Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2011, Article ID 687071, 8 pages
doi:10.1155/2011/687071

Research Article

Distributed Storage Manager System for Synchronized and Scalable AV Services across Networks

Frank X. Sun,[1] John Cosmas,[2] Muhammad Ali Farmer,[1] and Abdul Waheed[1]

[1] *British Institute of Technology & E-Commerce, 258-262 Romford Road, London E7 9HZ, UK*
[2] *Department of Electronic & Computer Engineering, School of Engineering and Design, Brunel University, Middlesex UB8 3PH, UK*

Correspondence should be addressed to Frank X. Sun, frank@bite.ac.uk

Received 29 July 2010; Revised 11 March 2011; Accepted 11 April 2011

Academic Editor: Stefania Colonnese

This paper provides an innovative solution, namely, the distributed storage manager that opens a new path for highly interactive and personalized services. The distributed storage manager provides an enhancement to the MHP storage management functionality acting as a value added middleware distributed across the network. The distributed storage manager system provides multiple protocol support for initializing and downloading both streamed and file-based content and provides optimum control mechanisms to organize the storing and retrieval of content that are remained accessible to other multiple heterogeneous devices.

1. Introduction

In the Savant project (http://dea.brunel.ac.uk/project/Savant/), as part of the overall system architecture providing the end-to-end application for producing, delivering, and using of enriched interactive TV content, the content access system (CAS) presents the scalable (where scalability means content personalization, device independence and network independence) and synchronized service to the user by means of multiple heterogeneous devices [1]. It is designed as a home media server which adapts the service so that it can be consumed in a personalized way using three different device classes with different properties: a conventional TV set for traditional viewing, a TabletPC as a portable powerful, highly interactive personal device, and a PDA as a portable lightweight personal device. While the TV set is connected directly to the home media server, the portable devices will communicate with the server using IP via a WLAN connection. Additional content to be synchronized with the main content, such as a signer or multiple camera views, can be delivered over broadband or broadcast networks. The CAS supports the presentation synchronization transparently.

Multimedia home platform (DVB-MHP) is an open middleware system standard designed by the DVB project for interactive digital television. The MHP enables the reception and execution of interactive, Java-based applications on a TV set. Interactive TV applications can be delivered over the broadcast channel, together with audio and video streams. These applications can be, for example, information services, games, interactive voting, e-mail, SMS, or shopping. Although there is a certain amount of storage management incorporated with multimedia home platform (MHP 1.2) specification, this is not sufficient for the storage requirements of the home media server system [2]. At present, the storage management functionality specified by MHP is restricted and based on the Java File I/O. These operations permit file access to persistently stored content and access to content downloaded using a specific broadcast channel. Nonetheless, additional control mechanisms are necessary to sort out multiple content formats that are consumed by the terminal in a variety of ways. In addition, there is no support for managing the downloading and extraction of content by means of multiple protocols used within the broadcast network. Therefore, an additional storage management system needs to be defined and implemented.

This paper provides an innovative solution, namely, the distributed storage manager that opens a new path for highly interactive and personalized services. The presented

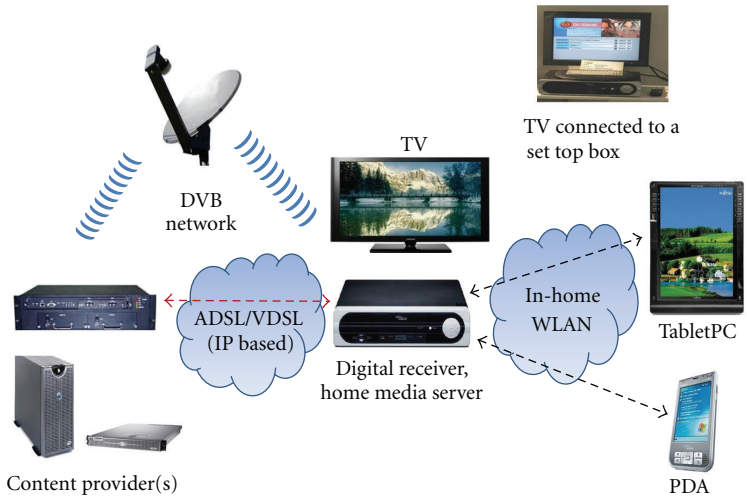

Figure 1: Scalable rich media TV service.

