat 7.2) as an operating system. In this evaluation, the target contents are stored in the local host since the retrieval time and the transfer time is not the essential part of this evaluation.

In the figure 4.10, 'Retrieval' is the time for keyword matching retrieval. It took some time to complete because all the meta-data structure is scanned during keyword match in this prototype. If there were indices for the keywords, it would be faster. 'Content Creation' contains the processing time for retrieving all of the S-expression and AMF data, evaluating methods in the templates and creating the derived content structure in the UA. 'Rendering' contains the transforming S-expression to the HTML data and displaying it. That means it also contains the transferring time of the image data.

According the the graph, the processing time grows in proportion to the number of the composed contents. The processing time for 'Rendering' was approximately same as the plain HTML or SMIL browsers made by the Xsmiles or JMF.

As a result, the evaluation process for 10 derived contents finished within approximately 1 second. Though the process is not optimized, whole processing time is only 1 second longer than the plain displaying process (4 seconds). It can be said that our model is sufficiently practical for the personal information service.

4.4 Considerations

In this section, we discuss the features of the design and implementation of ASIA.

4.4.1 Restriction definitions for content modifications

In our restriction management model, the permission to apply the operations to the content is defined as a set of rules which consists of 'Condition' and 'Restriction'. In ASIA, it can be defined as a meta-data for the content.

The 'Condition' defines the condition to select the restriction. It can limit the person by defining the condition of attribute values. Other restriction languages mainly use the user identifier or secret key to limit the user. For example, XrML defines the user by setting the attribute 'Principal'. 'Principal' description contains user identifier or secret key. 'Condition' of the AMF provides more flexible and easy way to identify the person. The 'Restriction' defines the applicable operations. It can define combination of the operations using 'applied' attribute. These features newly introduce the restriction definition including modifications.

In our model, the operation have to follow the result of the evaluation according to the operations applied to the derived content. It is a complex process to evaluate such structures dynamically. We implemented this feature as a simple and flexible module by using S-expression and Lisp interpreter in the prototype system.

4.4.2 Inheritance of the restrictions

In the proposed model, the derived content is represented as a recursive structure. The operation to the derived content is evaluated according to the structure. The operation to the derived content is evaluated correctly by interpreting the operations to the each original content. Also, the condition is evaluated dynamically with the

publisher's authority at the content retrieval. These features enables inheritance of the restrictions in the content sharing process.

4.4.3 Efficient evaluations

In our model, the structure of the derived content and AMF definitions can be treated separately from the raw content data. Since the restrictions for the derived content is evaluated using only the content structure and AMF data, it is needless to retrieve the whole content data to evaluate the operations. This realizes efficient evaluation even when the original content is a huge video data.

This feature is suitable for the dynamic content sharing. For example, in the keyword searching service, if ten contents are matched to the keyword but only one content is allowed to compose, the rest of nine contents are not displayed. At this time, only the structures of these nine contents are needed for evaluation in our model. Therefore, the result of the searching can be displayed without retrieving raw content data of these needless nine contents.

4.4.4 Security considerations

The proposed model is basically belongs to the 'sharing type', which means the strength against the illegal copies has same security level as the streaming service. If the CA and UA in ASIA are connected to each other by public network, it requires encryption mechanism (e.g. SSL, IPsec, etc.) to prevent information leakage.

In our model, protection takes effect only in the host server. It does not trace the outside of the system. Therefore, it cannot avoid to capture the displayed screen and reusing it illegally. To avoid this, content watermarking technique like cIDf is required to identify illegal user.

The security for the operations in our model depends on the host server. The host server have to be run on the secure environment which can protect attacks from the dishonest user. For the better performance, UA should run near the user but if it runs on the user's personal computer, the UA has to be tamper-free. It is easy to achieve if the hardware has limitations like set-top-box, but is very difficult for the normal personal computer. Therefore, the host server should run by the trusted organization.

Moreover, the model requires the reliability of the attributes of users. UA stores the attributes of the user and it is referred by the evaluation of the content restric-

tion. For example, if a man pretends a woman and registers the gender attribute as female, he can use a service for women only. In our prototype, attributes defined by the user are used without any assurance. Therefore, following improvements are required, for example. (1) Limit the person or organization to the trusted ones who define the attributes (2) Query the reliability of the attributes on-the-fly (3) Use the 'attribute certificate'[23] technique.

