xxfs:
ネットワーク資源の
ファイルシステムによる抽象化に関する研究

関　康治[†]　植原啓介[†]
楯岡孝道[††]　村井純[†]

本研究では単一のファイルシステムプロトコルを通じて，数々のネットワーク上のサービスを抽象化することを目指す．さらに管理者だけでなく一般ユーザ権限しか持たない者でも，サービスをファイルシステムに導入できるようにする．その第一段階として，他ホスト上には一般ユーザ権限しか持たぬものが自ホスト以外のホストに存在するファイルツリーを自ホストのファイルツリーに統合できるようにするなシステムを構築する．

xxfs:
Study for Network Resources Abstraction by File System

YASUHARU SEKI,[†] KEISUKE UEHARA,[†] TAKAMICHI TATEOKA[††]
and JUN MURAI[†]

End of this study is network service abstruction by a file system protocol. Not only super-user, general user can add any filesystem to there system. At first, I made a file system mechanism which is user who do not have super-user permission can add filetree to his/her filetree.

1. 背　景

現在の多くのオペレーティングシステムでは情報の保持・記録にファイルシステムを使用している．

ファイルシステムは，一般的なデータを保存するには広大な記憶容量をもつ補助記憶装置に保存するデータを，より細かい単位で管理するために使用されている．ユーザは，このファイルシステムを使用して自分がコンピュータ上で作成したデータやプログラムを保存する．

ファイルシステムは，当初自ホストに存在する補助記憶装置を操作するためのものであった．しかし，やがてコンピュータがネットワークを介して操作されるようになってくると，ネットワークを経由してファイルを操作する場面が増えてきた．このような要求に応えて作られたのがネットワークファイルシステムである．

ネットワークファイルシステムを用いると，別のホスト上に存在するファイルをあたかも自ホストに存在するかのように使用する事ができる．これはユーザが多くのホストを自由に使える環境では有効であり，広く利用されている．

2. 現状と問題点

現状では，ファイルシステムの多くは単純に補助記憶装置にデータを書き込むだけの機能しか持たない．しかし，ファイルを管理するということは，そこに存在するデータを単純に読み書きする事だけではない．時間的なファイルの変遷を保持しておきたい事もあるし，一度消去したデータを元に戻したい事もある．しかし，このような機能は，現状ではファイルシステムに統合されず，コマンドとして実現されている．

また，現状のファイルシステムでは，新たにファイルツリーを拡張するためには管理者である必要がある．このため，ユーザが他ホストに存在する自分のファイ

† 慶應義塾大学環境情報学部
Faculty of Environmental Infomation, Keio University
†† 電気通信大学大学院 情報システム学研究科
The Graduate School of Information Systems, University of Electro-Communications

ルを自ホストで操作しようとする場合には，自ホストにファイルを複製する必要がある．しかし，ユーザが自分の権限でファイルツリーを拡張して，他ホストのファイルツリーを自ホストに結合できる様になっていれば，より便利にファイルの操作ができる．

本研究ではこのような問題について解決を図る．

2.1 ファイルツリーの追加

2.1.1 現　　況

ある一群のユーザが幾つものホストを使用可能な状態を考える．ユーザはそのうちのどのホストを使用している場合であっても同一の動作環境を得たいと考えると，ユーザ自身が使用できるファイルはそれらのホスト間で共有できるようにするべきである．これはSun NFS (Network File System) 等の共有ファイルシステムを用いることにより実現可能である．

しかし，Sun NFS はファイルを公開する側が，あるホストからのファイル操作要求を許可するか否かを明示的に設定する必要がある．この設定は管理者しか行なえない．また，mount 要求する側も mount コマンドを実行するのは管理者である必要がある．