functionality of distributed storage manager provides an enhancement to the MHP 1.2 storage management functionality acting as a value-added middleware distributed across the network. The distributed storage manager system provides multiple protocol support for initializing and downloading both streamed and file-based content and provides optimum control mechanisms to organize the storing and retrieval of content that are remained accessible to other multiple heterogeneous devices such as Set Top Box, TabletPC, and PDA. A novelty distribution module based on the CORBA distributed computing environment which was embedded in the distributed storage manager was designed. The distributed storage manager provides location transparency of service and the functionality to transport multimedia content from server to client or vice versa.

This paper first describes the key parts of the content access system and then describes the architecture of the distributed storage manager system, introduces each of the components, and describes their functions within the system. Finally, conclusions are drawn.

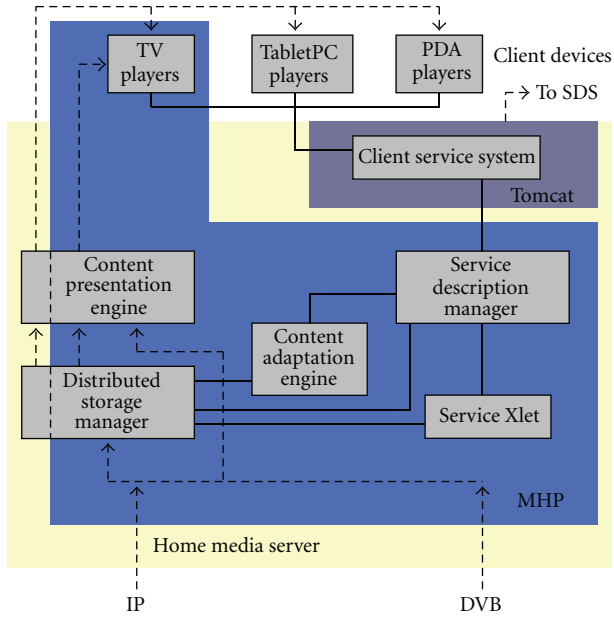

Figure 2: System architecture of content access system.

2. Content Access System (CAS) Architecture

For the purpose of supporting service scalability, a content access system has been developed [3]. The content access system (CAS) was considered as an engine for controlling and managing services. It presents the scalable service to the user by means of multiple heterogeneous devices (Figure 1). The setup of the CAS consists of a home media server (HMS), a Fujitsu-Siemens Activity 300, and three clients, a TabletPC (Fujitsu-Siemens Stylistic (ST Series)), a PDA (Fujitsu-Siemens PocketLoox), and a TV which is connected to a set top box (another Fujitsu-Siemens Activity 300).

The HMS receives all information from the service delivery system (SDS), stores essence and metadata, and makes it accessible for the clients via the client service system. With the exception of the client devices, all components built upon MHP (multimedia home platform) middleware shown in Figure 2 reside on the HMS.

The CAS presents the scalable service to the user by means of multiple heterogeneous devices. The core element of the CAS is a home server at the premises of the customer that provides fine-granular scalability and personalization of the service. It will adapt the service such that it can be consumed using three different device classes with different properties: a conventional TV set for linear viewing, a TabletPC as a portable powerful, highly interactive personal device, and a PDA as a portable lightweight personal device. To realize this, content transcoding may be necessary on the home server. While the TV set will be connected directly to the home server, the portable devices will communicate with the server using IP via a WLAN connection. Using the

service on the three different device classes is termed service scalability. If service components have been transmitted using multiple networks, it is the responsibility of the CAS to reassemble these components to form a composite service and to present them synchronously.

The content presentation engine presents and synchronizes content to the user in the form specified by the user interface, so it has two purposes: content delivery and content synchronization. The service description manager is responsible for receiving, updating, maintaining, and interpreting the service description and triggers processes depending on the information found there. It resides inside the MHP world and is composed of two logical subparts, called passive and active parts, respectively. The content adaptation engine is responsible to adapt multimedia content such that it can be displayed on a variety of terminals with different resources and can be transferred over networks of varying properties. The service Xlet fulfils two purposes within the CAS. First it signals a DVB service, for example, a TV programme, which carries a specific service. After tuning into the channel, the service Xlet is started through the MHP application manager. The service Xlet then controls the application in the CAS. The second function of the service Xlet is to transfer the metadata to the CAS modules.

