第 13 部

広域分散ファイルシステム

第 1 章

DFS-WG の概要

1.1 現状

インタネットでは、ここ数年、gopher や WWW(World Wide Web) に代表される情報共有サービスが重要なサービスなってきている。情報共有サービスは、インタネットを従来の電子メイルのようなコミュニケーションサービス、ログインや FTP のようなリモートアクセスといった通信手段から、世界規模での巨大な情報環境に変えようとしている。もともと、インタネットは学術研究資源を共有することによって研究活動を推進することが目的であったので、このような流れはインタネット本来の目的に合致したものである。さらに、最近ではインタネットに向かって発信される情報も、学術情報に留まらず、コンピュータベンダや出版社などからの商業的な情報などにも広がりをみせている。

このような情報共有サービスは、インタネット本来の目的に合致するとはいえ、実現には、新しいアプリケーションプログラムの開発だけでなく、通信プロトコルやファイル管理レベルでの新しい問題を発生させ、これを解決するための新技術が必要となっている。

WIDE プロジェクトの分散ファイルシステム・ワーキンググループ (DFS-WG) では、インタネット上での分散ファイルシステムについて研究、試作および検討を行なってきた。DFS-WG は、この分散ファイルシステムを従来のオペレーティングシステムのファイル機能の拡張としてとらえるだけでなく、情報共有サービスに代表される分散情報環境のプラットホームとして位置付け、関連する技術の研究を行なってきた。実際に分散ファイルシステムは、情報共有環境の実現手段として有効であり、情報共有環境に必要とされる技術も、分散ファイルのモデルに基づいて検討することにより、問題が明確になってくる。

一方、インタネットの適用領域の広がり、利用する機器や通信メディアのバリエーションの広がりが分散ファイルに対して、以下のような新たな問題を発生させている。

より厳密なアクセス管理とセキュリティ機能の実現

インタネットが扱う情報が従来の公開を前提にしたパブリックな情報から、特定のユーザに限定した情報、価格をもった情報、プライベートな情報なども扱うようになってきたからである。

移動体環境のサポート

移動体通信環境における分散ファイル共有の問題も深刻である。以前から、間欠接続通信環境における分散ファイルの研究は広くおこなわれてきているが、実用レベルのものはまだない。一方で、携帯型計算機が急速に高機能低価格化しており、無線 LAN や携帯電話、広域無線通信の利用も容易になってきているが、これらの装置や通信システムを我々は十分に使いこなしてはいないのが現状である。

低速から超高速まで、幅広い通信帯域での適切なサービスの提供

移動体環境では技術的な問題より経済的な問題、周波数割り当ての問題などで必ずしも高速通信が期待できない。したがって、300 ボーから 9.6k ボー程度の程度の低速でかつ遅延の大きな通信をうまく利用せざるおえない。一方、国内の専用線の価格は急激に下がる方向にあり、また、広域でも光ファイバによる数 100 メガからギガ帯域の超高速通信サービスの利用が可能になりつつある。インタネットの分散ファイルシステムは、このような、低速通信から超高速通信まで、はば広い通信帯域での適切なサービスが提供できるように設計されている必要がある。

1.2 DFS-WG の活動概要と研究テーマ

以上のような観点に基づいて、DFS-WG では以下の点に注目して研究をおこなっている。

分散ファイルのアーキテクチャ

インタネット上の展開する計算機群および、それと 1 対 1 で接続する計算機上のファイル共有のモデルとメカニズムについての検討。現実のインタネット環境では、利用する通信メディアの速度や価格、共有するファイル内の情報の特性によって、利用する通信プロトコルやキャッシュ戦略、書き込み時の同期戦略は異なったものを利用しなければならない。また、NFS や FTP といった既存のファイルシステムとの共存環境も重要である。このような異なったインプリメントを統合的に扱うアーキテクチャを提唱している。

効率の良い情報アクセスの手段の技法

アクセス効率の改善のポイントはキャッシュ戦略である。WG では従来から提案されているキャッシュ戦略を見直し、さらに、インタネット、組織内ネットワーク上の実際のトラフィックを解析したり、WWFS のアクセスパターンを分析することにより、新しいキャッシュ戦略を検討している。キャッシュ戦略の善し悪しは、単に効率を高めるだけで

なく、インタネットのトラフィックを軽減し、限られた通信資源を効果的に利用するうえでも重要である。また、キャッシュ戦略は移動体や低速通信環境においてはアクセス時間を劇的に短縮させるのに役立つ。

ファイル交換のためのプロトコル

ホスト間でファイルの内容を交換するためのプロトコルとして、NFS や FTP が広く用いられている。しかし、これらのプロトコルには技術的に幾つかの欠陥がある。NFS は現在の多くの実装が LAN 環境に特化してチューンナップされているため広域環境ではそのまま利用できない。たとえば、ステートレスでブロック単位のアクセスを基本としているためファイル単位でのキャッシュ戦略の適用が難しい、再送時間の算出がインタネット向きでない、制御メッセージが多くオーバヘッドが大きいなどである。FTP はユーザが利用することを前提にしているため、エラーメッセージやディレクトリー表示の書式が一定でなく計算機処理向きでないといった問題や、仕様上はブロック転送機能が規定されているが、実際にこの機能を実装している FTP サーバが少ないといった問題がある。gopher や WWW で用いられているプロトコルもそれぞれのアプリケーションに特化しており汎用でないといった問題がある。WG ではこれらの問題を解決するための新しい計算機間プロトコルの設計をおこなっている。

