웹서비스를 이용한 사용자 수준 네트워크 파일시스템 프레임워크

김도형○ 박현희 양승민
숭실대학교 컴퓨터 학과
keedi○@naver.com darklight@realtime.ssu.ac.kr yang@computing.ssu.ac.kr

User-Level Network File System Framework with Using Web-Service Protocol

Keedi Kim○ Hyeon-hui Park Seung-min Yang
Department of Computing Graduate School, Soongsil University

요 약

네트워크 파일시스템은 원격의 서비스와 자원에 대해 로컬의 그것과 같은 투명성을 제공하기 때문에 널리 사용된다. 네트워크 파일시스템을 제작하거나 수정하는 일에는 운영체제의 커널과 통신 프로토콜에 대한 고려가 필요하므로 복잡하고 많은 시간이 걸린다. 기존의 파일시스템 프레임워크를 사용할 경우 운영체제의 커널 모듈에 대한 고려는 줄일 수 있지만 여전히 통신 프로토콜에 대한 고려는 필요하다. 따라서 네트워크 파일시스템의 빠른 제작을 도와주는 네트워크 파일시스템 프레임워크가 필요하다.

본 논문은 웹서비스 프로토콜인 SOAP과, 사용자 수준 파일시스템 프레임워크인 FUSE를 이용해 사용자 수준 네트워크 파일시스템 프레임워크인 NFSF를 설계하고 구현한다. NFSF는 단일 클라이언트 모듈과 서버 상위모듈, 서버 하위모듈의 3단계 계층을 두고, 중요 파일시스템 API를 서버 하위 모듈로 바인딩한다. 통신 프로토콜을 포함한 단일 클라이언트 모듈과 서버 상위 모듈을 프레임워크에서 제공하여 네트워크 파일시스템 제작 시 서버 하위 모듈만을 제작하게 함으로써 네트워크 파일시스템 제작과정을 `로컬 저장 장치 자체에 대한 고려'로 축소시킨다.

1. 서 론

현재의 컴퓨팅 환경은 네트워크의 자원을 적극적으로 사용하는 네트워크 컴퓨팅 환경으로 발전하고 있다. 로컬 장치의 성능이 부족하거나, 또는 그렇지 않더라도 네트워크 상으로 제공하는 서비스를 받기 위해서 많은 장비들이 네트워크의 자원을 이용하고 있다. 이렇게 네트워크상의 자원을 이용할 경우 네트워크 파일시스템은 로컬 자원을 사용하는 것과 동일한 투명성을 제공한다.

이러한 네트워크 파일시스템이 제공하는 투명성을 기초로 네트워크 서비스에 적합한 네트워크 파일시스템을 사용한다면 서비스를 제공하기 용이해진다. 하지만 해당 네트워크 서비스에 적합한 네트워크 파일시스템을 제작하기 위해 고려할 사항은 복잡하다. 우선 네트워크 통신 및 프로토콜에 대해 이해해야하며, 다음으로 네트워크 파일시스템을 제작할 운영체제 커널 내부 파일시스템의 구조 및 동작을 이해해야한다. 이 두 부분을 설계하고 구현하는 것은 많은 시간을 요하는 작업이다.

현재 리눅스에서는 널리 쓰이고 있는 파일시스템 프레임워크가 몇 가지 존재한다. 이러한 파일시스템 프레임워크를 사용하면 파일시스템을 만들기 위해 운영체제 커널 하부 모듈을 설계하는 작업을 생략할 수 있다. 그렇지만 이들 프레임워크를 사용하더라도 네트워크 파일시스템을 제작하기 위해서는 여전히 통신 프로토콜을 정의해야하며, 이 프로토콜을 사용하여 데이터를 주고받기 위한 부분을 구현해야 한다. 또한 네트워크 파일시스템의 서버 드라이버와 클라이언트 드라이버를 모두 구현해야 한다. 그렇기 때문에 실제 저장장치를 접근하기 위한 부분을 제외한 나머지 고려사항을 네트워크 파일시스템 프레임워크로 제공해서 네트워크 파일시스템의 제작 과정을 단순하게 할 필요가 있다.