3. Distributed Storage Manager System Architecture

3.1. Functional Overview of Distributed Storage Manager System. Although there is a certain amount of storage management facilities incorporated with MHP, this is rather rudimentary and not sufficient for the storage requirements of terminal system since it is restricted and based on the Java.File.IO package. Also an MHP-enabled terminal requires additional content management and control specifically if content/data is being delivered over multiple transport protocols, channels, and networks. There are not control mechanisms to organize multiple content formats that are consumed by the terminal in various ways. Additional storage and retrieval management is required for multimedia content that is associated with the main AV multimedia content. The current storage management mechanisms provided by MHP do not distinguish between the various uses of content, and it does not implement any particular searching mechanism, so it makes the process of content retrieval inefficient. So a storage manager system for MHP was required as part of the terminal middleware to accomplish a rather complex task involving the access and extraction of content via different DVB data delivery channels, such as DVB object carousel, DVB PES, DVB private sections, DVB IP sections (multicast IP/UDP packages via MPE), and various IP transport channels. In addition, the location of content extracted over such channels must be recorded in a consistent manner. This permits the search and retrieval of the absolute file location of content in an efficient and timely manner.

The storage manager system has access to an interface to download content on to multiple heterogeneous devices. There are three different user devices as the heterogeneous devices:

(i) Set top box (Fujitsu-Siemens Activy 300) is connected with TV and remote control. It provides large storage, and its components include a wavelancard, IP connection (also accessible via UMTS), and a DVB card.

(ii) TabletPC (Fujitsu-Siemens Stylistic (ST Series)) is operated via touch screen and keyboard and used as a portable device at home. Its components are a DVB-T card and a wavelancard for the local network.

(iii) PDA (Fujitsu-Siemens PocketLoox) is used as the mobile device for "on-the-move" usage. It is equipped with storage media and can automatically connect to the STB. Its components are wavelancard and UMTS connection.

So as one of the fundamental objectives of the storage manager system was to allow multiple heterogeneous devices and multiple server processes in a distributed environment. There is no mechanism for distributing content to different types of terminal devices within MHP, so the storage management system has to be implemented on a distributed platform. Therefore, an additional storage management system needs to be defined and implemented to provide solutions to these issues.

Meanwhile, the storage manager system is responsible for storing, retrieving, and organising input content from the object carousel, private sections (broadcast channel), and IP content (DSL, GPRS) via the broadband channel as well as transcoded content from the content adaptation engine. The storage manager provides the access control with functions to trigger and extract content from the delivery channels, that is, DVB broadcast channel and IP broadband channel [4–7]. These functions include the following:

(i) DSMCC object carousel, DVB PES, and DVB private sections downloading from the DVB transport stream,

(ii) IP extraction within multiprotocol encapsulation (private sections)

(iii) IP extraction from the broadband channel.

The storage manager, is also responsible for the storing, retrieval, managing and maintaining of persistently stored content decided by the user and also manages the removal of no longer used content. The SM runs on the home media server; it provides a mechanism for organizing and managing the persistent storage on the home media server. When any content is persistently saved to disk, any metadata describing the content is also persistently saved with the content. The storage manager also stores metadata that describes the user profile/preferences during the system initialization process.

A structured database is generated by the storage manager from metadata, which contains the location of content on the hard disk. The storage manager also is responsible for providing absolute file locations of content that is stored on disk to external other CAS system components, for example, service description manager or content adaptation

engine. The storage manager allows external access to it and executes storage manager functionality. This therefore allows these external other CAS components to drive the content presentation engine and present content to the user. The storage manager provides a reference or pointer (URL or absolute file path) of the content and forwards it to the calling component. The storage manager reports the current location of content to the service description manager, thus keeping the service description up to date.

3.2. Storage Manager System Functional Components. The storage manager system enables applications to store and retrieve content to/from the home media server persistent file systems. For the storage manager (SM) to communicate to external software components (i.e., service description manager, content adaptation engine), java events/listeners are used and an external interface is provided for external software entities to submit requests to the storage manager.

