UPnP 기반 유비쿼터스 가상저장공간 설계

강보영*, 남영진**, 김성률*, 서대화*
*경북대학교 임베디드소프트웨어협동연구센터
**대구대학교 컴퓨터·IT 공학부
e-mail : {bykang,srkim,dwseo}@escrc.re.kr, yjnam@daegu.ac.kr

Design of UPnP-based Ubiquitous Virtual Storage

Bo-Young Kang*, Young Jin Nam**, Sung-Ryeul Kim*, Dae-Wha Seo*
*Embedded Software Cooperative Research Center, Kyungpook National University
**School of Computer & Information Technology, Daegu University

요 약

디지털 기술의 발전으로 다양한 컨버전스 제품들이 출시되고 있으며 컨텐츠의 디지털화가 가속화됨에 따라 대용량의 멀티미디어 데이터를 편리하게 분산 저장·관리 및 공유할 수 있는 기술 개발이 요구되고 있다. 본 논문에서는 이러한 요구 사항을 반영한 UPnP를 이용한 유비쿼터스 가상저장공간에 대해 설계 및 구현한다. UPnP를 이용한 유비쿼터스 가상저장공간은 유비쿼터스 기술을 기반으로 정보가전기기 내에 존재하는 저장장치를 UPnP를 이용하여 하나의 큰 가상저장공간을 구축하고, 가상저장공간 내에 멀티미디어 파일을 힌트정보와 함께 저장함으로써 대용량의 멀티미디어 파일을 보다 지능적으로 탐색할 수 있는 기법을 제공한다.

1. 서론

디지털 기술이 점차 컨버전스화 및 유비쿼터스화 방향으로 진화됨에 따라 언제, 어디서나 개인화된 디지털 컨텐츠를 즐길 수 있는 디지털 네트워크화가 이루어지고 있으며, 컨텐츠의 디지털화가 가속됨에 따라 멀티미디어 데이터와 같은 디지털 컨텐츠들의 데이터량 또한 기하급수적으로 증가하고 있다. 이러한 환경하에서 사용자들은 문서, 그림, 오디오, 비디오 등 다양한 형태의 컨텐츠들을 공유하기를 원하고 있을 뿐만 아니라 멀티미디어 컨텐츠 활용도 역시 증가하고 있는 추세이다. 따라서 대용량의 멀티미디어 컨텐츠의 보다 효과적인 저장 및 공유 기술의 필요성이 존재한다. 이러한 필요성을 만족시키기 위하여 기존에 다양한 연구들이 진행되고 있으나 대부분의 연구들이 중앙에 고가의 대용량 멀티미디어 서버를 둠으로써 중앙서버로 부하가 집중되는 현상이 발생하며 신뢰성과 확장성이 떨어진다는 문제점을 안고 있다 [1-2].

본 논문에서는 이러한 문제점을 해결하기 위해 현재 널리 사용되고 있는 다양한 종류 및 크기의 스토리지를 UPnP 기술로 통합하여 홈네트워크 또는 유비쿼터스 단말기에서 스토리지의 특성에 구애 받지 않고 대용량 멀티미디어 파일을 분산 저장 및 관리하며, 3-Any(anytime, anywhere, any device)를 만족하는 UPnP를 이용한 유비쿼터스 가상저장공간에 대해 설계 및 구현한다. 그리고 사용자가 원하는 멀티미디어 데이터를 효율적으로 찾는 검색 알고리즘을 제시한다.

