파일 공유를 위한 분산 파일 서버 설계 및 구현

박주영* · 고석주*· 강신각*

*한국전자통신연구원

Design and Implementation of Distributed File Transfer Server

Juyoung Park* · Seok Joo Koh**· Shin Gak Kang*

*Electronics and Telecommunications Research Institute

E-mail : jypark@etri.re.kr

요 약

파일 공유를 위한 방법으로 FTP, NFS나 최근 P2P기술 등이 많이 제안되었지만, 현재까진 아직 FTP를 많이 사용하고 있다. 본 논문에서는 거의 기존 파일 전송 서비스 환경의 변경을 하지 않으면서도 고용량의 데이터를 보다 효율적으로 공유할 수 있는 방법으로써 Distributed File Server(DFS)방식을 제안한다. DFS의 특징은 기존 서버 클라이언트 방식의 FTP 환경을 변경하지 않고, 단지 다수의 서버들과 클라이언트 사이에 DFS를 설치함으로써 클라이언트에게 고용량 파일 저장 장소를 가진 FTP서버를 제공할 수 있다는 것이다. RedHat7.2 상에서 DFS 기능의 검증을 위한 프로토타입 구현을 통해 최근 대두되고 있는 고용량 데이터들로 인한 파일 시스템의 부족등의 문제점을 해결할 수 있음을 보였다.

ABSTRACT

Although there are a lot of file sharing mechanisms in the Internet such as NFS and P2P, FTP is one of the most common mechanisms used until now. In this paper, we propose a more enhanced file sharing mechanism, the Distributed File Server (DFS) method. The most remarkable feature of DFS is that it can be simply adopted in the current Internet FTP services by locating a DFS node between existing FTP servers and clients without changing any existing file transfer environment. To verify the proposed mechanism we implemented a DFS prototype on RedHat 7.2 system and we finally show that it can reduce the file storage problem caused by big-sized multimedia data.

키워드

enhanced FTP service, distributed file server

I. 서론

최근 인터넷에서는 서비스의 종류가 하드웨어와 사용자의 서비스 요구의 고급화로 인하여 멀티미디어 서비스가 주를 이루고 있다. 그런데 서비스를 제공하기 위해 해결해야할 점들 중의 하나는 고용량의 데이터의 공유 방법을 들 수 있다. 해결책으로써 파일 서버의 업그레이드나 구형 PC 자원을 공유하여 프로세싱 능력뿐만 아니라 저장 공간을 증대시킬 수 있는 분산 서버 시스템에 관한 연구도 많이 진행되어 왔다. 파일 공유 방법으로써 FTP[1], NFS[3] 등의 기술과 최근 P2P[4] 기술 등의 공유 메커니즘이 많이 제안되었지만, 아직까지 인터넷에서 파일을 주고받는 가장 대표적인 경우는 FTP이다[2].

본 논문에서는 고용량 멀티미디어 데이터의 효율적인 공유 방법과 이를 적용할 때 기존 서비스 환경의 변경을 최소화할 수 있는 방법을 고려하였다. 제안하는 Distributed File Server(DFS)의 특징은 기존 FTP 환경을 바꾸지 않고, 단지 DFS를 설치함으로써 사용자에게 광대한 데이터 저장 장소를 제공할 수 있다는 것이다. 본 고에서는 DFS의 간략한 소개와 제안 배경을 2절에 언급하였다. 3절에는 FTP를 비롯하여 파일

을 공유하는데 사용되는 관련 연구를, 4절부터 7절까지는 DFS 제안을 위한 요구사항, 설계, 구현 및 동작 검증을 실었으며, 마지막 8절에 본 논문의 결론 및 향후 연구에 관해 기술하도록 한다.

II. DFS 소개

멀티미디어 서비스가 증가함에 따라 자연히 데이터의 크기가 증가하게 되었으며, 이러한 파일들로 인하여 하나의 FTP서버가 관리하려면 고용량의 하드디스크 레이드나 CD 주크박스등을 사용한다. 만일 이웃 파일 서버의 디스크 용량을 공유하는 경우에 있어서는 NFS등을 이용하여 디스크 마운트등을 해야 하는데, 이를 위해선 파일을 export하거나 마운트하려는 시스템 관리자의 허가를 받아야 한다. [3]

본 DFS는 시스템 관리자가 아닌 일반 사용자들이 자신이 공유하고 싶은 파일들을 기존의 FTP기술을 추가적인 수정 없이 이용할수 있도록 하는 방법이다.