The SM triggers the carousel download when a service has been selected. The SM sets the carousel root based on MHP properties and the selected service locator (DVB locator). It also controls access to the content contained within the carousel. The request coming from the service description manager for carousel content consists of the relative file path of the desired carousel content. The relative file path is combined with the carousel root, and then the absolute file path of the content is provided. After the SM has confirmed that the content is available on the carousel, then the absolute file location is returned.

A section filter (ring filter) is used to extract data carried within the private sections of the transport stream (DVB over IP). This software component assumes that the payload of the private section is IP packets, that is, MPE. The payload of the IP packets and the use of the IP packets are design specific. The Java.net package is used to create sockets thus supporting the sending and receiving of IP (UDP) packets via the return channel. The SM loads a user profile once a login has been successful. The contents of the user profile are design specific although the profile should at least provide the SM with the user's persistent root directory.

The current implementation supports transmission of media items over the DSMCC object carousel and private sections. After receiving a DVB locator, the module decides by way of the DVB SI if the media item is transferred in an object carousel or in private sections. The term "media item" is used for a set of files, for example, HTML pages usually refer to pictures, style sheets, and other files. Media items in the object carousel are stored immediately after they were requested. This means that the file has to be in the carousel when the request comes in. The actual extraction of files from the object carousel is part of the MHP implementation. The module therefore uses the MHP DSMCC API to make use of it.

After a media item has been stored, the storage manager informs its listeners by events that there is a new content available. There are two kinds of events: one indicates that the storing succeeded (Storage Event), and the other one indicates that the storing failed (Request Failed Event). Failure of storing media items can occur for instance if some sections of a media item get lost. The components of storage manager and their relationships are listed below and shown in Figure 3.

(i) Properties manager: the properties manager is used to store and load system properties, terminal characteristics, user preferences, and user profiles.

(ii) Persistent storage: the persistent storage facility is provided by the storage manager.

(iii) Storage manager implementer: the storage manager implementer provides the root class of the storage manager subsystem and provides overall control of the software components created by it.

(iv) Media item manager: the media item manager is defined for the storage manager so that the storage manager knows which media item it has downloaded, where to find the media item it should remove (i.e., delete from the persistent storage), and how to do a clean-up operation.

(v) Download clients: the download clients defines an interface for all the clients to trigger a file download and proposes a common abstract interface for all download clients.

(vi) Carousel manager: the carousel manager is central instance to manager locating and attaching of DSM-CC object carousel.

(vii) Content location table: the content location table is used to construct a table, based on the supplied metadata, containing information about content.

(viii) Content container object: it represents a storage container for content description data fields including a content item's relative filename, absolute file name, a collection of associate content relative filenames, segment start time, and segment duration.

3.3. Storage Event Dispatching Algorithm. When a request media item event is issued by the service description manager, the storage manager uses the following algorithm to determine if and how the media item should be downloaded and which storage event is thrown:

(i) with checking in the persistent storage file, if the media item is already locally available and has already been cached, then the event will be ignored;

(ii) if the media item has an RTSP (real-time streaming protocol) media locator point to the service delivery system (SDS), save the information that it is available in the media item management and dispatch event that the media item is now available with this media locator;

(iii) if the media item has a DVB locator, initiate download of media item and when successfully downloaded, save the information that it is locally available in the media item management and dispatch event that the media item is now available (overwriting its current availability status) with a new local media locator.

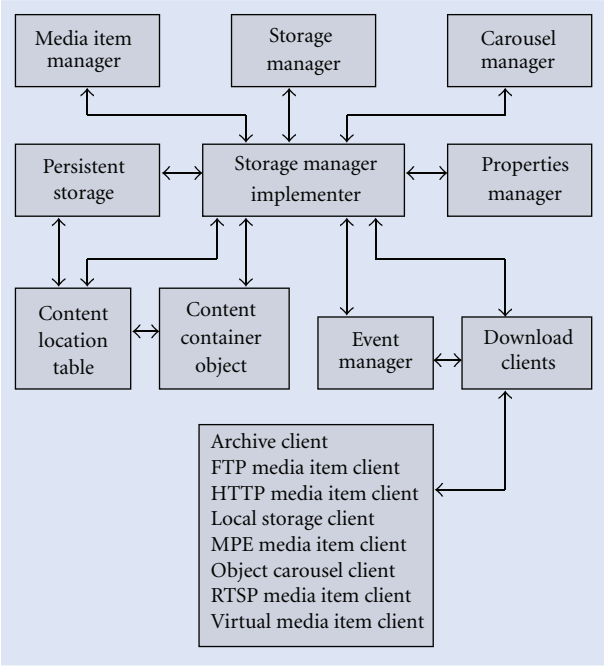

Figure 3: The distributed storage manager system components.