본 논문에서는 네트워크 파일시스템의 구현이 용이한 네트워크 파일시스템 프레임워크인 NFSF(Network File System Framework)를 제시한다. 본 논문에서 제안하는 NFSF는 서버와 클라이언트의 통신을 위해서 웹서비스 프로토콜을 사용하며, 기존의 파일시스템 프레임워크의 플러그인으로 동작함으로써 이식성과 확장성을 높인다. 또한 클라이언트 파일시스템 드라이버를 단순하게 단일화하여 NFSF에서 제공하는 한편, 서버 파일시스템 드라이버는 상부와 하부로 나누어 상부 모듈을NFSF에서 제공함으로써 네트워크 파일시스템의 개발 효율성과 유지보수 편의성을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 네트워크 파일시스템을 정의하고 구성요소를 알아본다.또한 제안하는 NFSF를 설계하기 위해 필요한 표준 웹서비스 프로토콜과 사용자 수준의 파일시스템 프레임워크를 소개하고 이 기술들을 적용하는 방법을 설명한다. 3장에서는 NFSF를 설계하고 구현한 후 프레임워크를 적용해서 네트워크 파일시스템을 제작한다. 4장에서는 다른 파일시스템 프레임워크와 비교하여 NFSF를 분석한다. 마지막으로 5장에서는 NFSF의 특징과 장점에 대해서 요약 정리한 후 향후 과제에 대해서 논한다.

2. 관련 연구

2.1 네트워크 파일시스템

네트워크 파일시스템은 원격에 위치한 파일시스템을 로컬 파일시스템처럼 이용할 수 있는 프로토콜로 Sun Microsystems에서 제작한 NFS[1]가 대표적인 예이다. 네트워크 파일시스템을 이용해서 원격의 파일시스템을 마운트 할 경우 상위 계층에서 마운트 지점 아래에 위치한 파일을 시스템 콜을 통해 접근하면 네트워크 파일시스템이 시스템 콜의 요청을 받아서 직접 네트워크로 파일을 수신하여 처리한 후 그 결과를 시스템 콜의 결과로 보낸다. 그렇기 때문에 네트워크 파일시스템을 이용하면 원격지의 파일을 읽고 쓸 수 있을 뿐만 아니라 원격지의 프로그램도 실행은 물론 관련 파일들을 연결하는 일도 자동으로 로컬의 파일들과 마찬가지로 처리된다.

2.2 SOAP

SOAP는 Simple Object Access Protocol의 약자로 분산 환경에서 웹서비스를 제공하기 위해 설계된 가벼운 프로토콜이다. 이것은 세 개의 부분으로 구성되어 있다. 메시지를 운송하는 XML-envelope와 XML로 프로그램 객체를 인코딩하는 규칙과 RPC를 수행하는데 필요한 협약으로 이루어져 있다[2,3,4,5,6]. SOAP의 명세는 매우 유연하며, 프로토콜의 확장도 지원한다. 현재 W3C (World Wide Web Consortium)의 권고안은 SOAP 1.2 버전이다[2].

2.3 FUSE

FUSE[7]는 AVFS(A Virtual File System)[8]을 지원하기 위해 시작한 프로젝트로 리눅스 환경에서 사용자 수준의 파일시스템 제작을 지원하는 프레임워크이다. FUSE는 간단한 API 라이브러리를 제공하며, 설치가 쉽고 사용자 공간과 커널 사이의 인터페이스가 매우 효율적이다. 또한 사용자 수준의 파일시스템을 지원하기 때문에 루트 권한이 없는 사용자도 FUSE 그룹 권한을 가진다면 파일시스템을 제어할 수 있기 때문에 사용자에게 파일시스템 접근 권한을 유연하게 제공할 수 있다. FUSE는 현재 리눅스 커널 2.4와 2.6에서 모두 안정적으로 동작하고 있다.

3. NFSF(Network File System Framework)

NFSF는 크게 두 부분으로 나눌 수 있다. 하나는 공통 모듈이며, 다른 하나는 세부 모듈이다. 공통 모듈은 NFSF가 제공하는 부분이며 세부 모듈은 NFSF가 제공하는 인터페이스를 이용해 프레임워크를 사용해서 파일시스템을 제작하는 개발자가 작성하는 부분이다. 공통 모듈은 클라이언트 파일시스템 모듈과 서버클라이언트 통신 모듈, 서버 파일시스템 상위 모듈로 구성된다. 세부 모듈은 서버 파일시스템 하위 모듈로 구성된다.

3.1 NFSF의 설계

본 논문에서 제안하는 NFSF의 목표는 원격으로 제공할 저장장치에 접근하는 방법을 구현하는 것만으로 네트워크 파일시스템 제작 과정을 완료하는 것이다.