2. 배경지식

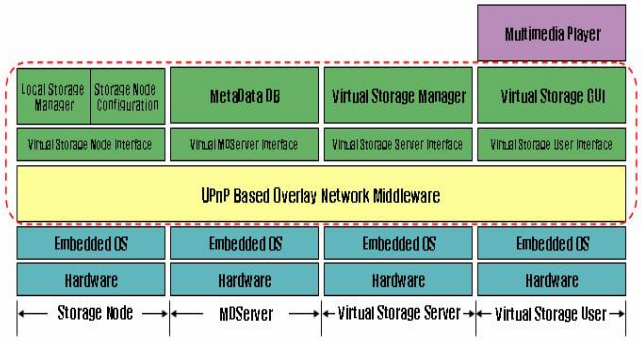
UPnP(Universal Plug and Play)는 PC, 주변장치, 지능형 가전제품, 무선 장비 등과 같은 장치들을 네트워크에 접속시켰을 때, 인터넷과 웹 프로토콜을 사용하여 서로를 자동으로 인식할 수 있도록 해주는 표준이다[3]. UPnP 이용 시 사용자가 어떤 장치를 네트워크에 추가하면 그 장치는 스스로 구성을 완료하여 TCP/IP 주소를 받고, 다른 장치들에게 자신의 존재를 알리기 위해 인터넷 HTTP에 기반을 둔 발견 프로토콜을 사용하게 된다. 예를 들어, 현재 네트워크에 접속되어 있는 카메라와 프린터가 있고 그 프린터를 통해 사진을 출력하려고 할 때, 카메라의 단추를 누르면 카메라가 "발견 요청" 신호를 네트워크에 보냄으로써 이용 가능한 프린터가 네트워크상에 있는지 찾도록 할 수 있다. 그러면 그 신호를 받은 프린터는 자신의 위치를 URL의 형태로 카메라에게 응답하게 된다. 카메라와 프린터는 공용의 언어로 XML을 사용하거나 프

로토콜 협상을 통해 어떤 방식으로 의사소통을 할 것인지를 정할 수 있게 된다. 의사소통을 위한 공용언어가 결정되면, 카메라는 프린터를 제어하고 선택된 사진을 출력할 수 있게 된다.

UPnP는 컨트롤 포인트 그리고 장치와 이들의 서비스를 기초 구성요소로 가진다. UPnP 장치는 우리가 일상에서 볼 수 있는 카메라, 프린터 등의 장치 내부에 UPnP 기능이 내장된 것을 말한다. 서비스는 가장 작은 제어 단위로서 장치가 사용자에게 제공하며 상태 테이블, 컨트롤 서버, 그리고 이벤트 서버로 구성된다. 상태 테이블은 상태 변수를 통하여 서비스 상태를 모델링하고, 컨트롤 서버는 액션 요청을 받아 이를 실행하며 상태 테이블을 갱신한다. 이벤트 서버는 자신의 상태가 바뀔 때, 컨트롤 포인트에게 이를 이벤트로써 알리는 기능을 한다. 컨트롤 포인트는 홈 네트워크에 연결된 장치들을 감지하고 제어하는 기능을 한다. 장치를 검색하여 서비스 정의와 서비스 목록을 가지며, 서비스의 액션을 실행시키고 장치로부터 이벤트를 받는 기능을 한다.

3. UPnP 기반 가상 저장 공간 설계

UPnP 기반 가상저장공간은 정보가전기기 내에 존재하는 다양한 종류의 저장장치들을 UPnP 를 이용하여 하나의 큰 가상저장공간 형태로 제공하고 멀티미디어 데이터를 분산시켜 저장·관리 및 공유하는 서비스를 제공하는 시스템이다. 또한, 가상저장공간 내에 멀티미디어 파일을 힌트정보와 함께 저장함으로써, 대용량의 멀티미디어 파일들을 보다 지능적으로 탐색할 수 있는 기법을 제공한다.