If media item has a DVB locator (not RTSP, ftp, or http), then the media item is normally downloaded with the appropriate download client. As usual, the media item management is notified that the media item has been downloaded. The media item might already be available because it has a second RTSP media locator. So if the media item has two media locators (RTSP and DVB), then the storage manager dispatches two storage events: one for the RTSP that is dispatched directly after the media item has been requested that indicates the content is available on the service delivery system (SDS). In parallel, the storage manager is downloading the content signaled in the DVB locator and when this download is finished, dispatches a second storage event for the same media item, saying that the media item is now available in the persistent storage, thus sparing the SDS from further requests.

3.4. Principle of Storage Manager System Operations. The storage manager (SM) is initialised at terminal boot up during the MHP environment initialisation. This is done during the system runtime. Therefore, the SM is integrated into the implemented MHP stack extending its current rudimentary persistent storage system. Since the storage manager implementer is the root software entity, it is responsible for the initialisation of all other SM software components and processes.

Upon initialisation the storage manager is loaded by the java runtime class loader. The storage manager implementer sets the service locator when the user selects a new service. An empty content location table (CLT) is initialised by the storage manager implementer; the CLT attempts to locate the persistent storage configuration file. This configuration file contains CLT entries of content that were persistently stored during the previous terminal operation. The file is then parsed recreating new content container objects (CCOs) of the previous persistent storage entries. The newly created CCOs are then added by the CLT.

The requests from the service description manager are handled by storage manager to make media items content available. The SM finds out where to get the requested media item from by analyzing the media locator and uses the appropriate module to access the content. By analyzing the request media item event, the SM gets the type of download and then passes the request to appropriate download client to download. Download client defines the interfaces for the internal storage manager modules. Each download client handles one media item transport mechanism; it also provides a common method to dispatch events and to call a suitable download client for a specific media item request.

Upon receiving a request from an external software subsystem, the request is passed to the storage manager implementer where the request is processed. If the request is for the absolute location of content, the relative filename is extracted from the request. The storage manager implementer then queries the CLT that performs a binary search of CCO. The CCO performs a matching function and validates the existence of the file using the stored absolute file location. If this search is successful, the corresponding CCO is extracted by the CLT and passed to the storage manager implementer. The storage manager implementer extracts the absolute file location and associate content file names and returns them to the external software sub-system that initiates the request. If no match is found or the validation test fails, an NULL is returned.

Since some form of content requires associated content (e.g., an HTML page may embed an image, therefore the location of this image is also required), the storage manager stores the mapping between content file names and associate content file names. If the request is to store a content item, the storage manager implementer extracts the relative filename and associate content file names from the request. The storage manager implementer then searches the absolute file location and relative filename. A new CCO is created by the storage manager implementer, which then instructs the CLT to store the CCO. If the request is to delete a content item, the storage manager implementer instructs the CLT to search and retrieve the target CCO. If found and the CCO validates the existence of the content file, the storage manager implementer instructs to remove the content with the corresponding absolute file location from the hard disk.

When a terminating signal is received by the MHP system, that is, the session had ended, the mechanism is provided by the storage manager implementer to allow the MHP system to inform the storage manager implementer of the termination. When this notification is received, the storage manager implementer instructs the CLT to create a backup of all persistently stored entries to the persistent storage configuration file.

3.5. Components and System Testing. To test all the internal components and input and output of the storage manager,

three testing environments were created to allow it to be tested in an integrated way. All the testing processes in the testing environments were as predicted and were successful. The three testing environments are presented as follows.

(i) Simulate MHP: a class known as "simulate MHP" was created to load all the storage manager components into a regular Java runtime environment. The aim of this environment is to enable all components to be tested. It is clear that since crucial parts of the distributed storage manager rely on MHP, not all components can be tested with this or will fully work even if they run.