情報発見のサポート

ファイルシステムではファイル名が情報を識別するキーとして働く。分散ファイルシステムも、このローカルなファイル名空間をインタネット全体に自然に拡張することが普通に行なわれる。一方で、巨大なファイル名空間を探索したり、必要な情報を検索するには多くの CPU 能力とネットワーク能力を必要とする。この問題を解決するために archie や WAIS は、このような情報検索をサポートするアプリケーションプログラムを利用する方法と、なにかしらの方法で、インタネット上の検索鍵空間とファイル名空間のマッピングをファイルシステムがとる方法の二つが考えられる。後者のアプリーチは従来のアプリケーションプログラムが何の変更もなしに情報検索を行なえる可能性を示している。

携帯計算機における分散ファイルシステム

携帯計算機でも、通常のインタネットに接続された計算機と同等の情報環境を必要とする。最近のように高機能の携帯型計算機や無線通信環境が手軽に入手できるようになると、その要求はさらに高くなってきた。このような環境での技術的なポイントは、移動体からインタネットへのアクセスプロトコル、低速な間欠接続を前提にした最適なキャッシュ戦略、および、この環境を意識したアプリケーションプログラムである。通信プロトコルについては、 VIP(モバイルコンピューティング)WG と協調しながら検討を進めている。

より厳密なアクセス管理とセキュリティ機能の実現

インタネットが扱う情報が従来の公開を前提にしたパブリックな情報から、特定のユーザに限定した情報、価格をもった情報、プライベートな情報なども扱うようになってきたからである。インタネットのような複数の組織が管理するネットワークの場合、全体で精密な情報アクセス管理をするのはむずかしい。広域分散ファイルのように蓄積型の情報システムの場合、キャッシュを含めた情報フロー全体での厳密なアクセス管理は、特に難しい。このような混在管理者環境での、ユーザ認証、アクセス制御のためのメカニズム、キャッシュからの情報漏洩の防止等を行なうための方式について検討を進めている。

実装技術およびサポートツール

高速通信環境での実用的な分散ファイルシステムは性能が重要である。性能を高めるためには上述のキャッシュ戦略や通信プロトコルの改善は当然であるが、キャッシュとして利用するローカルディスクの領域管理、トランザクションを高速で処理するための、オーバヘッドの小さなマルチプロセス/スレッド技術が重要である。これらの技術はオペレーティングシステム技術にも密接に関係している。また、実際に分散ファイルシステムを運用する上で重要なのが、コンシステンシーチェックやバックアップのためのツール (UNIX でいうところの fsck、dump、restore) も必要であり、これらのツールのよりエレガントな実現方法を検討している。

1.3 本報告書の構成

DFS-WG の今年度の報告は、まず、第 2 章で、これまでの WWFS の普及状況および運用事例について詳しく述べ、定量的な評価を通して WWFS 92 モデルの有効性を確認する。さらに、こうした運用・評価を通して得られた WWFS 92 の問題点をプロトコルアーキテクチャ、ネットワークアーキテクチャの二つの面から論じる。こうした運用の結果、WWFS ユーザの爆発的な拡大、および先進的なアプリケーションに対する新たな応用などの成果が得られ、より大規模な運用や、さらに広い応用範囲を考慮した新たなモデルが必要となった。こうした応用分野の広がりを考慮した WWFS 92 の改良モデル WWFS 94 について述べる。第 3 章では WWFS 94 が要求するプロトコル特性を満たし、高い記述能力と新しい技術への柔軟性をもつプロトコル ITP の概要を述べ、それを構成する要素技術、基礎概念を明らかにしたのち、様々な情報システムの統一的な扱い、および既存のプロトコルからのプロトコル変換について述べる。第 4 章では移動型計算機に適した、導入が容易な分散ファイルシステム Personal File System (PFS) について述べる。最後に第 5 章でインタネット上の情報資源発見に関する分析と、情報検索システム HORD について述べる。

第 2 章

WWFS

2.1 はじめに

今日、広域ネットワークの規模の拡大にしたがって、その上で快適な情報アクセスを行なうための機構が必要とされている。

広域ネットワーク環境は、多くの中継ノードと、複数のアクセス点からの均質な情報提供サービスによって特徴づけられる。それらによって、既存の情報アクセスを支援するシステムではアクセス時間の悪化、障害発生時の機能停止などの問題が発生する。

アクセス時間が悪化する理由の一つとして、プロトコル設計上の欠陥が挙げられる。長距離の通信路では一般に遅延の分散が LAN 環境とは比較にならないほど大きく、このような遅延特性をプロトコル設計で考慮していない場合、アクセス時間が極めて悪化する場合がある。