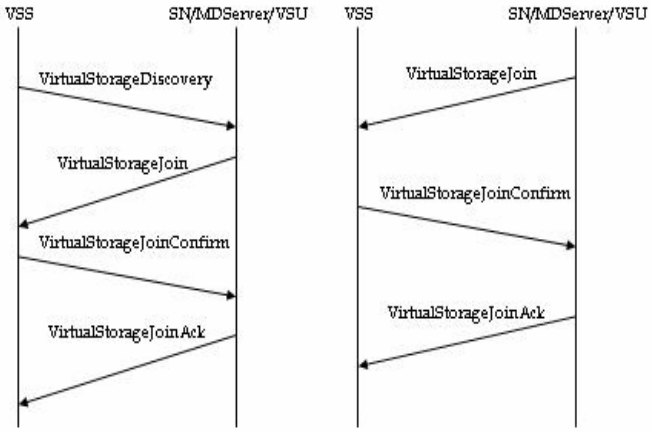


(그림 1) UPnP 기반 가상저장공간 구조

(그림 1)과 같이 UPnP기반 가상저장공간은 Storage Node(SN), MD Server, Virtual Storage Server(VSS), 그리고 Virtual Storage User(VSU)로 구성되어있다. 각 컴포넌트는 임베디드 리눅스를 기반으로 하고 있으며 가상 네트워크 구성을 위한 미들웨어로서 UPnP를 이용하고 있다. Storage Node는 항상 네트워크에 연결된 장치로서 VSS의 요청에 따라 VSS로부터 수신한 데이터를 256KB의 오브젝트 단위로 자신의 로컬 메모리 영역에 저장·관리하거나 데이터 오브젝트를 읽어서 송신하는 기능을 가진다. MDServer는 가상저장공간에 저장될 파일의 파일 ID, 파일 이름, 오브젝트 ID 리스트 및 부가적인 정보와 같은 정보를 관리하는 컴포넌트로 VSS의 요청에 따라 키워드를 기반으로 파일에 대한 정보를 검색하여 검색된 결과를 VSS에게 제공한다. Virtual Storage Server는 네트워크 내에 존재하는 SN의 로컬 메모리 영역을 하나의 가상 저장공간으로 구축 및 관리하는 컴포넌트로서 크게 세가지 기능을 가진다. 첫째, VSS는 유비쿼터스 가상 저장공간에 접속한 모든 SN, MDServer, VSU의 상태를 관리하는 가상저장공간 구축 및 유지· 관리한다. 둘째, VSS는 파일의 쓰기, 읽기, 삭제 및 캐싱 기능을 가진다. 즉, VSS는 VSU가 쓰기를 요청한 파일을 수신하여 256KB의 스트라입 유닛 단위로 분할한 뒤 자신이 관리하는 SN에 저장하고 VSU의 요청에 따라 SN으로부터 파일을 읽거나 삭제하기도 한다. 또한, VSU에 의해 자주 요청되는 파일을 캐싱함으로써 응답시간을 줄여 준다. 마지막으로, VSS는 보안 기능을 제공한다. Virtual Storage User는 가상저장공간으로부터 원하는 파일을 검색하여 파일을 읽거나 자신의 로컬 메모리 영역에 저장된 데이터를 가상저장공간에 저장 관리 할 수 있는 장치이다.

3.1. 오버레이 네트워크 미들웨어

VSS, SN, MDServer, VSU들이 서로 자동으로 인식하고 물리적인 위치에 투명한(location-transparent) 데이터 입출력을 제공하기 위해서, UPnP를 기반으로 한 오버레이(Overlay) 네트워크 미들웨어 계층을 구성한다[4].



(그림 2) 오버레이 네트워크 미들웨어 계층 구축 과정

(그림 2)는 오버레이 네트워크 미들웨어 구축 과정을 나타낸 것이다. VSS는 UPnP 컨트롤 포인터로 동작하며 SN, MDServer, VSU는 UPnP 디바이스처럼 동작한다. 가상저장공간 구축 방법은 크게 두 가지로 나눌 수 있다. VSS가 나중에 가상저장공간에 접속하는 경우와 SN, MDServer, 혹은 VSU가 나중에 가상저장공간에 접속하는 경우이다. 이미 SN, MD-Server, 또는 VSU는 네트워크에 존재하고 VSS가 나중에 네트워크에 접속하면 VSS는 VirtualStorageDiscovery 메시지를 브로드캐스팅하여 현재 네트워크에 들어와 있는 SN, MDServer, VSU를 검색한다. VirtualStorageDiscovery를 수신한 SN, MDServer, 또는 VSU는 권한이 없는 사

용자가 가상저장공간에 접속하는 것을 방지하기 위해 디지털 서명한 인증서를 포함한 VirtualStorage-Join 를 VSS에게 전송한다. 이 메시지를 수신한 VSS는 인증절차를 통해 가상저장공간에 SN, MDServer 또는 VSU를 등록하고 VirtualStorageJoinConfirm 메시지를 전송한 후, VSS가 VirtualStorageJoinAck를 수신하면 오버레이 네트워크 미들웨어 구축이 완료된다. 반면 VSS 보다 SN, MDServer, 혹은 VSU가 나중에 네트워크에 접속할 경우에는 디지털 서명한 인증서를 포함한 VirtualStorge-Join 메시지를 브로드캐스팅함으로써 이를 수신한 VSS에 의해 가상저장공간이 구축된다[5].