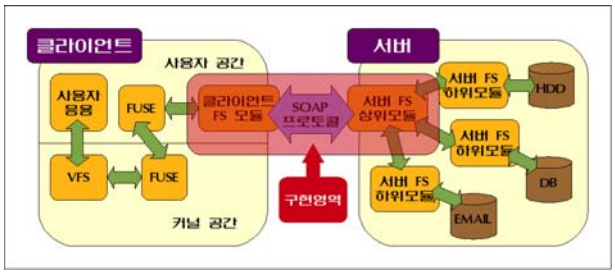

그림 1 NFSF를 적용한 파일시스템

그림 1은 NFSF가 서버와 클라이언트 사이에서 어느 계층에 존재하며, 어떻게 동작하는지 보여준다. FUSE 파일시스템 프레임워크의 플러그인 형태로 동작하기 때문에 제안하는 NFSF는 서버와 클라이언트 모듈 모두 사용자 수준에 존재한다.

3.1.1 클라이언트 파일시스템 모듈

NFSF의 한 부분인 클라이언트 파일시스템 모듈의 가장 큰 특징은 서버 측에서 제공하는 파일시스템에 상관없이 단일한 모듈로 존재한다는 점이다. 클라이언트 파일시스템 모듈은 FUSE에서 사용자 수준의 파일시스템을 구현하기 위해 제공하는 API 각각에 대응하고 있으며 이를 모두 바인딩해서 통신 채널을 통해 보낼 준비 작업을 한다.

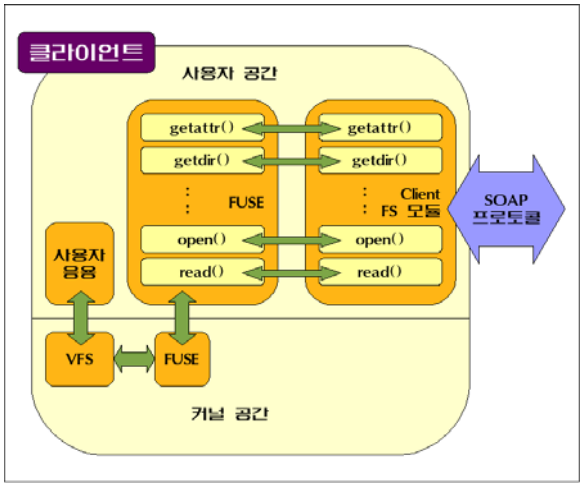

그림 2 NFSF: 클라이언트 파일시스템 모듈

그림 2는 NFSF를 사용했을 때의 파일 시스템에 사용자의 응용이 접근을 시도할 때의 과정을 보여준다. 우선 사용자의 응용이 FUSE를 통해 마운트 되어 있는 원격 자원을 접근하려고 하면, 해당 파일시스템 시스템 콜을

호출한다. 리눅스에서는 이 시스템 콜 호출은 VFS에게 전달되며 VFS는 이 시스템 콜이 FUSE 파일시스템을 사용하기 위한 것임을 판단한 후 커널 공간의 FUSE 커널 모듈에게 시스템 콜에 해당하는 동작을 요청한다. FUSE 커널 모듈은 VFS로부터 요청받은 시스템 콜을 다시 사용자 공간의 FUSE 모듈에게 요청한다. 여기까지는 일반적인 FUSE를 사용한 파일시스템과 동일한 방식으로 동작한다.

클라이언트 파일시스템 모듈은 사용자 공간의 FUSE 모듈의 요청을 받아 SOAP 프로토콜을 사용해 서버 측으로 보낼 준비를 한다. 클라이언트 파일시스템 모듈은 FUSE 모듈의 요청을 서버로 보내기 위한 스텁(stub) 코드로써 존재하며 구현할 파일시스템의 변화와 무관하기 때문에 단순하며 일관된 모습으로 존재한다. 그러므로 NFSF가 지원할 운영체제별로 하나의 클라이언트 모듈이 존재하게 된다. 이것은 이 프레임워크를 유지 보수하는데 편의성을 제공한다.

3.1.2 통신 채널

통신 채널은 앞서 언급한 SOAP 프로토콜을 사용해서 구현을 한다. 이 때의 전송 프로토콜은 HTTP를 사용했으며 클라이언트와 서버가 실제로 자료를 주고받기 위해 사용하는 부분이다. 그림 3에서는 통신채널을 통해 클라이언트와 서버가 데이터를 주고받는 모습을 보여준다.