アクセス時間が悪化する別の理由として、最適でない情報アクセスによるトラヒックの増加が挙げられる。クライアントが LAN 環境を前提として設計されている場合、広域ネットワークにおけるサーバの最適配置戦略を有効に利用できない。例えば、ネットワークトポロジを無視した情報アクセスが行なわれた場合、ボトルネックとなる通信路におけるトラヒックが増大し、結果としてアクセス時間が悪化する。また、すべてのクライアントがサーバに対して直接アクセスする場合、複数のクライアント間においてアクセスの重複が起こる。これにより、クライアント、及び、サーバのノード数の増加に対して、データ転送量が爆発的に増加する。この結果、広域ネットワーク環境で提供される情報資源に対して円滑にアクセスできない。

さらに、広域ネットワークでは中継ノード数が多く、またネットワークサービスに必須となる基礎的なネットワークサービス (例えば BIND[152]) が複雑に関連しているため、系全体を考えた場合、障害発生率が高くなる。このため、障害を回避する機構がない場合、システムが機能しなくなる恐れが高い。

これらの問題点を解決するために、DFS WG では 1991 年よりサーバ・クライアント間に csd と呼ばれる中間ファイルサーバを設計・実装した。csd は、従来 LAN 環境で用いられてきたクライアントにおける広域ネットワークの利用を支援し、同時に、複数のクライアントからのファイルアクセス要求を集約し、最適化することを主な目的としている。これを実現するために、csd では情報のキャッシュ、サーバ選択、プロトコル変換、障害回避などの技術を実装している。

csd では複数のクライアント間でのキャッシュ共有を実現しており、これによって情報共有における通信路遅延の影響を減らし、同時にデータ転送量を抑えることができる。

またこの機能とは独立して、csd では地域およびネットワークトポロジに基づいてコスト最小となるサーバを選択する機能を提供し、より大局的なファイルアクセスの最適化を行なっている。これにより、通信路遅延を小さくでき、キャッシュされていない場合におけるアクセス時間を改善する。

また、csd で NFS - FTP プロトコル変換を行なうことで、LAN に適した情報アクセスプロトコル NFS と広域ネットワークに適したプロトコル FTP を効果的に組み合わせることができる。その結果として、広域ネットワーク上の情報を LAN 上の情報と同様に扱うことができ、情報アクセス環境を改善することができる。

さらに、csd において障害発生時にサーバを切替えることで、従来 LAN 環境で用いられてきたソフトウェアを変更することなく、ユーザに対して信頼性の高い情報アクセス環境を提供することができる。

これまで、以上のような機能を実現した csd を核とした分散ファイルシステムモデル WWFS 92 を設計し、このモデルに基づいて実装をすすめてきた。WWFS 92 モデルでは、広域ネットワークにおける情報共有の支援を目的とし、なかでも、ファイルの更新の頻度が比較的稀で、かつファイルの読み出しのみを目的とするような「疎な」情報の共有を対象とする。また、WWFS 92 モデルではボリュームを単位としてインターネット上の情報資源に対するアクセス方法、障害時における代替サーバなどを記述する。

こうした技術により、従来のクライアント・サーバモデルに基づいた場合に発生していたトラヒックを、定常時において約 1/4、初期オーバーヘッドを考慮した場合およそ 1/40 程度に削減できることが明らかになった。

さらに、1993 年には、このような WWFS 92 モデルに基づいた WWFS バージョン 109.5 を広く配布し、奈良先端科学技術大学院大学、慶應義塾大学、Rechenzentrum der Ruhr-Universitaet Bochum, University of Dortmund, University of Stuttgart などにおいて運用されている。リリース 109.5 ではネットワークトポロジを記述したファイルを用意することで、コスト最小となるサーバの選択を近似し、これによってトラヒックの最適化を支援した。このほか、Mule, Mosaic, Chimera, freeWAIS などの UNIX およびインターネット環境における標準的なツールに対するインタフェースを提供し、LAN 環境を前提としたクライアントからも csd を有効に利用できることを実証した。

しかしながら、現在 csd が採用している NFS プロトコルおよび FTP プロトコルの設計と実装にはいくつかの問題点が存在し、応用範囲が限られてしまっている。例えば、csd が相互接続され協調動作するようなサーバ系の構築が困難であり、またセキュリティ機能を利用できないため公開可能なファイルしか扱えないという問題がある。さらに、対象があらかじめファイル長の決められたファイル全体に限定されてしまうといった制限がある。

こうしたプロトコル特性に起因する問題点を解決するため、本研究では情報転送プロトコル ITP (Information Transfer Protocol) を設計し、そのプロトタイプを実装した。ITP は直交性の高い四つの基礎的概念 (コマンド、リプライ、環境、属性) からなり、サーバ

間の協調、様々なプロトコルセマンティクスの統一的な扱いによる異種ファイルシステム/情報システムの接続、および、ネゴシエーションを基本としたプロトコルの拡張性を特徴とする。

本研究では、協調サーバ系の構築、csd キャッシュ技術の適用分野の拡大、異種ファイルシステムの相互接続、セキュリティ技術の導入、次世代トランスポートプロトコルの導入などを容易に行なうための枠組として ITP の設計・実装をすすめている。