기본적인 메커니즘은 FTP를 이용하여 파일을 받거나 올리기 위해 사용자들이 실제 파일이 저장되어 있는 파일 서버에 접속한 후, 파일 검색과 다운로드를 하는 대신, DFS 서버에 접속하여 파일을 다운받는다. DFS서버는 클라이언트와 실제 파일 서버의 중간에 위치함으로써 사용자 계정, 디렉토리 리스트, 실제 파일 서버의 공유정보 및 연결 등에 관한 관리만을 수행한다.

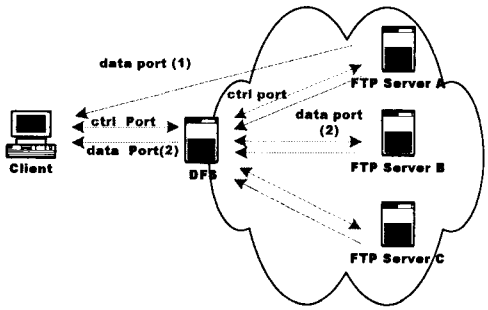

그림 1 DFS기본 개념

동작 방식은 우선 파일을 공유하려는 사용자들이 자신이 공유하려는 디렉토리를 DFS에 등록한다. DFS는 등록된 공유 디렉토리를 공유 정보에 따라 FTP 서비스를 수행한다. 파일을 다운로드하려는 클라이언트들은 그림 1에서 보는바와 같이 기존의 FTP 클라이언트 프로그램으로 DFS에 접속한 후 파일을 다운받는다. 이때 일반 FTP 클라이언트들의 다운로드는 실제 파일이 저장되어 있는 노드로부터 이루어지지만, 디렉토리 리스트의 브라우징은 실제 파일이 저장된 시스템이 아니라 DFS의 캐시 항목이 브라우징됨으로써 실제 파일 시스템의 디스크 엑세스를 크게 줄일 수 있다.

III. 관련연구

본 관련연구에서는 저용량의 구형 PC들을 서로 공유하여 프로세싱 능력뿐만 아니라 저장 공간을 모아 고성능의 서버로서 동작할 수 있는 기술을 사용하는 분산 시스템에 대한 연구보다는 인터넷 멀티미디어 서비스의 발달에 따라 데이터의 고용량에 따라 저장 공간을 보다 효율적으로 사용할 수 있는 방안에 관점을 맞추도록 한다.

전형적인 FTP서비스는 원격호스트와 데이터를 전송하기 위해 제안된 프로토콜로써, 각기 다른 시스템에서 데이터를 원활하게 전달할수 있는 메커니즘이다. FTP의 가장 큰 특징은 데이터 전달을 위한 연결과 연결 제어를 위한 연결이 분리되어 있다는 것이다. 다른 하나는 송수신 파일 시스템이 상이할 경우 이를 유연하게 접목시키는 전달 방식을 사용한다. 마지막으로 제어 연결은 서버측에서 패시브 오픈을 하고 클라이언트쪽에서 액티브 오픈을 하지만, 데이터의 전달은 패시브 혹은 액티브 오픈을 선택적으로 할 수 있다.

NFS(Network File System)는 유닉스 시스템에서 파일 공유를 위하여 가장 많이 사용되고 있는 파일 시스템 방식이다. 서버 관리자는 공유 대상 파일 시스템을 export 하고 클라이언트들은 이를 마운트하여 로컬 파일 시스템처럼 사용한다. 그런데 NFS는 파일 시스템 자체를 제공해줌에 따른 보안 문제등으로 인하여 시스템 관리자 차원의 주의 깊은 사용이 필요하다.