ここでは Sun NFS を例としたが，一般にファイルツリーの拡張をすることができるのは管理者だけである．

2.1.2 問　題　点

昨今のようにワークステーションなども安価になり，コンピュータが業務に欠かせなくなってくると，一人のユーザがネットワーク的に離れた場所に複数のアカウントを持つことも珍しくなくなってきた．このような状態では，あるアカウントを利用している時に別の管理形態をもつホストに存在するファイルを使用する必要が出てくる．しかし，多くの場合，ユーザは双方のホストの管理者であること少なく，Sun NFS による mount は難しい．このような場合，ネットワークを通じて相手ホストに login して当該ファイルを操作する方法と，当該ファイルを ftp(File Transfer Protocol) などを利用して自ホストが操作できるファイルツリーに複製して操作する方法とが考えられる．ファイルを取得した場合は，一通りの操作が終了した後に元のホストに返却する工程を踏む必要がある．

2.2 ファイルシステムの追加

ファイルに関するサービスは多くの場合カーネル空間に組み込まれている．

例えばディスクよりデータを読み込み，それを解釈してファイルとして出力する機能を持つモジュールはその機能性から言ってユーザ空間で処理するよりはカーネル内部で処理する方針の方が適している．

現況では，ファイルを読み書きするだけでない付加価値を持ったファイルシステムを開発しようとする場合には統一的で有効な開発手段は確立されていない．

例えば，ファイルリビジョンコントロールシステムを備えたファイルシステムを独自に開発することを考える．リビジョンコントロールは多くの場合，ユーザが個々にどのようなバージョン管理を行なうか設定を行なったり，ファイル毎にバージョンをどのように管理するかなど細かく行なえるようにしておくべきである．このような操作はカーネル内で行なうべきではない．ユーザ空間で行なう場合と比較して，デバッグの難しさや，設定の読み込みをどうするかなどの問題が多く存在するからである．

上記のような問題を解決する方法の一つとして，Sun NFS プロトコルなどのカーネルとユーザ空間で通信するファイルシステムプロトコルを仲介に利用してファイルシステムを開発する手法がある．しかし，現況ではこのようなファイルシステムは独自にプロトコルスタックを実装しているため，この実装に多くの労力を伴っている．

また，mount システムコールは管理者のみが発行できる．このため，ユーザが新たにユーザ空間で動作するファイルリビジョンコントロールシステムを内包したファイルシステムを開発して使用しようとした場合には，管理者に依頼してマウントをして貰うか，mount コマンドに suid ビットを立てて貰う必要がある．

2.3 アプローチ

以上の問題を解決する方法として，本研究ではファイルシステムを容易に開発し，使用するためのシステムを設計・実装する．

本システムでは，ファイルシステムの開発者に対し，プロトコルスタック部分を提供することで，開発者はファイル操作部のみを開発すればよいようになる．

また，一般ユーザ権限しか持たぬ者であっても新たにファイルシステムを開発し使用できるため，別のホストに存在するファイルを操作する際にも，ユーザが必要に応じて自分のファイルツリーに，必要なファイルを含むファイルツリーを組み込むことができる．

3. システムの目的

上記現状と問題点に基づいて新ファイルシステム (xxfs) の研究を行なう．

本研究においては下記の項目を実現する事を目的とする．

- 一般ユーザ権限しか持たない者がその権限の範囲

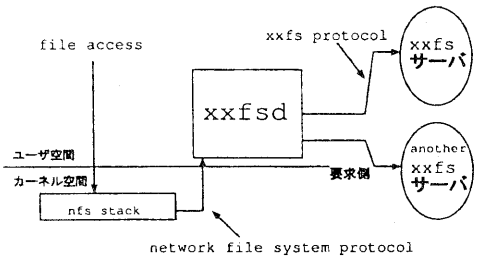

図1　xxfs の概要

内でファイルツリーの拡張をすることができるようにすること．

- ファイルシステムの開発者が容易に新たなファイルシステムを設計・導入できるようにすること．

これらを実現するための枠組と機構を設計する．

4. システムの概要

まず，システムの概要について説明する．

xxfs は xxfsd というデーモンと複数の xxfs サーバから構成される．

xxfsd はネットワークを用いて xxfs サーバ と xxfs プロトコル を利用した通信をする．xxfsd は内部に当該 xxfsd が管理している xxfs サーバのリストを持ち，xxfs サーバ からの要請に応じて，xxfsd 内部で保持している通信可能な xxfs サーバリスト に当該サーバを登録する．

xxfs サーバ は一般ユーザ権限でも動かすことができるファイルサーバであり，xxfsプロトコルを理解して，xxfsd からのファイル操作要求を実行する．xxfs サーバ は起動時に xxfsd に登録の要請を送り，その後 xxfsd からのファイル操作要求を待つ．