ITP を用いることで、csd によるサーバ系を構築することが可能となり、広域ネットワークにおけるファイルアクセスを効率化できる。また、ITP では様々なプロトコルセマンティクスを統一的に扱うことで、これまでファイルシステム以外の技術を用いて構成されていた応用分野に対しても csd のキャッシュ技術を応用できる。また ITP が提供するプロトコル拡張のための枠組によって、公開鍵暗号、ディジタル署名などのセキュリティ技術の発展や、次世代の高速ネットワークにおけるトランスポートプロトコルを容易に導入できる。ITP については次章で詳しく述べる。

2.2　WWFS の運用

1991 年に提案した WWFS 92 モデルおよびその可能性を検証し、実際にインターネット上で日常的に用いることができるシステムを構築することを目標として、数百台からなる UNIX ワークステーション環境での WWFS の運用、企業内インターネットにおける運用、ならびにインターネットにおける WWFS の普及活動をおこなってきた。

本章では、こうした WWFS の運用事例および普及状況について詳しく述べ、定量的な評価を通して、WWFS 92 モデルの有効性を確認する。

2.2.1　これまでの経過

1992 年より、WWFS のリリースエンジニアリングおよび研究活動と並行して WWFS の運用を支援する様々なツール群の開発をすすめた。その成果として、1993 年 11 月 7 日に NCSA Mosaic[153] の WWFS 対応バージョン NCSA Mosaic/WWFS が完成し、1993 年 11 月 16 日に HTTP[154], Gopher[142] などのプロトコルからのデータ変換ツールが完成した。1993 年 11 月 24 日に、HTTP, Gopher から WWFS への変換サーバ wwfs.aist-nara.ac.jp の運用を開始し、それと同時に、こうしたハイパーテキスト GUI を統合した WWFS バージョン beta 109.0 をリリースした。wwfs.aist-nara.ac.jp では US National Center for Supercomputing Applications[155] や Global Network Navigator[156] などのサーバ上の、従来 HTTP でしかアクセスできなかった情報を WWFS を用いてアクセスできるようにプロトコルおよび URL[157] の変換を行なう。このようにして変換されたマルチメディアファイル群を wwfs.aist-nara から anonymous ftp を用いてアクセスできるようにし、NCSA Mosaic/WWFS と組み合わせることで迅速なアクセスを可能にした。

以上で述べたような周辺技術の確立とほぼ並行して、インターネット全体においても HTTP, Gopher といった情報アクセスツールの発生するトラヒックが爆発的に拡大し、こ

期間	全アクセス	WWFS	WWFS 利用率 (%)
1993/6/16 – 1993/11/7	364,408,856	291,168,212	80
1993/11/12 – 1994/1/31	149,233,436	107,631,696	72

表 2.1: 大阪大学の広域ファイルサーバ ftp.center.osaka-u.ac.jp (FCO) に対する、奈良先端科学技術大学院大学からのファイルアクセスの合計転送バイト数と、そこで WWFS によるアクセスが占める割合を示した。奈良先端大から FCO に対するアクセスのうち約 8 割近くが WWFS を用いたアクセスであることがわかる。

うした状況を改善するための persistent shared cache サーバの重要性が認識されはじめた。こうした背景から、1993 年 12 月 20 日に csd による共有キャッシュ、csd による NFS - FTP プロトコル変換、地域およびトポロジに基づくサーバ選択 (GTR)、ボリュームによるインターネット上の資源記述、UNIX から派生した多くの OS のサポート、HTTP, Gopher プロトコルからの移行ツール、NCSA Mosaic/WWFS などを統合したバージョン 109.5 をリリースした。

また、1993 年 12 月 23 日には freeWAIS[158] を用いた情報検索と、WWFS を用いた効率的な情報アクセスの組合せが可能となり、この技術を用いて、wwfs.aist-nara において NCSA, GNN 等のマルチメディアファイル群の freeWAIS を用いた検索サービスを開始した。

さらに、1994 年 1 月 2 日には、X Window System の一般的な構成で Motif ライセンスを必要とせずに動作するマルチメディアデータブラウザ Chimera[159] が WWFS に対応し、より広い環境において WWFS への GUI が利用できるようになった。

2.2.2 奈良先端科学技術大学院大学における運用と評価

奈良先端科学技術大学院大学では、1993 年 4 月の学生受け入れと並行して全学統合情報処理環境「曼陀羅」におけるネットワークサービスの構築を開始し、ディスク・ネットワークの有効利用ならびにインターネット上の情報資源へのより迅速で円滑なアクセスを目標として、筆者が中心となって 1993 年 6 月に全学規模の WWFS Pilot Project を開始した。約一ヶ月の実験運用の後、1993 年 7 月 19 日をもって WWFS は全学ネットワークにおける基盤サービスとして位置づけられ、その定常的な運用を開始した。1993 年 11 月 23 日には NCSA Mosaic/WWFS が全学で利用可能になり、GUI を用いた情報アクセス環境を提供した。