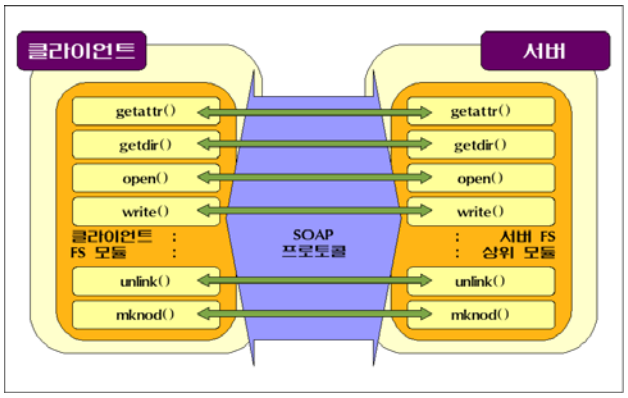

그림 3 NFSF: 통신 채널

SOAP 프로토콜을 이용한 내부 통신은 SOAP 자체 구현물로 존재하기 때문에 NFSF에서는 이를 이용하기 위해 FUSE가 요청한 API를 클라이언트가 객체로 묶어서 서버에게 객체를 전달하며, 서버는 클라이언트에게 API 객체를 처리한 결과를 보내도록 구현한다. 뒤에서 언급하겠지만 서버 상위 모듈은 클라이언트로부터 SOAP 통신을 통해 받은 API 객체를 처리하여 결과를 되돌려 보내지 않고 다시 한번 서버 하위 모듈로 API 요청을 바인딩 한다.

3.1.3 서버 파일시스템 상위 모듈

서버 파일시스템 모듈을 상위 모듈과 하위 모듈로 계층을 분리한 것은 상위 모듈에서 네트워크 파일시스템에 있어 필요한 공통분모를 최대한 끌어냄으로써 네트워크 파일시스템 제작자는 저장장치를 접근하는 방법에 대해서만 고민할 수 있도록 하기 위해서다. 즉 파일시스템 제작자는 FUSE 프레임워크에서 제공하는 API를 지원할 스토리지에 대해 로컬에서 접근하는 것과 동일하게 구현만 하면 된다. 나머지 데이터 통신 부분과 클라이언트 구현과 FUSE와의 통신 부분은 NFSF가 처리한다.

서버 파일시스템 상위 모듈은 클라이언트 파일시스템 모듈과 웹서비스 프로토콜인 SOAP을 이용해서 통신하므로 현존하는 여러 가지 프로그래밍 언어로 작성할 수 있다. 또한 이 부분은 서버 파일시스템 하위 모듈과 독립적이므로 프로그래밍 언어별로 바인딩을 해서 NFSF 차원에서 제공한다면 여러 프로그래밍 언어로 하위 모듈을 제작할 수 있다. 즉 이것은 네트워크 파일시스템을 다양한 프로그래밍 언어를 사용해서 제작할 수 있다는 것을 의미한다.

3.1.4 서버 파일시스템 하위 모듈 NFSF의 적용

서버 파일시스템 하위 모듈은 서버 파일시스템 상위 모듈과 저장장치 사이에 존재하는 모듈로 파일시스템 개발자가 작성해야하는 부분이며 NFSF에서는 최소한의 틀만 제공한다. 서버 파일시스템 하위 모듈을 보면 FUSE에서 제공하는 API와 동일한 함수 목록을 가지는데 이것은 서버 파일시스템 상위 모듈이 해당 API들을 적절한 하위 모듈에게 바인딩을 시키기 때문이다. 즉 서버 파일시스템 상위모듈은 리눅스 커널의 VFS와 비슷한 기능을 수행하며 실제 저장장치에 접근 하는 것은 하위 모듈이 처리한다.