It should also be confirmed whether the restriction definitions are defined by the proper creator and correctly correspond to the content. To improve the reliability of the restriction definitions, digital signature framework like XML Signature[31] can be used.

Chapter 5

Concluding Remarks

In this paper, we have proposed new mechanisms for implementing dynamic content sharing applications.

Firstly, we proposed a mechanism to realize 'location-dependent information service', a typical dynamic content sharing application in the mobile environment. In our mechanism, content objects are recomposed to a composite object according to the user status.

Also proposed was an architecture for implementing this mechanism in which the object recomposer is located on a fixed network in the mobile environment. We have finished implementation of the basic mechanism of a prototype system called "LODIS". Behavior appropriate for location dependent information service has been confirmed. The system could be improved by:

- making it easier to develop mobile objects. For example, by providing a debugger with scope visualization tool,
- changing the service according to user profiles or access logs. For example, enable the system to change the guidance information based on a user's previous behavior, and
- prefetching or caching data to improve performance of the system. For example, prefetching or caching images using user's current location and direction.

Secondly, we proposed a new restriction management model to realize the trustful dynamic content sharing. By our management model, restrictions for the content modifications in the derived contents can be treated. We defined a restriction description language called 'AMF' based on our model. We have developed a prototype

system called "ASIA" and proved the feasibility of our model. Our prototype system requires following improvements.

- The performance for the evaluation
  Optimization for the evaluation process is remained to be considered. In addition, the searching of the AMF definitions from the distributed servers has to be treated.

- Strengthen the security
  As described in the former section, there are many ways to strengthen the security of our system. It depends on the security policy of the system. Moreover, the privacy of the user should also be considered.

- Extend the applicable systems
  Our model assumes the closed content delivery network. Enhancing the model to the open network is a challenging task.

We are going to improve the usability of the system as well as the problems listed above, toward the powerful and useful content sharing system.

Acknowledgment

I would like to express my sincere appreciation to Prof. Shinji Shimojo of Cybermedia Center of Osaka University as a supervisor of this dissertation. I would also like to express my deep thanks to Prof. Hideo Miyahara, the President of the Osaka University and Prof. Teruo Higashino of Osaka University. I wonder whether I could have accomplished the dissertation without their advice and continual encouragement.

I would also like to express my appreciation to Mr. Norihiko Sakurai and Mr. Katsumi Teranaka of NTT Japan for providing me with such a good chance to study. I am especially grateful to Dr. Tetsuji Satoh of NTT Communication Science Laboratories for his supports in my study and research work.

I deeply thank Assistant Prof. Kazutoshi Fujikawa of Nara Institute of Science and Technology, and Prof. Masayuki Murata of Osaka University for their encouragement. I really thank Prof. Shojiro Nishio and Assistant Prof. Masahiko Tsukamoto of Osaka University for a number of valuable suggestions and discussions which greatly improved the quality of my works. I would like to thank Mr. Takeshi Okuda of Nara Institute of Science and Technology, Ms. Kaori Toyoki of Hitachi Ltd. for their cooperations on this study. I would like to express my thanks to Assistant Prof. Naoki Wakamiya of Osaka University, Dr. Junzo Kamahara of Kobe University, Assistant Kazunori Ueda of Osaka University, and members of Multimedia contents architecture research group for their encouragement.

And I want to thank Mr. Yoshihiro Umemoto, Mr. Fumiyuki Tanemo, Mr. Tomohiro Hasegawa, Mr. Taro Yamamoto of NTT Japan and my friends for their support and encouragement.

Finally, I am sincerely grateful to my family Madoka and Aki for their supports and patience.

Yuuichi Teranishi
Osaka, December 2003

References

[1] World Wide Web Consorcium: "Synchronized Multimedia Integration Language (SMIL 2.0)". available at `http://www.w3.org/TR/smil20/`.

[2] TV-Anytime Forum: "Specifications and Requirements of the TV-Anytime Forum". available at `http://www.tv-anytime.org`.

[3] Jan Bormans, Keith Hill: "MPEG-21 Overview v.5". available at `http://www.chiariglione.org/mpeg/standards/mpeg-21`.

[4] J. Kamahara, S. Shimojo, et al. TV News Recommendation System with Automatic Recomposition. *Proceedings of 1st International Conference of Advanced Multimedai Content Processing*, 1998.