(ii) Offair package (Stand-alone CAS without DVB input): an environment was created that relies on MHP but not on DVB input (off-air package). The aim of this environment is to set up the full workflow in the CAS. This especially allows testing communication between the components. Therefore, all components should implement the off-air functionality. The off-air storage manager class simulates downloading files by having all files available as local files and throwing storage events after some predefined schedules. It sends storage events that contain new locators for requested media items after some configurable time. So it is basically a translator of media locators. The information when which media item should arrive can be found in an XML configuration file located in the carousal directory. This file is parsed by the off-air storage manager config file parser that creates pairs of media items and time "stamps" to simulate that the media item is downloaded after a certain time. To schedule the requested media items (if they have been found in the configuration file), the media item arrival scheduler is used. This environment focuses on home media server functionalities rather than TV client functionalities. The prerequisite here is to have the MHP-RI available, and this means that this environment is set up on the terminal. This environment would allow testing of all (server) components in their "natural" environment and testing of storage manager functionalities that do not need DVB input (FTP, etc.). Such an environment would be independent of DVB streams, so that it does not have to rely on the creation of a DVB stream for the off-air testing.

(iii) Stand-alone version with DVB input: play prepared DVB transport stream from hard drive. The storage manager gets its DVB input not from the DVB channel but from a DVB stream stored on disk. This also includes video playback of main video on TV.

By using the three integrated testing environments, the following points were accomplished, and all the components testing results have passed:

(i) storage manager Impl: testing of the initialization of all software components and interfaces within storage manager;

(ii) carousel manager: testing the interfaces to the MHP API by downloading DSM-CC object carousel;

(iii) download client: the download of content onto the hard disk from various download clients and the addition of CCO within the CLT;

(iv) media item management: testing of media item downloaded to the CAS. Add/remove a media item to/from the persistent storage;

(v) content location table: testing of the insertion, retrieval, and deletion of CCO. Testing of access methods and "synchronized" blocks and testing of binary search performance.

(vi) content container object: testing of access methods for storing and retrieving file names, file locations, and associated content file names.

3.6. Networking and Services

3.6.1. Distribution and the Use of CORBA. One of the fundamental objectives of the storage manager was to allow multiple clients and multiple server processes in a distributed environment. For this purpose, CORBA was used as its object request broker (Figure 4). CORBA is a mechanism in software for normalizing the method-call semantics between application objects residing either in the same address space (application) or remote address space (same host, or remote host, on a network). All CORBA code is encapsulated in storage manager servant, storage manager server and storage manager client classes that work at the interface level of the calling classes. This allows the CORBA transport layer to be replaced by different CORBA implementations.

To translate overloaded class methods into separate CORBA methods, this work goes into an IDL (interface definition language) file, which autogenerates the CORBA code during its compile process. The JDK1.3+ comes with the idlj compiler, which is used in this case to map IDL definitions into Java declarations and statements. The storage manager servant, storage manager server, and storage manager client class codes must deal with the translation step between the overloaded methods and the differently named CORBA equivalents.

3.6.2. Naming Service. Each object is registered with the naming service (Figure 5) at instantiation. It operates in a manner analogous to a file directory structure, adding the names of objects to each node of a branch, where the first level branch is the "context" and the second level the component. When CORBA methods need to find an object, a request is sent to the naming service to get the object's location. The naming service is in charge and searches by name. Components must be uniquely named, otherwise the original component will be overwritten by subsequent ones.

3.6.3. Event Service. The event service usually, but not necessarily, runs on the same machine as the naming service, listens for event notifications from the server(s), and is responsible for getting these events across the network. The