カーネルに xxfs が管理するファイルもしくはファイルツリーに対して操作要求があると，ネットワークファイルシステムプロトコルを用いて xxfsd と通信する．xxfsd はその操作されたファイルの存在するディレクトリに応じて該当する xxfs サーバ と xxfs プロトコル を用いて通信し，当該操作を実行する．

上記をまとめ，xxfs の概要を図 1に示す．

このファイルシステムはユーザ空間で動作する．カーネル空間上に実装した場合は，デバッグの難しさ，システムとしての設定の難しさ，移植性のなさが問題になるためである．また，カーネル内に組み込むと当該ホスト以外ではこのシステムの恩恵を受けることができないという理由もある．

4.1 デーモン部

xxfs の主要部はユーザ空間上でデーモン (xxfsd) として動作し，カーネルとネットワークファイルシステムプロトコルを通じて通信することで実現する．ファイルシステムプロトコルに Sun NFS や SMB などのネットワークファイルシステムプロトコルを利用することで，現行のカーネルに大きな変更を加えずとも xxfs を実現することができる．

また，ネットワークを通じて他のホストで動作している xxfsd を利用することにより，xxfsd が動作していない，もしくは技術的に動作させるのが難しいホストで xxfs を利用することもできる．

xxfsd の動作ホストに存在する情報は xxfsd 自体で取得することができるが，他ホストに存在するファイルを取得するためには，xxfsd 自身が更にホスト間の通信を行なう必要がある．このため，xxfsd は他の xxfs サーバ と通信する機構を持つ．

一つの xxfsd に対して xxfs サーバ は複数扱えるようあるべきなので，xxfsd 内部で現在使用可能な xxfs サーバ のリストを保持しておく．xxfsd はカーネルからの要求に応じて，必要な xxfs サーバ との通信を確保する．

他の xxfs サーバ との通信には，新規に制定した xxfs プロトコル を用いる．ここに既存のプロトコルを用いなかった理由は，カーネルと xxfsd との通信に既存のプロトコルを用いると，xxfs としての柔軟性が失われてしまうということが挙げられる．また，例えばファイルハンドルなどの単一のファイル指定方法に拘泥すると，そのプロトコルに合わせるための xxfs サーバ の開発者の手間が増える．これは xxfs プロトコル の方でファイルの指定方法が単一でないよう設計しておけば，xxfs サーバ 開発の際に好きな指定方法を使用して開発することができるようになる．

xxfsd 自体にライブラリとして新しいファイルシステムやカーネルと通信するプロトコルを付加させられるようにすることで，xxfsd の柔軟性を向上させることができる．例えば，xxfsd 自身に nfsd としての機能を持たせれば，そのまま xxfsd を nfsd と交換することで，今までの機能をそのままに xxfs の機能を享受することができる．また，xxfsd のカーネルとの通信をするネットワークファイルシステムプロトコルを SMB などに変更することによって，大きな変更無しに Windows98/NT などで xxfs の機能を使用することもできるようになる．

4.2 サーバ部

xxfs サーバ は xxfs プロトコル を理解し，その内容に応じてファイルの操作・参照を行ない，適切な内容を xxfsd 側に対して応答する機構を持つユーザ権限

```
# mount -t nfs localhost:/ /mnt/xxfs
```