앞의 두 기술은 시스템 관리자가 파일 시스템을 관리하는데, 일반 사용자들은 점차 관리자의 검열이나 제약으로부터 자유롭게 파일을 공유하기를 원하게 되었다. 이러한 요구사항으로 발전된 기술이 최근 많은 관심을 가지게 된 P2P기술인데, 이 기술을 통하여 일반 사용자들은 자신의 파일을 서로 자유롭게 주고 받을수 있게 되었다. 특히 시스템 관리자의 검열 대상인 음악파일이나 동영상등의 공유에 많이 사용된다.[4]

이 기술의 특징으로는 파일의 위치를 알려주는 메커니즘을 통해 파일의 위치를 파악하고 그 위치로부터 직접 파일을 다운받을 수 있다. 일반적인 동작 방식은 클라이언트가 디렉토리 서버로부터 파일의 위치를 파악한 후 실제 파일을 저장하고 있는 호스트에 접근하여 파일을 다운받는다. 단점은 파일 위치를 알리거나 질의하기 위해 디렉토리 서버와 빈번한 통신을 해야 하며, 서버/클라이언트 동작을 동시에 수행할 수 있는 응용을 설치해야 한다는 불편함을 감수해야 한다.

IV. 요구사항

그런데 앞의 해결책들은 몇몇 해결해야할 사항들이 있다. 첫 번째 파일 시스템의 위치를 검색이 용이해야 한다. FTP나 NFS를 사용할 경우 어떠한 파일이 어떤 파일 시스템에 있는지를 쉽게 검색할 수 있지만, P2P 기술을 사용할 경우 원하는 파일이 저장된 디렉토리의 검색에 대한 고려가 있어야 한다. 두 번째, 시스템 관리자에 독립적으로 동작할 수 있어야 한다. P2P의 경우 일반 사용자가 임의적으로 파일을 공유할 수 있지만, NFS나 기존의 FTP방식을 사용할 경우 공유하려는 파일을 시스템 관리자가 일일이 설정해야만 한다. 마지막으로 디플로이할 때 비용이 저렴해야 한다. 가장 좋은 방법은 현재의 환경을 변경하지 않고 바로 사용할 수 있는 것이지만, 만일 변경해야할 경우 가장 최소한으로 환경을 바꿀 수 있어야 한다.

V. 설계

제안하는 DFS는 FTP를 사용하는 사용자는 변화된 환경에 대한 아무런 지식 없이 사용할 수 있도록 설계되었다. 특히 파일을 공유하려는 파일 서버의 사용자는 자신이 임으로 디렉토리 구조를 변경할 수 있다. 그림 2에 설계한 DFS 기본 메커니즘을 나타내었다. 그림 2의 FTP디렉토리 캐시는 매번 클라이언트의 LIST 명령에 대해 실제 파일 서버의 파일 시스템이 접근되지 않도록 실제 디렉토리에 대한 캐시를 저장한다.

DFS의 기본적인 동작 시나리오는 다음과 같다.

1) 파일을 공유하려는 사용자는 자신의 FTP계정과 공유 디렉토리 경로를 DFS서버에 등록한다.

2) DFS는 FTP 클라이언트의 접속 요청을 실제 파일 서버의 계정과는 별도로 자신이 관리하고 있는 계정을 통하여 수락을 결정한다.

3) FTP 클라이언트의 요청을 제어하며, 파일 전송에 대한 연결을 제어한다.

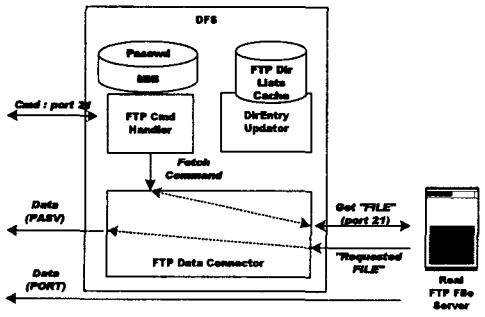

그림 2 DFS 메커니즘

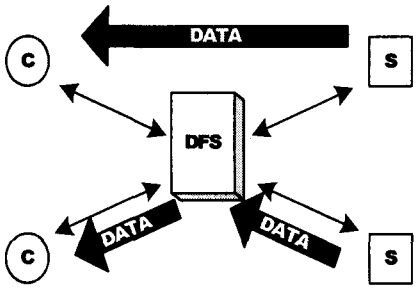

그림 3 DFS 데이터 전송 경로

4) 데이터 전송 경로는 그림 3에서 보이는 바와 같이 FTP 클라이언트가 액티브 데이터 연결(PASV명령)을 요청할 경우 DFS를 경유한 데이터를 전송하지만, 반대로 패시브 데이터 연결(PORT명령)을 요청할 경우 실제 파일서버로부터 데이터를 전달해준다.