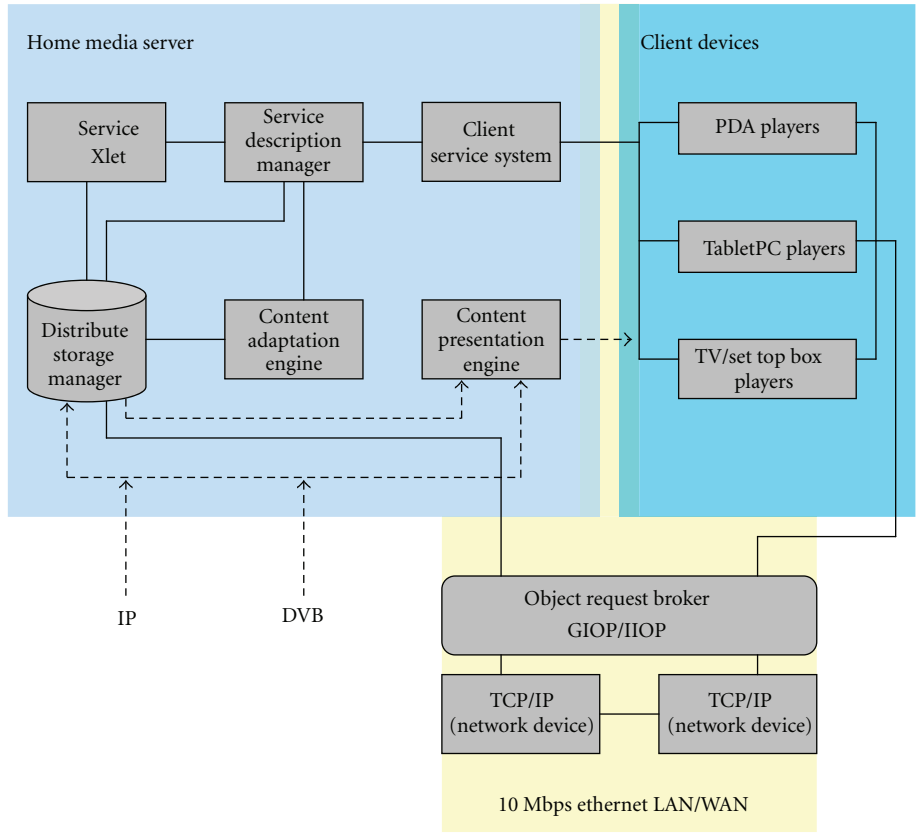

Figure 4: Structure of distributed platform for storage manager.

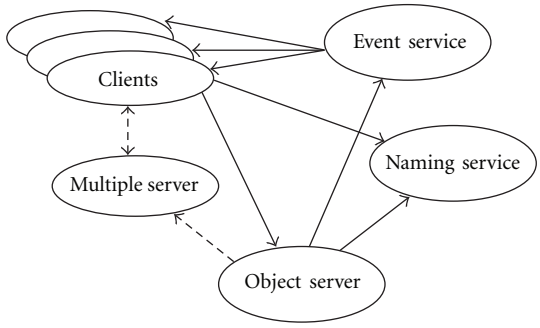

Figure 5: The underlying principle of multiple clients and servers.

event service uses the "push" model, that is, events are pushed out from the service to all registered clients. A concrete class on the object server sends a notify message over the event service to its companion on the client.

3.6.4. Object Server. This process is at the centre of the distributed storage manager architecture. The object server class contains the main entry point for the object server process.

Once the object server has an ORB, it can register the CORBA service. It starts by getting a reference to the root of the naming service. This returns a generic CORBA object and then transferred into a naming context object to register a CORBA service with the naming service.

For the client deployment, once a reference to the naming service has been obtained, it can be used to access the naming service and find the service such that the media item transfer service is found, and then the download media item method is invoked.

With the use of the distribution module (Figure 4), the request and response messages between the client and the server are exchanged through the 10 Mbps Ethernet LAN for the TV sets (set top box), for the portable device (PDA, Tablet PC) via a WLAN connection. A web server was constructed with the home media server, and the basis for communication (requests and responses) is HTTP. The storage manager instantiates and controls all components that run within the Tomcat environment. The object request brokers (ORBs) in both sides use the GIOP/IIOP (generic inter-orb protocol/internet inter-ORB protocol) protocol to exchange the messages. The client applications call the functions supported by storage manager to get the multimedia contents transparently from the object server, which is dispersed on the communication network.

3.6.5. Distribution Module Testing. To test the functionality of the distribution module, an application was developed on the basis of the distributed storage manager API. This application provides the interactive video retrieval service for video on demand (VOD). The distributed storage manager provides the client devices with various operations

to transport and control multimedia data in a location transparent manner. The client application and the server objects running on the heterogeneous system communicate interactively according to the VOD scenarios. The distribution module uses the storage manager API to obtain various multimedia data from the networks and parses and interprets the data encoded. The distribution module presented the interpreted data to the client devices and transferred the inputs from the client devices to the home media server. All the testing processes are as predicted and successful.