3.2. 가상저장공간 파일 검색 기법

유비쿼터스 가상저장공간에 저장된 멀티미디어 파일들은 힌트정보를 통해서 검색이 가능하다. 이러한 서비스를 제공하기 위해 가상저장공간에 파일을 저장할 때 저장될 파일에 대한 부가 정보를 함께 저장한다.

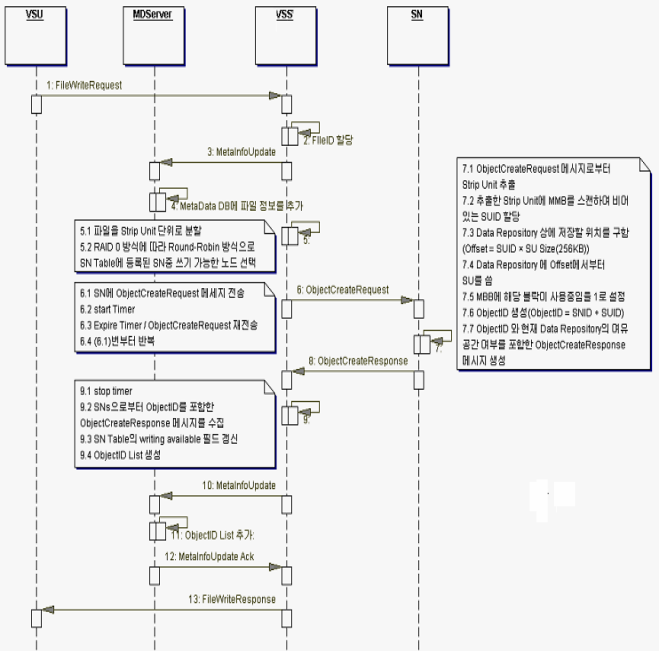
```
<?xml version="1.0"?>

<ADDITIONAL_INFO>
        <MUSIC>
                <TITLE>Love Actually</TITLE>
                <SINGER>MC the MAX</SINGER>
                <SONGWRITER>MC the MAX</SONGWRITER>
                <COMPOSER>MC the MAX</COMPOSER>
                <ALBUMINFO>3rd - Solitude Love</ALBUMINFO>
                                .
                                .
                                .
        </MUSIC>
</ADDITIONAL_INFO>
```

(그림 3) 부가 정보 표현 방법

(그림 3)은 가상저장공간에 저장되는 파일의 부가정보 저장 방법을 나타낸 것이다. 파일에 대한 부가정보는 확장성이 용이한 XML을 이용하여 표현하였으며 이 정보들은 VSS에 의해서 MDServer에 자동 등록되어 DB를 구축하게 된다. MDServer는 각각 파일에 대하여 File ID, File Name, ObjectID List, 부가정보 등 4가지의 필드를 기반으로 DB를 구축한다. VSU는 가상저장공간에 저장된 파일을 검색하여 읽기를 원할 경우 키워드를 입력하면 해당 키워드를 만족하는 파일 리스트(FileName+FileID)를 검색하여 사용자에게 전달하고 그 중 원하는 파일을 선택할 시에 파일의 ObjectID List를 참고하여 가상저장공간으로부터 파일을 읽어온다.

3.3. 가상저장공간의 파일 입출력

유비쿼터스 가상저장공간은 대용량의 파일 전체를 하나의 서버에 저장하여 관리하는 것이 아니라 256KByte의 Strip Unit 단위로 분할하여 가상저장공간에 접속 중인 SN에 분산 저장 관리한다.