5) 실제 파일서버와 DFS간의 제어 채널은 필요에 따라 연결 혹은 해지한다.

VI. 구현 및 시험

DFS기능을 검증하기 위하여 Linux RedHat 7.2환경에서 C를 이용하여 구현을 하였으며, 시험 환경은 그림 1과 같은 형태로 구축하였다. 동작의 검증을 위하여 기존의 FTP 파일 서버들을 서비스하는 DFS 서버에 리눅스 FTP 클라이언트와 윈도우의 ws_ftp를 사용하였다.

다음 그림 4는 DFS 서버가 파일 디렉토리를 마운트하기 위하여 실제 관리하는 파일 서버의 정보 설정 파일이다.

```
# DirEntry Module  (direntry.conf)
#local dir    remote dir info.
# read dir info
#/localdir    IPaddr loginID pass /remotedir
/home    129.254.165.150    jypark qwer1234    /home/jypark/dfs
# write dir info
#/localdir/incoming    IPaddr loginID pass /remotedir
```

그림 4 DFS 설정파일

여기서 파일에 저장되어 있는 서버의 아이디와 패스워드는 FTP 클라이언트가 검색하려는 디렉토리 정보가 DFS 서버의 캐시에 저장되어 있지 않을 경우, 해당되는 실제 FTP서버로 접근할 때 사용된다. DFS 캐시는 단지 디렉토리의 리스트만이 저장하며, 데이터 전달에는 관여하지 않는다. 그림 5에 DFS 시그널링 시퀀스를 나타내었다.

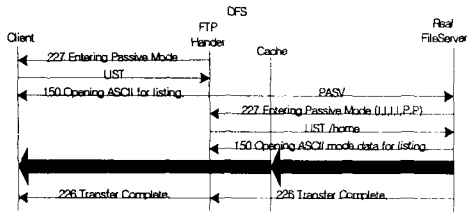

그림 5 DFS 디렉토리 리스트 캐싱

FTP클라이언트는 파일의 위치를 DFS 캐시를 통하여 검색하고, 데이터 전송을 위해 DFS는 다음과 같이 2가지 경우의 데이터 전달 방식을 지원한다.

1) 실제 파일서버로부터 직접 데이터 전달

FTP 클라이언트가 그림 6에서와 같이 PASV명령을 사용하였을 경우, 클라이언트는 DFS서버가 열은 데이터 채널을 통해 데이터가 전달되는 것을 가정하기 때문에, 이 경우 실제 파일서버로부터의 데이터는 DFS를 경유하여 클라이언트로 전달된다.

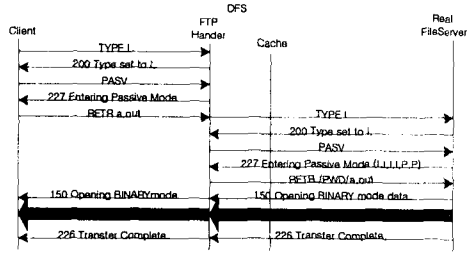

그림 6 DFS 데이터 전송 시그널링 (PASV)

2) DFS를 경유한 데이터 전달

만일 앞의 경우와는 달리 FTP 클라이언트가 그림 7에서 보인바와 같이 PORT명령을 사용할 경우, 클라이언트는 DFS서버로부터 데이터 채널이 연결되기를 가정하기 때문에, 이 경우 실제 파일서버로부터 데이터가 클라이언트로 전달된다.