4. Conclusions

The distributed storage manager provides a solution for extending the MHP middleware, and it provides an implementation of interfaces to the MHP 1.2 API that permits other external software entities to make requests and receive responses from the storage manager. It provides the capability to retrieve content from multiple network interfaces utilizing various protocols provided by the MHP platform; this permits the downloading and extraction of file-based and streaming content delivered by different DVB transport protocols, namely, DSM-CC object carousel, DVB PES, MPEG-2 private sections, multicast UDP/IP via MPE, and various IP transport protocols. The presented functionalities of the distributed storage manager have an added value over the already provided functionalities by the most recent MHP version 1.2, also it supports for currently popular DVB-IPDC data carousels such as FLUTE to deliver files over the Internet or unidirectional systems from one or more senders to one or more receivers.

The distributed storage manager also provides multiple protocol support for initializing and downloading both streamed and file-based content. In addition, complex but optimum control mechanisms are provided to organize the storing and retrieval of content that are remained accessible to other multiple heterogeneous devices.

The distributed storage manager also organizes the downloaded content upon the hard disk to permit efficient retrieval of content file locations and provides access mechanisms to the downloaded content. It performs the task of organizing content that involves providing a structured database to store the mapping of absolute file locations, relative file names, and relative associate file names on the hard disk. This permits the distributed storage manager to service applications/external software components (i.e., content presentation engine) or decoders/players with the physical location of requested content. It also permits other external software entities to make requests and receive responses from the storage manager.

Compared to the storage management incorporated with multimedia home platform, the distributed storage manager permits a metadata search and retrieval of stored content based on a set of generated keywords; in addition, submission of multiple keywords is permitted. The resulting list will contain various content tiles with their matching score and allows a faster content presentation to the end user.

The distributed storage manager works with DVB-J (service Xlet) application that is executed via MHP. The storage manager enables, interoperability of various implemented MHP applications. The distributed storage manager API is a set of high-level functions, data structures, and protocols which represent a standard interface for platform-independent application software. It is object oriented and is based on the Java programming language. Furthermore, the API enhances the flexibility and reusability of the distributed storage manager functionalities.

A novelty distribution module embedded in the distributed storage manager was designed and implemented. This module is based on the CORBA distributed computing environment, and the interfaces that define the distribution services are in the form of OMG IDL. The distributed storage manager provides location transparency of service and the functionality to transport multimedia content from server to client or vice versa.

References

[1] P. Wolf, G. Durand, G. Kazai, M. Lalmas, and U. Rauschenbach, "A metadata model supporting scalable interactive TV services," in *Proceedings of the 11th International Multi-Media Modelling Conference (MMM '05)*, Melbourne, Australia, January 2005.

[2] J. Cosmas, A.n Lucas, K. Krishnapillai, and M. Akhtar, "Storage manger system for DVB terminals," in *Proceedings of the Telecommunications, Networks and Broadcasting(PG Net '01)*, EPSRC, Liverpool UK, June 2001.

[3] U. Rauschenbach, W. Putz, P. Wolf, R. Mies, and G. Stoll, "A scalable interactive service supporting synchronized delivery over broadcast and broadband networks," http://dea.brunel.ac.uk/project/savant/pub/IBC04-SAVANT-rauschenbach_et_al-updated.pdf.

[4] U. Rauschenbach, J. Heuer, and K. Illgner, "Next-Generation interactive broadcast services," http://www.rauschenbach.net/Publications/docs/wsdb2004_rauschenbach_etal.pdf

[5] B. Heidkamp, A. Pohl, and U. Schiek, "Demonstrating the feasibility of standardized application program interfaces that will allow mobile/portable terminals to receive services combining UMTS and DVB-T," *International Journal of Services and Standards*, vol. 1, no. 2, pp. 228–42, 2004.

[6] A. Centonza, T. J. Owens, J. Cosmas, and Y. H. Song, "Differentiated service delivery in cooperative IP-based broadcast and mobile telecommunications networks," *IMA Journal of Management Mathematics*, vol. 18, pp. 245–267, 2007.

[7] Y. Zhang, C. H. Zhang, J. Cosmas et al., "Analysis of DVB-H network coverage with the application of transmit diversity," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 568–577, 2008.