(그림 4) 파일 쓰기 과정

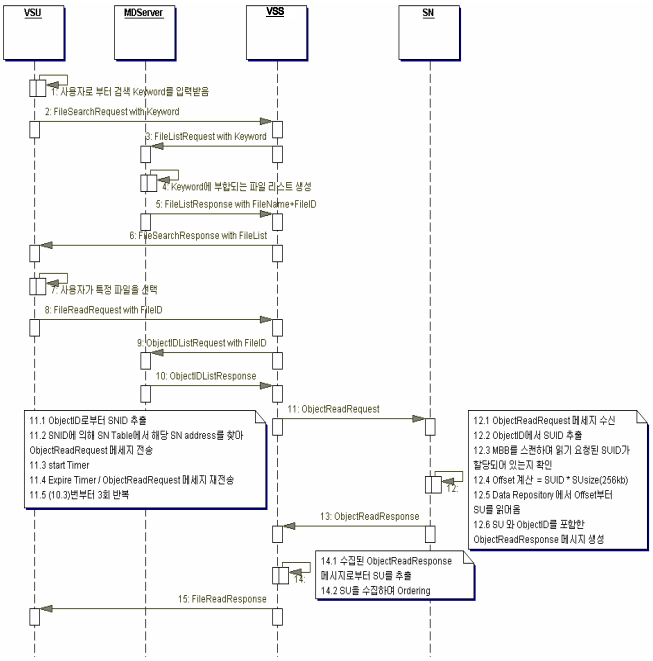
(그림 4)는 가상저장공간에 파일을 쓰는 과정을 나타낸 것이다. VSU가 가상저장공간에 파일을 쓰기 위해 VSS에게 파일의 부가 정보와 함께 파일 내용을 포함한 FileWriteRequest을 전송한다. 이 메시지를 수신한 VSS는 파일 내용과 파일에 대한 정보를 추출하여 FileID를 할당한 뒤 MDServer에 FileID와 부가정보를 등록한다. 파일은 256KB의 스트라입 유닛 단위로 분할하여 그 데이터와 SUID(Strip Unit ID)로 구성된 오브젝트를 라운드로빈 방식으로 SN Table에 등록된 SN중 쓰기 가능한 노드 선택하여 각각의 오브젝트들을 포함한 ObjectCreateRequest를 SN에게 전송한다. 이 메시지를 수신한 SN은 메시지로부터 SNID 추출하여 자신의 메시지가 맞는지 확인하고 오브젝트로부터 스트라입 유닛을 추출 자신의 로컬 메모리 영역에 저장한다. 이때 현재 SN의 로컬 메모리 영역에 저장 가능한 공간을 관리하는 MMB(Memory Map Block)를 스캔 하여 추출한 Strip Unit에 비어있는 SUID 할당하고 아래의 수식을 통해 자신의 로컬 메모리 영역에 저장할 위치를 구한다.

- $Offset = SUID \times SU\ Size(256KB)$

SN은 Strip Unit 을 저장한 뒤 MBB에 해당 블락을 1로 설정함으로써 사용중임을 표시한다. 그리고 ObjectID(SNID + SUID)를 생성하여 VSS에게 Object-CreateResponse 메시지 전송한다. 각각의 SN으로부터 ObjectCreateResponse를 수신한 VSS 는 메시지로부터 ObjectID를 추출 및 수집하여 MDServer의 해당 FileID에 대해 ObjectID 리스트를 등록 요청 하고 SN Table의 writing available 필드 갱신 한다. 이 과정들이 모두 끝나면 VSS는 VSU에게 FileWriteResponses를 전송함으로써 파일 쓰기 과정을 완료한다.

(그림 5)는 가상저장공간에 파일을 읽어 오는 과정을 나타낸 것이다. VSU가 유비쿼터스 가상저장공간으로부터 파일을 읽어오기 위해 VSS에게 키워드를 포함

한 FileSearchRequest을 전송한다. 이 메시지를 수신한 VSS는 MDServer에게 키워드를 만족하는 파일 리스트를 요청하고 MDServer는 키워드를 만족하는 파일 리스트를 검색하여 VSU에게 전달한다. VSU는 이 파일 리스트 중 만족하는 하나의 파일을 선택하여 VSS에게 파일을 읽어 줄 것을 요청한다. VSS는 MDServer에게 해당 파일의 ObjectID 리스트를 요청하고 VSS는 SNID와 SUID로 구성된 ObjectID로부터 SNID를 추출하여 각각의 SN에게 ObjectReadRequest를 전송한다. ObjectReadRequest를 수신한 SN은 SUID를 추출하여 파일이 저장된 위치를 계산한다. SN은 Offset으로부터 Strip Unit 크기만큼 데이터를 읽어 VSS에게 전달하고 VSS는 이 오브젝트를 VSU에게 전달한다.