図 2 mount する際のコマンドの例

で動作するデーモンとして設計する．要求に対して単純にファイルの操作・参照を行なうだけでなく，xxfs サーバが機能を付加することも考えられる．例えば，リビジョンコントロールシステムをファイルシステムに統合してしまうことで，特別な手続きを必要とせずにファイルのバージョン管理をすることができる．SMTP や HTTP などのネットワークサービスをファイルシステムに抽象化・導入することで，様々なネットワークサービスをファイルシステムを通じて使用することもできるようになる．新たなファイルシステムを作成する際には，プロトコルスタックまで用意してあるので，開発者はファイル操作部のみ作成すればよいようになっている．

また，xxfs サーバは一般ユーザ権限で動作できるため，管理者でなくとも自分で使用するファイルツリーを拡張する事ができる．

5. 動作概要

ここでは説明に具体性を持たせるために，カーネルとの通信プロトコルに Sun NFS version 2 を，ファイル操作要求に xxfs プロトコル を使用して，実際のファイル操作には xxfs サーバ のみを用いる事を前提にする．

5.1 マウント

xxfsd に存在するファイルツリーをマウントするためにはホスト側で mount システムコールを発行する必要がある．このため，xxfs は xxfs をマウントするためのディレクトリポイントが必要であり，また xxfsd によってサービスされているファイルツリーのルートディレクトリを当該ディレクトリポイントにマウントする必要がある．

具体的には，例えば /mnt/xxfs というディレクトリを用意し，図 2に示されるようなコマンドを実行する．

なお，3つ目の引数に存在するマウントポイントは現在はどのポイントを指定しても動作に変化はない．これは将来的には某かの意味を持たせる事も考えている．

5.2 デーモン部

xxfsd の起動後は，xxfs サーバ からのファイルツリーの登録要請とカーネルからのマウント要求・カーネルからのファイル操作要求を待つ．

カーネルからのマウント要求があった場合には，xxfsd が管理するファイルツリーのルートディレクトリのファイルハンドルを返す．

xxfs サーバ からファイルツリーへの登録要請があった場合には，ユーザ認証フェーズなどを経て，xxfsd 内部に持つ，ファイルツリーポイントを保持するデータリストに xxfs サーバ のホストやポートなどの情報を登録する．カーネルなどからファイル操作の要求があった場合に，このデータリストを参照して，要求ファイルのディレクトリに応じて適切な xxfs サーバ と通信する．

なお，カーネルから xxfsd が管理するファイルツリーのルートディレクトリの参照があった場合には，xxfsd が管理するファイルツリーのリストをディレクトリのリストの形で返す．これは，現在使用可能なクライアントのリストを示すものである．この部分についてはファイルの書き出し・ディレクトリ名の変更などはできない．

xxfs サーバ の対応付けには，現行では Sun NFS ファイルハンドルの上位 2bytes を利用している．xxfs サーバが登録要求をしてきた時点で，2bytes の名前を与える．xxfsd はカーネルから要求があったファイルについてファイルハンドルを解釈し，その上位 2bytes に応じた xxfs サーバ と接続する．

5.3 サーバ部

xxfs サーバ は動作を開始すると，コマンドライン等から判断して，目的の xxfsd に当該 xxfs サーバ が管理するファイルツリーの登録要求を行なう．

xxfs サーバ は xxfsd に対して xxfsd が管理するファイルツリーのどこにマウントするかの情報や，xxfsd がカーネルに対して export する際のユーザの権限についての情報などを渡す．

一度登録要求を送ってしまった後は，xxfs サーバ は xxfsd からファイル操作要求があるまでは待機状態になる．そして，xxfsd から要求があると，その操作に応じた内容の動作を行ない，結果を xxfsd に応答する．xxfsd はその結果を適切に処理し，カーネルに送付する．

このサーバ部には単純に要求に応じてファイルを操作するだけでなく，ファイルの操作に対応付けして，ネットワークサービスをファイルシステムに抽象化する事ができる．例えば，ある xxfs サーバ に ftp スタック を実装しておき，xxfsd のリクエストに応じて，ftp スタック を動作させることで，疑似的に ftp 先をマウントしてユーザ側からはあたかもローカルファイルを使用しているかのような操作をすることもできるようになる．また，例えば SMTP を適切にファイルシ