こうした運用の結果、大阪大学の広域ファイルサーバ ftp.center.osaka-u.ac.jp (以下 FCO) に対する奈良先端大からの FTP サーバ利用のうち約 8 割近くを WWFS を用いたアクセスが占めるようになった。このデータは、FCO において奈良先端大からのファイルアクセスを 1993 年 6 月 16 日から 1994 年 1 月 31 日まで、約七ヶ月以上にわたって観測した記録によるものである。表 2.1 にその具体的なデータを示した。

観測点	全アクセス	/wwfs	/wwfs 利用率 (%)
relay	2,159,545,557	608,189,284	28
firewall	1,990,971,919	203,193,847	10

表 2.2: オムロンにおける anonymous ftp によるファイル転送について、ファイルシステム全体に対する合計転送バイト数と、/wwfs 以下の WWFS が提供するファイルシステム全体に対する合計転送バイト数、および /wwfs へのアクセスが全体に占める割合を示した。このデータは、オムロン社内の中継マシン (relay) の ftp サーバおよびインターネットと社内ネットワークとの中継ノード (firewall) の ftp サーバに対する anonymous ftp アクセスを、1993 年 11 月 1 日から 1994 年 1 月 31 日までの三ヶ月にわたって観測したものである。

2.2.3　企業内インターネットにおける運用と評価

奈良先端大における全学 Pilot Project とほぼ並行して、1993 年 7 月半ばから、WIDE プロジェクトにおける共同研究組織であるオムロン株式会社において WWFS の運用実験が開始された。オムロンでは以下のような多くの利点を考慮し、WWFS を導入した。

1. socks[160] やパケットフィルタリングなどを用意しなくても firewall における csd の運用だけで社内からの資源アクセスの利便性を増すことができる、

2. 何度も telnet, ftp する必要がなくなりディスクやネットワークを有効利用できる、

3. 運用上の無駄な手間を省くことができる。

オムロンでは csd を firewall において運用し、WWFS が提供するファイルシステムを更に anonymous ftp ディレクトリ /wwfs において公開している。こうすることで、64Kbps などの低速なリンクによって firewall に接続された企業ネットワーク内部の遠隔サイトからも、インターネット上の資源に対して効率的なアクセスが可能となった。

このような anonymous ftp を用いた WWFS アクセスを 1993 年 11 月 1 日から 1994 年 1 月 31 日まで三ヶ月にわたって観測した結果、社内の中継マシンにおける ftp アクセスの約 3 割、および firewall における ftp アクセスの約 1 割が /wwfs に対するものであることがわかった。表 2.2 に具体的なデータを示した。

2.2.4　日本国内における運用と評価

奈良先端大やオムロンにおける大規模な運用と並行して、大阪大学大型計算機センターの協力のもとに筆者が中心となって広域ファイルサーバ FCO の運用を開始し、WWFS を中心とした先進的なネットワークサービスの利用推進を目的としたサービス構築をおこなってきた。

1993/6/16 – 1993/11/7 の期間中における WWFS によるアクセス

ドメイン名	転送ファイル数	転送バイト数	構成率 (%)
aist-nara.ac.jp	2,645	291,168,212	54.73
omron.co.jp	1,265	182,952,682	34.39
osaka-u.ac.jp	237	39,637,308	7.45
others	144	18,244,535	3.43

1993/11/12 – 1994/1/31 の期間中における WWFS によるアクセス

ドメイン名	転送ファイル数	転送バイト数	構成率 (%)
omron.co.jp	2,451	353,633,119	40.56
unresolved	947	116,349,589	13.35
tokai-ic.or.jp	1,450	105,688,215	12.12
osaka-u.ac.jp	387	104,209,076	11.95
toshiba.co.jp	1,548	73,534,485	8.43
aist-nara.ac.jp	674	62,192,491	7.13
aoyama.ac.jp	158	11,050,292	1.27
ipa.go.jp	164	10,624,828	1.22
others	411	34,518,181	3.97

表 2.3: FCO において実験参加サイトごとの転送ファイル数、転送バイト数を観測し、各サイトが WWFS アクセス全体に占める割合を示した。ここで、構成率 1% に満たないサイトは others に分類し、ドメイン名を逆引きできないファイル転送は unresolved に分類した。1993 年 11 月上旬を境として、WWFS アクセスに占める実験参加サイトの構成率が大きく変化していることがわかる。

また、奈良先端大やオムロンにおける運用事例をもとに、WIDE プロジェクトにおけるその他のいくつかの共同研究組織においても WWFS の運用実験が開始され、Mosaic, Chimera, freeWAIS といった先進的なアプリケーションとの共用により WWFS ユーザの拡大をみた。表 2.3 に FCO における観測データを示した。

こうした WIDE プロジェクトを中心とした普及活動の成果として、FCO に対するファイルアクセスのうち約 1/6 を WWFS が占めるようになった。このデータは、FCO において 1993 年 6 月 16 日から 1994 年 1 月 31 日まで、約七ヶ月以上にわたるアクセスを集計した結果得られたものである。表 2.4 にその具体的なデータを示した。