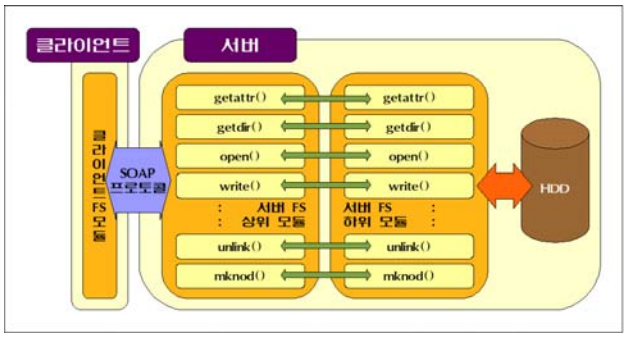

그림 4 NFSF를 적용한 예제 네트워크 파일시스템

그림 4는 NFSF를 적용해서 원격의 하드디스크를 접근하는 NFS와 동일한 기능을 갖는 예제 파일시스템을 작성한 예이다. 예제 파일시스템을 작성하는 과정은 리눅스 위의 일반적인 파일시스템을 접근하는 서버 파일시스템 하위 모듈을 작성하는 과정으로 축소할 수 있다. 이 과정은 사용자 수준의 프로그래밍이기 때문에 서버의 저장장치인 하드디스크의 파일시스템에 대한 고려를 할 필요가 없다. 하위 모듈을 완성하기 위해서는 NFSF에서 제공하는 FUSE API 25개에 대한 스텁 코드 내부를 채우면 된다. 예제 파일시스템은 일반적인 파일 및 디렉터리를 처리하는 루틴을 Perl로 작성해서 구현을 한다.

3.2 NFSF의 구현

여기서는 NFSF를 어떻게 구현했는지 동작원리를 설명한다. 구현하고 있는 25개의 API 중 하나인 getdir()을 예로 들어 클라이언트와 서버에서 바인딩을 처리하는 방법을 설명한다. 나머지 24개의 API도 전체적인 흐름에 있어 큰 차이가 없다. 그리고 이러한 FUSE API를 바인딩한 후 어떻게 객체로 만들어 SOAP 프로토콜을 통해 전송하고 결과 값을 전송 받는지를 설명한다. 설명을 위해 에러 처리 및 예외 처리, 객체의 초기화 및 소멸과 부가적인 기능 구현을 위한 부분들을 제거한 코드 조각을 사용한다.

3.2.1 NFSF에서 바인딩 하는 중요 함수

NFSF는 FUSE 파일시스템 프레임워크의 플러그인으로 동작한다. 확장성 있는 프레임워크를 지향하기 위해 클라이언트 파일시스템 모듈과 서버 파일시스템 상위 모듈은 모두 FUSE API를 서버 파일시스템 하위 모듈을 위해 바인딩을 한다. 세 단계에 걸쳐 FUSE API 객체를 바인딩하고 웹서비스 프로토콜을 통해 API 객체를 보내고 결과를 받는 작업은 제안하는 프레임워크의 핵심 부분이다.

표 1 NFSF에서 바인딩 하는 FUSE API 목록

FUSE API	수행 내용
getattr	stat의 결과와 비슷한 파일 속성을 반환
readlink	심볼릭 링크를 디레퍼런스
getdir	디엔트리를 리스팅
mknod	장치파일 생성
mkdir	디렉터리 생성
unlink	파일, 장치, 심볼릭 링크를 제거
rmdir	디렉터리 제거
symlink	심볼릭 링크 생성
rename	파일의 이름을 변경하거나 위치를 옮김
link	하드 링크 생성
chmod	파일, 디렉터리, 장치, 심볼릭 링크의 권한 설정
chown	파일, 디렉터리, 장치, 심볼릭 링크의 소유권 설정
truncate	주어진 오프셋에서 파일을 잘라냄
utime	접근시간 및 수정시간 변경
open	파일 열기
read	파일 읽기
write	파일 쓰기
statfs	파일시스템 정보 반환
flush	캐시 된 데이터를 동기화하기 위해호출
release	레퍼런스가 더 이상 존재하지 않는지 확인
fsync	파일 컨텐츠를 동기화하기 위해 호출
setxattr	추가 속성을 설정하기
getxattr	추가 속성을 얻기
listxattr	추가 속성을 나열
removexattr	추가 속성을 제거

<표 1>은 NFSF에서 바인딩 하는 FUSE API 목록을 나타낸 표이다. NFSF는 25개의 API에 대해서 서버 하위 모듈을 위한 스텁 코드를 제공하므로 파일시스템 개발자는 해당 API가 저장장치에 대해서 수행할 내용을 구현해서 스텁 코드 내부에 넣어야 한다.

3.2.2 클라이언트 파일시스템 모듈

클라이언트 파일시스템은 FUSE API 호출이 있을 경우 이를 해당하는 원격의 API와 바인딩을 하고, FUSE로부터 API의 호출 요청이 들어오면 해당 API에 적합한 파라미터를 SOAP 프로토콜을 통해 서버 파일시스템 모듈로 데이터를 전송한다.