ステム操作に抽象化する事で，ファイルを書き出す事で適切な相手に対しメールを送信する事もできるようになることも考えられる．

6. 設　　計

xxfs の具体的な設計について説明する．

xxfs の OS のカーネルからのリクエスト部分については，Sun NFS 等のネットワークファイルシステムスタックをそのまま利用する．

6.1 デーモン部

xxfs サーバと通信をする事に焦点を当てると，xxfsd は大きく以下の部分に分類する事ができる．

- カーネルからのファイル操作要求受付部
- カーネルからのマウント受付部
- xxfs サーバからの登録要請受付部
- xxfs サーバへの要求発信部
- xxfs のメインモジュール

ここでは要求待ち受けの要素は３つある．マウント要求の待ち受けとカーネルからのファイル操作要求待ち受け，xxfs サーバ の登録要請待ち受けである．

これらを効率良く実現するためには，これらの別々の機能を一つの待ち受けルーチンにまとめてしまう必要がある．これらをまとめることで，其々の部分に対して別のルーチンを実装する必要がなくなる．

具体的には，それぞれの待ち受け部を RPC(Remote Procedure Call) を利用して実装することで実現する．xxfsd は起動と同時にこれら３つのプロトコルの待ち受けリクエストを portmapper に登録する．いずれかの待ち受け部にリクエストが来た場合には，portmapper からの情報を元に適切な機能を呼び出す事になる．

xxfsd のメインモジュールでは xxfs サーバ の情報管理と xxfs サーバ の名前つけを行なう．xxfs サーバに関する情報は xxfsd 内部にデータを保持する．xxfs のファイルツリーのルートディレクトリの構造はこのデータに依存し，xxfs サーバ のみに対応する場合は，当該 xxfsd に登録要求をしてきた xxfs サーバ の数だけディレクトリが存在する事になる．

結果として，xxfs サーバ の登録要求に応じて xxfs のファイルツリーのルートディレクトリは変化する事になる．

将来的にはこのメインモジュール部にファイル自身のキャッシュを備えて高速化することを考えている．また，ファイルハンドラとファイル名をマッチングさせるルーチンを備え，新ファイルシステムの開発者の労力を軽減させる事も考えている．

6.2 サ ー バ 部

サーバ部は xxfs プロトコルスタック 以外の部分については大きな制約は設けない．

xxfs プロトコルスタック はファイルに対する操作機能毎に関数を呼び出す．例えば，ファイルのリードを行なうリクエストをカーネルが行なった場合には最終的に xxfs サーバ の xxfs プロトコルスタック が xxfs サーバメイン部のファイルリード関数を呼び出す事になる．メイン部の実装者はその関数の意味に応じた機能を実装すればよい事になる．

通常はこのメイン部では直接ファイルに対して操作する機能を実現するだけであるが，通常のファイルシステム以外を実装する場合にはその機能に抽象化された機構を実装する必要がある．機能の抽象化自体は xxfs サーバメイン部 を実装する者が考える．

6.3 新ファイルシステムの実装

xxfs では新たなファイルシステムを実装しやすくする事を目的の一つとしている．ここでは新たに付加価値を付けたファイルシステムを実装しようとする場合に xxfs のどこの機構にファイルシステムを実装するべきかを考える．

まず，一番に考えられるのは xxfs サーバ として動作させる事である．xxfs プロトコルスタック を実装し，その機能と新たなファイルシステムの操作を抽象化して結合する事で，新たなファイルシステムは動作をするようになる．ただし，この手法では一度 xxfsd とカーネルが通信した後，更に xxfsd と xxfs サーバとが通信する必要があるため，高速な動作は期待できない．開発初期においては容易に再起動可能な xxfs サーバ として実装する方が容易に開発できる．

このため，二番目に考えられるのは，xxfsd にモジュールとして組み込むことである．これは，xxfs サーバ を実装する場合と違い，デバッグの手間が増えるが，通信のオーバーヘッドはなくなる．