また、WWFS で実現された persistent shared cache によって、mirror[161], ftpsync に代表されるようなアーカイブ全体の複製をおこなう従来技術に比べて劇的なネットワーク利用効率の改善がみられた。FCO は大阪府立大学 総合情報センター、大阪大学 基礎工学部 情報工学科、オムロン株式会社の FTP サーバに対するマスタサーバとしても機能している (図 2.1) が、サーバ立ち上げにおけるオーバーヘッドを比較した場合、1993 年 10 月 30 日に開始された大阪府立大学からの mirror アクセスにくらべて、奈良先端

期間	FTP + WWFS	WWFS	WWFS 利用率 (%)
1993/6/16 – 1993/11/7	4,990,321,128	532,002,737	11
1993/11/12 – 1994/1/31	5,243,649,881	871,800,276	17

表 2.4: WIDE プロジェクトを中心とした普及活動の成果として、FCO に対するファイルアクセスのうち約 1/6 を WWFS が占めるようになった。ここでは mirror, ftpsync を除いた、WWFS を含む通常の FTP アクセスの合計転送バイト数と、WWFS によるアクセスのみの合計転送バイト数、および通常の FTP アクセスに占める WWFS アクセスの割合を示した。

大、オムロンからの WWFS アクセスは約 1/40 のトラヒックに抑えられるという結果が得られた。その具体的なデータを表 2.5 に示す。また、定常的な運用においても、オムロン、東海インターネットワーク協議会、大阪大学、奈良先端大の主要な 4 サイトによる WWFS アクセスは、大阪府立大学の単一サイトによる mirror アクセスとほぼ同程度のトラヒックしか発生しないという結果が得られた。その具体的なデータを表 2.6 に示した。

2.2.5　インターネットにおける普及活動

前節までに述べたような、1992 年からの WWFS の運用と実験参加サイトからの積極的なフィードバックを集積することで、インターネット上でのより大規模な運用が可能となった。こうした背景から、1993 年 12 月 20 日に WWFS バージョン 109.5 をリリースした。パッケージは wwfs.aist-nara.ac.jp の anonymous ftp ディレクトリ /WWFS において公開し、USENET のニュースグループ comp.infosystems, comp.archives.admin, comp.archives などを通じてインターネットコミュニティに広く通知された。その結果、20 日から 21 日にかけて、パッケージ、関連ドキュメントなどが合計 64Mbytes 転送された。

また、1994 年 1 月 17 日に USENET のニュースグループ comp.sources.misc に WWFS のパッケージがポストされ、その結果イギリス、ドイツ、台湾、米国などにおいても WWFS の運用が開始された。

1993 年 11 月 24 日から 1994 年 2 月 12 日の期間中、wwfs.aist-nara.ac.jp:/WWFS に対するアクセスは合計 19 か国、約 200 サイト、転送量 194Mbytes におよび、うち 41% は WWFS バージョン 109.5 に対するアクセスであった。これは、NCTUCCCA.edu.tw (Campus Computer Communication Association, National Chiao Tung University, Taiwan) および ftp.doc.ic.ac.uk (SunSITE Northern Europe, Department of Computing, Imperial College, London, UK) の 2 サイトにおいて設定された mirror サーバからのアクセスも含む。この期間中観測されたデータをトップドメイン別およびディレクトリ別に分類し、表 2.8 および表 2.9 に示した。

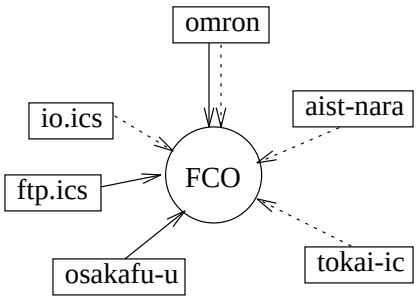

図 2.1: FCO は大阪府立大学 総合情報センター (osakafu-u)、大阪大学 基礎工学部 情報工学科 (ftp.ics)、オムロン株式会社の FTP サーバに対するマスタサーバとしても機能している。この図において、実線は mirror によるアクセス、点線は WWFS によるアクセスを示す。

アクセス方法	転送ファイル数	転送バイト数	転送バイト数/日
mirror	6,335	417,362,516	139,120,839
ftpsync	5,872	1,519,812,220	13,215,758
wwfs	3,941	466,130,334	3,820,740
others	23,619	4,458,318,391	29,722,123

表 2.5: FCO におけるファイル転送をアクセス方法別に集計し、mirror と WWFS のサーバ立ち上げにおけるオーバーヘッドを比較した。この観測期間中 (1993/6/16 – 1993/11/7)、大阪府立大学が 1993 年 10 月 30 日から mirror の運用を開始し、奈良先端大およびオムロンが 1993 年 7 月 19 日から WWFS の運用を開始した。大阪府立大学は /GNU, /X-contrib, /InterViews, /MSDOS, /rfc などのボリュームを mirror によってすべて転送しているが、奈良先端大およびオムロンでは WWFS によって FCO の全ボリュームのうち必要なファイルだけを要求時に転送している。こうしたファイルアクセス戦略の違いから、WWFS が発生するトラヒックは mirror と比較して約 1/40 に抑えられた。