그림 5는 NFSF 클라이언트 모듈의 스텁 함수의 하나를 보인다. 스텁 함수는 FUSE API 25개에 대해 각각 존재해야 한다. 스텁 함수가 수행하는 중요한 작업은 두 가지다. 하나는 FUSE API의 요청이 들어올 경우 SOAP 프로토콜을 통해 NFSF 서버 상위 모듈에게 메소드 호출을 요청하는 것이며 두 번째 작업은 요청한 메소드 호출의 결과를 다시 FUSE 모듈에게 반환하는 것이다. (그림 9)의 ①은 FUSE API의 getdir()을 바인딩 할 스텁(stub) 함수다. 이 때 네임스페이스의 오염을 막기 위해서 'x_' 접두사를 붙인 x_getdir() 함수를 정의한다. 그리고 ②는 FUSE의 요청이 있을 경우 SOAP 프로토콜을 통해 메소드 요청과 적합한 파라미터를 보내는 역할을 한다. ③은 호출한 메소드 요청의 결과를 FUSE에게 보내는 부분이다.

```
sub x_getdir {                                          ①
    my ( $remotedir, $dirname ) = @_;
    my $response = $client ->                           ②
            uri( $remote_libdir ) ->
            x_getdir($remotedir, $dirname);
            return @{$response->result};                ③
}
```

그림 5 NFSF 클라이언트 모듈의 스텁함수

```
Fuse::main(
            mountpoint => $mountpoint,
            getattr  => "main::x_getattr",
            # 나머지 24개의 API도 명시한다.
            # ...
);
```

그림 6 NFSF 클라이언트 모듈에서 스텁함수의 바인딩

그림 6은 스텁함수와 FUSE API를 바인딩 시키는 역할을 한다. NFSF 클라이언트 모듈에서 나머지 API의 동작도 대동소이하다.

3.2.3 서버 파일시스템 상위 모듈

서버 파일시스템 상위 모듈은 클라이언트 파일시스템 모듈과 서버 파일시스템 하위 모듈 사이의 다리 역할을 한다. 구현할 파일시스템의 실질 부분은 서버 파일시스템 하위 모듈에 존재하므로 클라이언트가 요청한 API 호출을 하위 모듈로 바인딩 시켜줘야 한다. 그림 7의 'Fuse::Driver::Linux'가 서버 파일시스템 하위 모듈을 의미하며 'dispatch_to()' 메소드를 통해 서버 모듈간의 바인딩 처리를 한다.

```
my $soap_server =SOAP::Transport::HTTP::Daemon
-> new (LocalAddr => $SERVER_NAME,
        LocalPort => $SERVER_PORT )
-> dispatch_to($libpath, 'Fuse::Driver::Linux');

$soap_server->handle();
```

그림 7 NFSF 서버 상위 모듈의 API 바인딩

3.3 NFSF를 적용한 네트워크 파일시스템

마지막으로 프레임워크에 포함되지는 않지만 NFSF를 적용해서 네트워크 파일시스템을 제작한다. 서버 파일시스템 하위 모듈은 파일시스템을 완성하기 위해서 직접 작성해야하는 최소한의 부분에 해당한다. 서버 파일시스템 하위 모듈은 프레임워크에 포함되지 않지만 프레임워크 차원에서 placeholder로써 최소한의 코드를 제공한다.

3.3.1 서버 파일시스템 하위 모듈

그림 7에서 언급한 'Fuse::Driver::Linux'의 코드 조각의 일부는 그림 8에서 볼 수 있다. 파일시스템 개발자는 FUSE API 중 'getdir'에 해당하는 'x_getdir'을 구현해야 하며, 나머지 API에 대해서도 마찬가지로 구현해야한다. 이 부분은 커널 수준이 아닌 사용자 수준의 프로그래밍이다.

```
sub x_getdir {
  my $self = shift;
  my ( $rootdir, $dirname ) = @_;
  $dirname = fixup($rootdir, $dirname);
  unless(opendir(DIRHANDLE,$dirname)) {
  return [-ENOENT()];
  }
  my (@files) = readdir(DIRHANDLE);
  closedir(DIRHANDLE);
  return [@files, 0];
}
```

그림 8 NFSF 적용 시 제작하는 서버 파일시스템 하위 모듈

4. NFSF의 동작환경 및 평가

4.1 동작환경

NFSF를 제작하기 위해 사용한 개발 환경은 다음과 같다. 개발 언어로는 Perl v5.8.4와 v5.8.8을 사용하며, Linux Kernel 2.6.17의 Ubuntu 6.10 Edgy Eft에서 작업한다. 네트워크 파일시스템의 서버는SOAP 통신을 위해 Perl SOAP::Transport::HTTP v0.60 라이브러리를 사용한다. 클라이언트는 SOAP 통신과 FUSE 구현 함수들을 바인딩 하는 Stub을 작성하기 위해 Perl SOAP::Lite v0.60, Fuse 2.5.3, Perl Fuse v0.07을 사용한다.