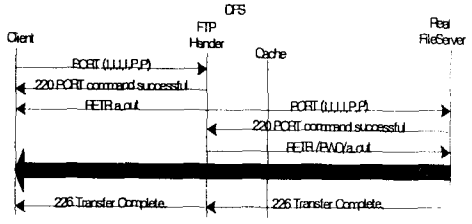

그림 7 DFS 데이터 전송 시그널링 (PORT)

구현된 DFS서버는 리눅스 응용으로 동작하며, 9090 포트에서 FTP요청을 수행하도록 구성하였다. 그림 8의 그림에서 보이는 것처럼 클라이언트 응용인 ws_ftp[5]가 9090 포트에서 서비스를 지원하는 DFS 서버에 접근하기 위해 서버 포트번호를 설정하였으며, 서버는 리눅스 콘솔에서 서비스를 수행하도록 하였다.

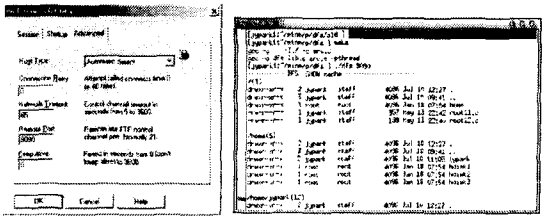

그림 8 ws_ftp 설정 및 DFS 동작화면

DFS를 이용하여 윈도우 응용과의 실험 화면을 그림 9에 로그화면과 함께 보였다. 로그화면에서는 DFS 서버에 접속한 로그와 함께 그림 9에 저장된 캐시의 내용이 리스트됨을 확인할 수 있었다.

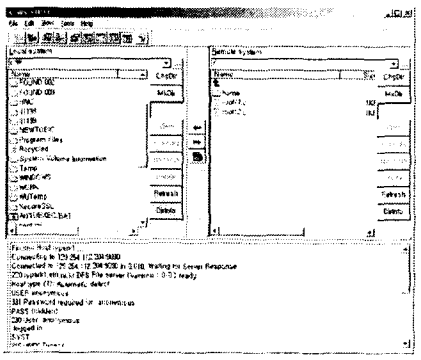

그림 9 DFS서비스 실행화면

VII. 결론

최근 멀티미디어 데이터의 증가에 따라 파일의 크기 또한 많이 증가됨에 따라 더욱 필요하게 되는 이런 추세에 따라 어느 하나의 FTP서버가 자신의 디스크만을 사용하여 모든 필요한 파일을 저장하기 보다는 분산되어 있는 파일들에 대한 정보를 하나의 시스템에 위치시킬수 있도록 하였다. 본 제안으로 현재 FTP서비스에 별도의 변경없이 클라이언트들에게는 하나의 대형 FTP서버를 제공하고 다수의 파일 공유자들에게 공유 데이터를 손쉽게 관리할수 있도록 하였다.

본 제안 메커니즘은 널리 사용되고 있는 FTP 서비스와, NFS의 파일 공유방식 P2P의 디렉토리 검색 방식을 적절히 융화하여 제안한 메커니즘으로써, 각각의 장점을 최대한 부각시켰다. 본 제안 메커니즘이 실제 서비스 환경에 사용될 경우 고용량 데이터들을 하나의 파일 서버에 저장할 필요가 없기 때문에, FTP서비스를 제공할 때 파일 시스템의 부족등의 문제점을 해결할수 있으리라 사료된다.

DFS는 향후 속도등의 개선을 위한 캐싱 메커니즘과 데이터 전송 방식에 대한 자가설정(auto-configuration)

에 대한 연구가 진행될 계획이다.

참고문헌

[1] Postel, "File Transfer Protocol(FTP)", RFC959, Oct. 1985

[2] Braden. R., "Requirements for Internet Hosts -Application and Support", STD 3, RFC1123, Oct. 1989.

[3] Sun Microsystems, Inc., "NFS: Network File System Protocol Specification", RFC1094, Mar. 1989.

[4] Andy Oram , "Peer to Peer - Harnessing the Power of Disruptive Technologies", O'Reilly & Associates, 2001

[5] ws_ftp, http://www.ipswitch.com/Products/WS_FTP/