これらの実装位置について，それぞれ図 3に示す．xxfs プロトコル を通じて xxfsd と通信しているものが，xxfs サーバ として動作させた場合であり，xxfsd 内部に存在するのが，xxfsd にモジュールとして組み込む場合である．

これら２つの内どちらを使うかは目的により，分けられるべきである．

6.4 問　題　点

設計上，また構造上幾つかの問題点が考えられる．

6.4.1 再 起 動 性

現在の実装では，xxfs ではカーネルと xxfsd の間の通信に Sun NFS プロトコルを使用する．Sun NFS

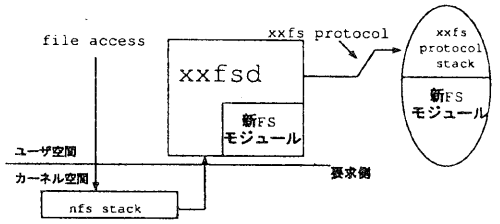

図3　新FSを組み込む位置

プロトコル の特徴の一つに，nfsd はクライアントの状態を保存しない事があげられる．このことで，nfsd が中途で動作しなくなったとしても，再度 nfsd を立ち上げ直す事によりクライアント側は問題なく使用を続ける事ができる．

しかし，xxfsd は xxfs サーバ の状態を保存する．このため，一度 xxfsd を再起動した場合にはこれらの情報は消える事になる．これにより，Sun NFS プロトコルの特徴である耐障害性に支障をきたす．これは xxfs サーバ が定期的に xxfsd に登録の確認を行なう事で復帰が可能であるが，復旧までに若干時間が必要になるという欠点を持つ事になる．

6.4.2　権限について

自由にユーザがファイルツリーを拡張できるようになると，当然のことながら使用権限を考慮しなくてはならない．

xxfs サーバ を使用する事を考えると，xxfs サーバ が動作するホストのユーザ管理形態と xxfsd を使用してマウントしているホストのユーザ管理形態は異なる事が多い．この場合，例えば xxfs サーバ からリクエストによってマウントされたファイルツリーは誰の権限で操作できるようにしておくかという問題が生ずる．

結局 xxfs サーバ を起動する時点で xxfsd に対してどのユーザ権限でファイルツリーに追加するかを指定する必要が出てくる．これは，設定ファイルで行なうか，xxfs サーバ を立ち上げる際にパスワードなどによる認証をする等の機能を xxfs プロトコルに実装する必要がある．

6.4.3　セキュリティについて

基本的には xxfs サーバ は一般ユーザ権限で動作する事で xxfs サーバ が動作する側では xxfs サーバ を動作させたユーザ権限以上の操作をする事はできない．

しかし，ホストに存在するの多くのファイルの読み込みは一般ユーザ権限でできるため，例えば passwd ファイルを含むファイルツリーを export された場合には，xxfsd をマウントしているユーザから xxfs サーバ が動作するホストの passwd ファイルが読める．このため，xxfs サーバ が動作するホストにアカウントを持たず，xxfsd をマウントしているホストにはアカウントを持つユーザが passwd ファイルに記述されたパスワードエントリを crack するという可能性がある．

また，例えば，xxfs サーバ を実行するユーザがルートディレクトリを export したとすると，xxfsd を用いてそのファイルツリーをマウントしているサーバからは xxfs サーバ を実行しているホストの設定ファイルの多くを読む事ができる．

xxfs サーバ自体がそのようなディレクトリをマウントできないようにするという手法も考えられる．しかし，xxfs 自身がファイルシステムを容易に開発できることを目的としているため，ユーザが xxfs サーバ を開発する事も考えなくてはならない．そのため，このような前提を言う事は難しい．

結論として，xxfs サーバ が許可したユーザ以外のユーザは，xxfs サーバ経由でディレクトリやファイルを読めないようにするような機構を xxfsd もしくは xxfs プロトコル に内包させる事が要求される．