4.2 평가

<표 2>는 네트워크 파일시스템 제작 시 파일시스템 프레임워크 사용 여부와 프레임워크의 종류에 따라 갖는 특징을 비교한 결과다. 파일시스템 프레임워크를 적용할 때와 그렇지 않을 때의 가장 큰 차이는 클라이언트 커널 모듈의 제공 여부이다. 파일시스템에 있어서 클라이언트 커널 모듈은 운영체제의 파일시스템 커널과 연동하는 부분이다. 직접 네트워크 파일시스템을 제작한다면 커널 모듈 부분을 반드시 작성해야 하지만 파일시스템 프레임워크인 LUFS[9]와 FUSE를 사용하거나 네트워크 파일시스템 프레임워크인 NFSF를 사용하면 프레임워크에서 제공하는 커널 모듈을 사용할 수 있다.

표 2 네트워크 파일시스템 제작 시 프레임워크 간 비교

분류 / 특징	네트워크 파일시스템 제작	LUFS	FUSE	NFSF
프레임워크 사용여부	미사용	사용	사용	사용
클라이언트 커널 모듈 제공	X	O	O	O
클라이언트 모듈의 호환성	X	X	X	O
개발언어 선택의 유연성	X	X	O	O
통신 모듈 제공	X	X	X	O

프레임워크에서 제공하는 커널 모듈을 사용할 수 있다는 점은 LUFS와 FUSE, NFSF가 동일하나, LUFS와 FUSE는 네트워크 파일시스템 클라이언트 모듈을 만들기 위한 API를 제공하는 것이기 때문에 각기 다른 네트워크 파일시스템에 대해 클라이언트 모듈을 재작성해야 하는 번거로움이 존재한다. NFSF는 네트워크 파일시스템의 클라이언트 모듈을 단일화하고 의존적인 부분을 서버 측 처리로 넘김으로써 네트워크 파일시스템의 종류에 상

관없이 동일한 클라이언트 모듈을 사용할 수 있다.

또한 네트워크 파일시스템을 직접 제작하거나LUFS를 사용하는 경우 개발 언어에 제약이 있으나 NFSF는 FUSE와 마찬가지로 네트워크 파일시스템을 만드는데 있어 개발 언어의 제약이 거의 없다. 이것은 NFSF를 적용할 경우 네트워크 파일시스템 개발 시 제작해야하는 서버 측 모듈이 웹-서비스(SOAP) 규약의 원격 프로시저 호출을 사용하기 때문이다. 현재 SOAP을 사용하기 위한 라이브러리는 대부분의 범용 운영체제와 언어에 구현되어 있다[5].

파일시스템 프레임워크인 LUFS와 FUSE는 네트워크 파일시스템을 직접 제작할 때와 마찬가지로 서버 파일시스템 드라이버와 클라이언트 파일시스템 드라이버가 데이터를 주고받기 위한 부분을 구현해야 한다. NFSF는 SOAP 프로토콜을 사용하기 때문에 서버와 클라이언트의 통신 방법에 대한 고려를 생략할 수 있다.

제 5 장 결론 및 향후 과제

5.1 결론

본 논문에서 제안한 NFSF는 가벼운 클라이언트 파일시스템 드라이버와 서버 파일시스템 모듈을 상위 모듈과 하위 모듈 2 계층으로 나누어 상위 모듈을 제공함으로써, 실제 저장장치에 의존적인 부분은 하위 모듈이 처리하게 한다. 이러한 설계 자체의 특성은 그 자체로서 다음 두 가지 장점을 가진다. 첫 째, 씬-클라이언트(thin-client)의 구현으로 클라이언트 파일시스템 드라이버의 유지 보수와 배포가 수월하다. 둘 째, 범용성 있는 SOAP 프로토콜을 이용해 네트워크 통신을 수행하기 때문에 이질적인 플랫폼에 이식하기 쉽다.