[5] Kazuyoshi Mii, Yuuichi Teranishi, Akira Kojima, Hiroki Akama, Tetsuji Satoh, Katsumi Teranaka, Junzou Kamahara, Shinji Shimojo and Shojiro Nishio. ENHANCEMENT OF NEWS ON-DEMAND SYSTEMS VIA USER-ORIENTED NEW DIGEST COMPOSING MECHANISM EUROMEDIA'99, Apr 1999.

[6] A. Acharya, B. R. Badrinath, T. Imielinski, and J. C. Navas. A WWW-based Lication-dependent Information Service for Mobile Clients. *Fourth International WWW Conference*, 8(1), Jan 1994.

[7] A. Acharya, T. Imielinski, and B. R. Badrinath. DATAMAN project: Towards a Mosaic-like Location-Dependent Information Service for Mobile Clients. In *Rutgers University Department of Computer Science*, 1994.

[8] F. Bennett, T. Richardson, and A. Harter. Teleporting – Making Applications Mobile. In *IEEE Workshop on Mobile Computing Systems and Applications*, Dec. 1994. Santa Cruz.

[9] S. Hirano. HORB — The Magic Carpet for Network Computing —. http://ring.etl.go.jp/openlab/horb/.

[10] F. Kaashoek, T. Pickney, and J. Tauber. Dynamic documents: Mobile wireless access to the WWW. In *Workshop on Mobile Computing Systems and Applications*, 1994.

[11] G. M. Voelker and B. N. Bershad. *MOBISAIC: AN INFORMATION SYSTEM FOR A MOBILE WIRELESS COMPUTING ENVIRONMENT.* Kluwer Academic Publishers, Boston, Mar 1996.

[12] XMCL initiative: "XMCL (eXtensible Media Commerce Language)". available at `http://www.xmcl.org/`.

[13] ContentGuard: "XrML (eXtensible rights Markup Language)". available at `http://www.xrml.org/`.

[14] Content ID Forum: "Content ID". available at `http://www.cidf.org/`.

[15] Nelson, T.H.: "Literary Machines", Theodor H.Nelson (1981)

[16] 櫻井紀彦，木俵豊，高嶋洋一，谷口展郎，難波功次：“コンテンツ流通における著作権保護技術の動向”, 情報処理学会論文誌 Vol.42, No. SIG15 (TOD12), pp. 63–76 (2001)

[17] 加賀美千春, 森賀邦広, 塩野入理, 櫻井紀彦：“コンテンツ流通における自律管理を目的としたカプセル化コンテンツ Matryoshka”, 情報処理学会研究報告 2000–DPS–97, pp. 99–104 (2000)

[18] ContentGuard: "RightEdge", available at `http://www.contentguard.com/`.

[19] InterTrust: "MetaTrust Utility", available at `http://www.intertrust.com/`.

[20] 中江政行, 細見格, 市山俊治：“ユーザ要求に適合したサービスを提供するカプセル化コンテンツ”, 電子化知的財産・社会基盤研究会, pp. 79–86 (1999)

[21] 申吉浩:“著作権保護に応用される暗号技術”, 情報処理学会研究報告 2000–IM–36, pp. 27–34 (2000)

[22] World Wide Web Consorcium: "Resource Description Framework (RDF) Model and Syntax Specification ". available at `http://www.w3.org/TR/REC-rdf-syntax`.

[23] Jan Bormans, Keith Hill: "MPEG-21 Overview v.5". available at `http://www.chiariglione.org/mpeg/standards/mpeg-21`.

[24] R. Housley: "An Internet Attribute Certificate Profile for Authorization". RFC3281, (2002)

[25] Hewlett-Packard Company: "The jena semantic web toolkit". available at `http://www.hpl.hp.com/semweb/jena-top.html`.

[26] Sun Microsystems, Inc.: "Project JXTA". available at `http://www.jxta.org`.

[27] X-smiles.org: "X-SMILS, an open xml-browser for exotic devices". available at `http://www.xsmiles.org/`.

[28] Dublin Core Metadata Initiative: "DCMI Metadata Terms". available at `http://dublincore.org/documents/2003/03/04/dcmi-terms/`.

[29] Sun Microsystems, Inc.: "JavaTM Media Framework API". available at `http://java.sun.com/products/java-media/jmf/`.

[30] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax and Semantics", RFC 2396, (1998)

[31] Donald E. Eastlake 3rd, Joseph M. Reagle Jr., David Solo "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, (2002)

[32] Sun Microsystems, Inc.: "The Source for Java Technology". available at `http://java.sun.com/`.