アクセス方法	転送ファイル数	転送バイト数	転送バイト数/日
mirror	7,570	746,702,533	13,100,044
ftpsync	1,944	432,433,754	6,177,625
wwfs	8,190	871,800,276	11,942,470
others	31,503	4,371,849,605	59,079,049

表 2.6: FCO におけるファイル転送をアクセス方法別に集計し、mirror と WWFS の定常時のトラヒックを比較した。この観測期間中 (1993/11/12 – 1994/1/31)、東芝 研究開発センター 情報・通信システム研究所、東海インターネットワーク協議会、大阪大学 基礎工学部 情報工学科、大阪大学 情報システム工学科において WWFS の運用が開始された。こうした事実を考慮しても、主要な 4 サイトによる WWFS アクセスは大阪府立大学の単一サイトによる mirror とほぼ同程度のトラヒックしか発生しないことがわかる。

ボリューム名	転送バイト数	構成率 (%)
/win3	282,293,000	32.38
/GNU	234,261,143	26.87
/sun-info	112,456,905	12.90
/X-contrib	84,680,926	9.71
/info-mac	49,481,039	5.68
/rfc	33,858,962	3.88
/tcl-tk	24,068,626	2.76
/internet-drafts	9,678,331	1.11

表 2.7: 1993 年 11 月 12 日 – 1994 年 1 月 31 日の期間中における WWFS によるアクセス内容をボリュームごとに集計し、転送バイト数の構成率 (総転送バイト数に対する割合) が 1% をこえるものについて、多いものから順に並べた。

トップドメイン名	転送ファイル数	転送バイト数	構成率 (files, %)	構成率 (bytes, %)
de	299	12,167,038	3.92	5.97
tw	685	3,813,728	8.99	1.87
uk	52	2,757,469	0.68	1.35
com	341	12,050,375	4.48	5.91
edu	191	12,019,949	2.51	5.89
gov	104	22,720,233	1.36	11.14
jp	5,004	110,239,868	65.67	54.05

表 2.8: 1993 年 11 月 24 日から 1994 年 2 月 12 日の期間中における wwfs.aist-nara.ac.jp に対する anonymous ftp のうち、/WWFS ディレクトリ以下についてドメイン毎に集計し、転送バイト数の構成率 1% を超えるトップドメインを抜粋した。

ディレクトリ	転送ファイル数	転送バイト数	構成率 (files, %)	構成率 (bytes, %)
/WWFS/binaries	254	85,917,657	3.33	42.12
/WWFS/dist	494	84,592,308	6.48	41.47
/WWFS/web	2,317	14,898,846	30.41	7.30
/WWFS/util	93	14,603,019	1.22	7.16
/WWFS/papers	27	1,967,805	0.35	0.96
/WWFS/vol	4,199	997,970	55.10	0.49
/WWFS/irc	86	599,650	1.13	0.29
/WWFS/README*	150	388,024	1.97	0.19

表 2.9: 1993 年 11 月 24 日から 1994 年 2 月 12 日の期間中における wwfs.aist-nara.ac.jp に対する anonymous ftp のうち、/WWFS ディレクトリ以下についてディレクトリごとに集計した。

2.3　WWFS 92 の評価

前章で述べたような運用・評価を行なっていく中で、多種多様なユーザ層からの様々な提案、多様なネットワーク構成における運用、長期間にわたる WWFS の運用経験などを通して、さまざまなことが明らかになった。本章では、WWFS 92 に基づいて構成された csd を通して得られたこうした経験をプロトコルアーキテクチャ、ネットワークアーキテクチャの二つの側面から論じる。

2.3.1　プロトコルアーキテクチャ

WWFS 92 モデルの実装をすすめていく途中で、WWFS 92 で採用したプロトコルの欠陥、サーバ・クライアントにおけるプロトコル実装の問題点などが明らかにされた。本節では、特に WWFS 92 との親和性という観点からこうしたプロトコルの問題点を明らかにし、それらを克服するためにとった回避手法についても言及する。

NFS の問題点

NFS[124] は Sun RPC[162] を基盤として構築した LAN 環境のための分散ファイルシステムである。NFS の問題点は NFS プロトコルの問題点と NFS 実装の問題点に区別できる。

NFS プロトコルは広域ネットワークでは実用的な速度で動作しないが、その理由は NFS プロトコル自体の問題点と実装上の欠陥が複雑に絡み合っている。NFS プロトコル仕様書はトランスポートプロトコルとして TCP の使用を禁じていないが、今日ひろく用いられている NFS の実装は UDP のみをサポートしている。また、多くの NFS 実装では UDP パケットの再送を一定間隔で行なうという欠陥を含んでおり、このことと UDP への限定という二つの実装上の欠点が重なって、広域ネットワークにおける NFS 利用を妨げる主な要因となっている。