제안하는 프레임워크를 적용할 경우 얻을 수 있는 이득은 세 가지다. 첫째, 저장장치에 의존적인 서버 파일시스템 하위 모듈만을 제작하면 되므로 네트워크 파일시스템을 빠르게 개발할 수 있다. 하위 모듈에서 기본적으로 필요한 코드는 placeholder 형식으로 추가적으로 제공할 수 있다. 파일시스템 제작자는 placeholder 함수 내부만 작성하면 되므로 네트워크 파일시스템의 개발과정을 크게 단축시킬 수 있다.

둘째, 파일시스템 제작 시 프로그래밍 언어의 선택이 자유롭다. 서버 파일시스템 상위 모듈은 클라이언트 파일시스템 모듈과 SOAP을 이용한 웹서비스 통신을 하므로 쉽게 다른 언어로 제작할 수 있다. 프레임워크에서 여러 프로그래밍 언어로 서버 상위 모듈을 기본적으로 제공한다면 서버 하위 모듈을 C나 Perl 뿐만 아니라 C++, Java, Python, Ruby 등의 언어로도 구현할 수 있으므로 파일시스템을 제작하는데 있어 선택의 폭이 넓어진다.

셋째, 파일시스템을 작성 시 고려사항이 줄어든다. 서버 파일시스템 하위 모듈은 네트워크 통신과 파일시스템 시스템 콜 API 객체의 요청 및 변환 응답, 메시지 패싱과 같은 네트워크 통신 작업과 분리되어 있다. 또한 클라이언트 파일시스템 모듈이 FUSE 모듈과 연동하여 리눅스 커널 하부의 VFS와 통신을 대신 처리해주므로 서버 파일시스템 하위 모듈은 커널 프로그래밍 작업과도 분리되어있다. 따라서 파일시스템 제작자는 운영체제 커널 모듈 구현과, 네트워크 통신 모듈 구현에서 자유로우므로 저장장치 자체의 접근을 어떻게 할지에 대해서만 집중하면 된다.

5.2 향후 과제

이러한 여러 가지 장점에도 불구하고 제안한 네트워크 파일시스템 프레임워크는 성능 면에 있어 두 가지 단점을 지니고 있다. FUSE를 사용한 일반적인 네트워크 파일시스템과 비교해서 2 단계, FUSE를 사용하지 않은 파일시스템과 비교해서 3 단계를 더 거치므로(FUSE를 통해 거치는 부분을 포함한다면 4 단계) 필연적인 성능저하를 동반한다. 또한 프레임워크의 광범위한 적용을 위해 선택한 범용적이고 확장성이 뛰어난SOAP 프로토콜은 XML이라는 텍스트 형식의 인코딩을 사용하기 때문에 바이너리 통신을 이용하는 경우에 비해 추가적인 성능저하가 발생한다. SOAP 프로토콜이 짧은 메시지의 교환에서는 큰 성능저하가 없지만, 파일시스템과 같은 잦은 메시지 교환과 대용량 파일 접근과 같은 긴 메시지 교환에 있어서 바이너리 프로토콜에 비해 많은 저하가 있기 때문이다. 향후 이 두 가지 성능저하 문제를 제안한 네트워크 파일시스템 프레임워크가 가지는 장점을 보존하면서 해결하는 연구가 필요하다.

또한 현재는 네트워크상의 사용자 인증이나 보안과 관련한 부분은 구현되어 있지 않다. 다양한 언어로의 서버 상위 모듈 제작, FUSE를 사용하지 않는 리눅스 이외의 운영체제에 제안한 프레임워크를 적용하는 방법과 관련한 연구가 필요하다.

참고문헌

[1] B. Pawlowski, C. Juszczak, P. Staubach, C. Smith, D. Lebel, and D. Hitz. “NFS Version 3 Design and Implementation”. In Proceedings of the Summer 1994 USENIX Technical Conference, Jun 1994.

[2] World Wide Web Consortium. “Web Service Activity”. http://www.w3.org/2002/ws/.

[3] Simon ST. Laurent, Joe Johnston, and Edd Dumbill. “Programming Web Services with XML-RPC”. O’REILLY, 2001.

[4] Randy J. Ray, and Pavel Kulchenko. “Programming Web Services with Perl”. O’REILLY, 2002

[5] James Snell, Doug Tidwell, and Pavel Kulchenko. “Programming Web Services with SOAP”. O’REILLY, 2001

[6] Ethan Cerami. “Web Services Essentials”. O’REILLY, 2002

[7] “Filesystem in Userspace”. http://fuse.sf.net/.

[8] “A Virtual File System”. http://avf.sf.net/.

[9] “Linux Userland File System”. http://lufs.sf.net/.