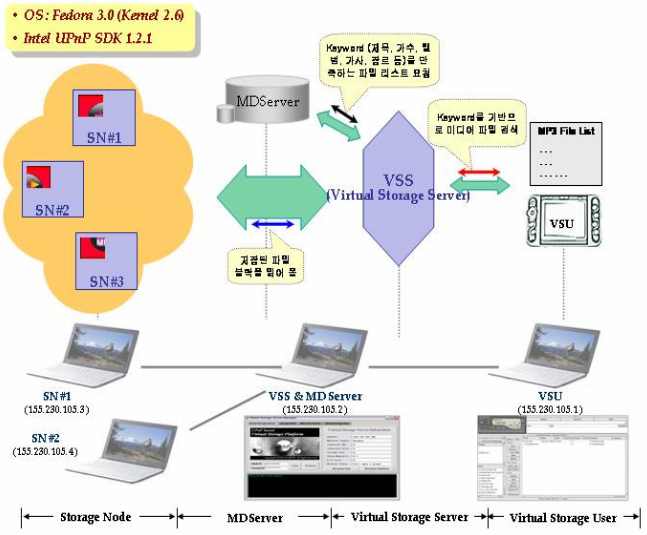


(그림 5) 파일 읽기 과정

3.4. 시스템 적용 모델

(그림 6)은 본 시스템을 적용한 시스템 적용 환경과 UPnP를 이용한 유비쿼터스 가상저장공간의 파일 읽기, 쓰기 및 삭제 기능 중 키워드를 기반으로 미디어 파일을 검색하여 재생하는 과정을 나타낸 것이다.

멀티미디어 플레이어가 탑재되어 있는 VSU는 가상 저장 공간에 저장되어 있는 MP3 파일을 재생하기 위하여 가수명, 작사명, 작곡가명, 앨범정보 및 가사정보 등과 같은 키워드를 기반으로 원하는 파일을 검색한다. MDServer로부터 키워드를 만족하는 모든 파일 리스트를 전달 받은 VSU는 검색된 파일 리스트 중 재생하기 원하는 파일을 선택한다. 선택과 동시에 VSS는 MDServer로부터 해당 파일의 ObjectID 리스트를 얻어오고 각각의 SN로부터 오브젝트를 읽어온다. 이렇게 수집된 오브젝트는 VSU에게 전송되어 VSU의 멀티미디어 플레이어에 의해서 스트리밍 데이터 형태로 재생된다.



(그림 6) 시스템 적용 환경 및 시나리오

4. 결론 및 향후 연구과제

본 연구에서는 대용량의 멀티미디어 파일을 정보가전기기 내에 존재하는 저장공간에 효율적으로 분산 저장·관리 및 공유 할 수 있는 UPnP기반 유비쿼터스 저장공간을 설계하였으며, 직관적으로 멀티미디어 파일/컨텐츠를 검색할 수 있는 서비스를 제공하였다. 본 연구에서는 모든 정보가전기기들이 항상 on이 되어있다는 가정을 한 반면에, 향후 연구에서는 정보가전기기들의 on/off 상태의 주기성을 분석하여 효과적으로 파일을 분산 저장하고 공유할 수 있는 기술이 개발되어야 한다. 또한, 정보가전기기에서 UPnP를 지원하지 않을 수 있다는 문제점이 존재하며, 이를 해결하기 위해 UPnP, Jini, HAVi 등의 다양한 제어 미들웨어를 지원할 수 있는 통합형 미들웨어가 개발되어야 할 것이다.

참고문헌

[1] 박상현, 손재기, 박창원, “능동형 미디어 스토리지 플랫폼 연구,” 한국정보통신설비학회 하계학술대회논문집, 2004년

[2] 박종인, 김문정, 엄영익, “분산 멀티미디어 서비스 환경에서 이동에이전트 기반의 검색 메커니즘,” 한국인터넷정보학회 춘계학술발표논문집, 2002년

[3] UPnP™ Device Architecture v1.0.1 Draft, http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf

[4] Intel Software for UPnP Technology, www.intel.com/technology/upnp

[5] Open Source Linux SDK for UPnP Devices 1.2.1, http://upnp.sourceforge.net/#downloads