7. 実　　装

現在，xxfsd は複数の xxfs サーバ を制御できる機構を備えている．また，xxfs プロトコルに応じて単純にファイルの操作を行なう xxfs サーバ を実装した．この xxfs サーバ をあるホストで動作させ，xxfsd を通じて動的に mount/umount ができるようになっている．この段階では，カーネルとの通信部は単一のプロトコルのみに対応し，xxfsd の機構は xxfs サーバ と通信するために特化されている．

xxfsd とカーネル部の通信には Sun NFS version 2 を利用し，Linux 2.0.3x(slackware3.5) 上で xxfsd が動作し，xxfs サーバは別の管理形態をしている solaris2.6 上で一般ユーザ権限で動作するよう実装した．xxfsプロトコルは，現在は Sun NFS version 2 に準拠した作りになっており，ファイルの指定には Sun NFS ファイルハンドルを使用している．

Linux から Sun NFS を使用して xxfsd をマウントし，xxfs サーバから登録要求を送信すると，Linux 側から solaris 上に存在するファイルツリーを表示したり，ファイル操作したりすることができるようになっている．

7.1　実際の使用例

ネットワーク上に複数のアカウントを所持しており，そのうち一つを使用している際に，別のアカウントを使用している時に実装したファイルを参照・操作したくなる事がある．このような時に xxfs を用いると楽

に作業を続ける事ができる．

以下に実際の使用法を具体的に記す．

まず，自分の使用しているホストで xxfsd を動かし，管理者が xxfsd によってサービスされているファイルツリーをマウントする．この時にはまだ xxfsd にはファイルツリーは登録されていないので，xxfsd が管理しているディレクトリの中を見ても何も登録されていない．

次に xxfs サーバを使用したいファイルの存在するホストで動かす．この際，一度使用したいファイルの存在するホストで xxfs サーバを実行するためのコマンドを実行する必要がある．xxfs サーバが無事に立ち上がると，xxfs サーバは自分の使用しているホストで動作している xxfsd に対して登録要求する．この際 xxfsd は，パスワード認証などにより，登録リストに登録して良いか判断する．

xxfsd はこの時点で xxfsd が管理するルートディレクトリに新たなディレクトリを追加する．なお，この xxfsd が管理するルートディレクトリ部分についてはユーザがディレクトリを作る事やファイルを作る事はできない．

自分が使用しているホストで xxfsd によって管理されているディレクトリにカレントディレクトリを移し，ディレクトリの中を見ると，xxfsd は xxfs サーバに対してディレクトリ閲覧の要求を発行する．xxfs サーバは要求に応じて当該ディレクトリの内容を応答し，その結果が自ホストのコマンドライン等に返される．

このような状態でファイルを操作すれば，通常の Sun NFS を利用して共有しているファイルシステムであるかのように扱う事ができる．

8. 関連研究

8.1 Sun NFS

Sun NFS は他のホストに存在するファイルツリーを，あたかも自ホストのファイルツリーの一部であるかのように操作できるファイルシステムである．Sun NFS の特徴は，NIS とともに使用する事によって，いくつものホストで同様の環境を容易に作れる事にある．しかし，ファイルツリーを提供するホストと使用するホストで，ユーザの管理形態が同じである事を前提として作られているため，異なる管理形態のホスト間では使用しづらい．また，ユーザが個人でファイルツリーを拡張する事は考えられておらず，管理者がファイルツリーの管理を行なう．

8.2 wwfs

xxfs は wwfs のアイディアを元にして作られている．wwfs は Sun NFS を使用して FTP で提供されるファイルシステムを自ホストのファイルツリーに統合するためのファイルシステムである．wwfs はそのマウントするファイルシステムを ftp に限定している．本研究では，wwfs により汎用性を持たせた物を設計・実装している．

xxfs では，xxfsd の動作中に動的にファイルシステムを導入・使用する事ができる．また一般ユーザ権限を持つ者が必要に応じてファイルツリーの追加をできることも wwfs と比べた特徴の一つである．

9. 今後の予定

今後，xxfsd をモジュール化し，カーネルとの通信部とファイルへの操作要求部をそれぞれ容易に新しいプロトコルに変更できるようにする．これによりカーネルの通信部を，例えば SMB や appletalk を使用できるライブラリを導入する事によって，これらのプロトコルを用いた通信をすることができるようになる．また，操作要求部に xxfs プロトコルを用いた通信をしないモジュールを使用することによって，ローカルファイルシステムに対するアクセスを高速化させることができる．xxfs プロトコルスタックはこの時点で xxfsd の数あるモジュールの中の一つに過ぎない扱いとなる．

上記のシステムでは xxfsd にモジュールを追加するためには xxfsd の再構築を行なう必要があるが，これをダイナミックに結合するようにして，新たなファイルシステムを付加したりカーネルとの通信プロトコルを追加したりできるようにする事で，システムに柔軟性を与える事ができる．

また，xxfsd 内部にキャッシュの機構やファイル特定方法の変換機構を備える．この段階では主に高速化や高性能化を図る．

現況では構造的なセキュリティについてもまだ脆弱だが，ネットワーク上におけるスニファ対策などについては非常に脆弱であると言わざるを得ない．これをもっと頑強なセキュリティ構造を実装する必要がある．例えば，DES や RSA などの使用できるライブラリを用いて xxfs プロトコルを隠蔽する方法や，kerberos などを用いてセッション毎に通信相手を保証する事などが考えられる．このようなセキュリティ面については構造面も含めてもっと改良してゆく余地がある．

今後はシステムの柔軟性を権限・セキュリティに破綻をきたさない限り最大限に引き出す事を考えて開発する．

また，この xxfs についての評価も行なってゆく．

10. おわりに

xxfs は本来新たなファイルシステムを容易に開発し，テストするにはどのような枠組が必要かということを考えるうちに，xxfs のようなものがあればよいと考えに至った．現在私の所属する研究室では幾つかの mobile 環境を意識したネットワークサービスを考案・実装している．本研究はこれらネットワークサービスがファイルシステムを利用するものである場合に有用である．この中で，不安定なネットワークにおけるファイルの同一性を保証する研究がなされているが，これをこのファイルシステムと組み合わせる事でより使用しやすい環境を整える事ができる．

このシステムを利用して多くの有用なシステムが開発される事になるよう，公開を予定している．

謝辞

この論文を書く上で，忙しいのに色々と助言を下さった東 秀明氏・土井祐介氏，また論文を書く環境を壊さないよう気遣ってくれたラップトップコンサルタントの面々と徳田村井楠本中村共同研究室のみなさんに感謝の意を表する．

参考文献

1) SUN microsystems, Inc:
NFS:Network File System Protocol Specification, RFC1094 (1989)
2) Youki Kadobayashi: WWFTP: A filesystem-oriented file transfer protocol, WWFS Research Group (1993)
3) 門林雄基, 山口 英, 宮原秀夫: Internet における資源アクセス装置の提案, 情報処理学会論文誌, Vol.37,No.5 (1996)
4) 門林雄基, 山口 英, 宮原秀夫: インターネットにおける可用性改善のための代替サーバ利用技術, 情報処理学会論文誌,Vol.38,No.2 (1997)
5) 楯岡孝道, 植原啓介, 砂原秀樹, 寺岡文男: PFS: 通信環境に動的に対応するファイルシステム, インターネットコンファレンス'96 論文集, 日本ソフトウェア科学会インターネットテクノロジ研究会／日本 UNIX ユーザ会／ WIDE プロジェクト (1996),pp.81-88
6) David S.H.Rosenthal:
Evolving the Vnode Interface, USENIX Summer Conference, Anaheim, California (1990)
7) John Bloomer, 森島 晃年監訳:
RPC プログラミング, アスキー出版社 (1995)
8) Hal Stern, 倉骨 彰訳, 砂原 秀樹監訳:
NFS&NIS, アスキー出版社 (1992)