こうした問題に対し、トランスポート層に TCP を用いる、サブネットアドレスが異なるネットワーク間において NFS を利用する場合は binary backoff 再送アルゴリズムを用いる、といった解決策が提案されているが、NFS プロトコルでは通信アーキテクチャとして RPC を採用しており、またプロトコルの粒度が非常に小さく設計されているため、単純な処理をするだけでも複数の RPC request を発行しなければならず、広域ネットワークのような RTT (round trip time) が大きい通信路において必要以上に大きな遅延を生じてしまう、という問題点がある。このプロトコルの欠陥は、相対的に RTT が大きい高速ネットワークのような通信路においても同様に問題となりうる。

このような理由から、WWFS 92 では NFS - FTP プロトコル変換を行ない、NFS の問題点を回避することを目指した。しかしながら、多くの NFS クライアントの実装では、ファイル転送が行なわれている間、UNIX カーネル内の NFS クライアントスレッドを待たせておくことは望ましくない。従来のファイルシステムとの利用感の一貫性および意

味的な統一性という点で、広域ネットワークに対してファイルアクセスを行なっている間でもクライアントが待つことが自然だと考えられるが、csd でこのような実装をおこなった場合、いくつかの問題が生じる。具体的には、一定間隔で再送されるパケットを無視する処理をしなければならない、複数のセッションで csd に対してアクセスし、一定時間以上 NFS クライアントスレッドを待たせておくと利用可能なカーネルスレッドがなくなり NFS 全体が応答しなくなる、といった危険な障害がある。このような理由から、こうした実装を広く採用することは一般に望ましくないと思われる。また、NFS プロトコル自体もこうした状況を想定していないため、いくつかのプロトコル設計上の欠陥がある。具体的には、クライアントに対して ICMP source quench のような再送アルゴリズムを制御するコマンドがない、クライアントとサーバが一度接続を切り離すといったような非同期のインタラクションを考慮していない、といった欠点がある。

以上で述べたような、NFS クライアントを待たせることができないという NFS プロトコル/実装の問題点を、カーネルに手を加えることなしに回避し、必要最小限度のプロトコルオーバーヘッドで補うために、外部プロトコル UIP (user interface protocol) を開発し、libc で提供されているファイルシステムインタフェースとアプリケーションの間に libww (WWFS client library) を挿入することで通常の NFS との利用感の一貫性および意味的な統一性を実現した。

FTP の問題点

FTP[115] は TCP の上で動作するファイル転送のためのプロトコルである。FTP は異なるアーキテクチャ・OS・ファイルシステム間におけるファイル転送を主眼として設計されたが、その応用分野に関して長期的な展望を欠いたままで設計・実装がすすめられた。このため、WWFS 92 モデルに基づいて csd をより緻密に実装していく過程で、多様な OS、様々なファイルシステムの運用形態、FTP の実装の細かな違い、FTP サーバの拡張機能などをサポートすることができず、インターネット上に広く存在する様々な FTP サーバすべてを WWFS を用いて統一的にアクセスすることは不可能であることが明らかになった。また、もっとも一般的に用いられている UNIX 上の FTP サーバはファイルシステムが持つファイル属性情報のいくつかを欠落させるため、NFS サーバとまったく同様にアクセスすることが可能な NFS - FTP プロトコル変換サーバの実現は非常に困難であることが明らかになった。

具体的には、FTP プロトコルには以下のようなプロトコル設計上の問題点がある。

1. STAT コマンドの出力形式が規定されていないため、OS ごとにディレクトリ一覧の出力形式が異なり、それらすべてを csd でサポートすることは事実上不可能である。また、パス名の標準形式も規定されていないため、OS ごとに異なるパス名の表現形式をサポートすることができない。

2. FTP サーバからの応答が機械可読でなく、サーバ側におけるエラー状態を判別することができない。

3. anonymous ftp でログインした場合、サーバ側におけるクライアントのユーザ ID, グループ ID を知ることができず、その結果、ファイルアクセスの可否をクライアント側だけで評価することができない。

4. UNIX ファイルシステムにおいてファイルが hard link されている場合などに、異なるファイル名をもつファイルの同一性を確認できない。また、異なるファイルサーバにおいて同一ファイル名をもつ二つのファイルの同一性を確認するためにはファイル全体を転送しなければならず、発生するトラヒックの面からみて現実的でない。

5. 1 から 4 までの欠陥を補うために FTP プロトコルに機能拡張が付加されている場合においても、そのような拡張機能の有無を調べる方法が規定されていないため、それらを効率的に利用することができない。

以上のような問題点を回避するため、現在の csd では以下のような対策をとった。

- STAT コマンドの出力形式としては UNIX の ls -l 形式のみをサポートし、UNIX 以外の様々な OS に対してもこの形式による STAT 出力を要請する。

- FTP サーバからの応答のうち、text string 部分を無視して行頭の 3 オクテットに記述されている完了コード /エラーコードのみを判別し、エラーを検出した場合はすべて NFS のエラーコード NFSERR_IO (I/O error) に変換する。

- ファイルの r ビットのいずれかが立っていた場合、ファイルにアクセスできるものとして NFS のパーミッション 0444 (読み出し可) に変換し、実際に FTP ファイルアクセスを行なってエラーとなった場合のみ I/O error とする。

- hard link されているファイルに関して例外処理を行なわず、すべて異なるファイルとして扱う。

- 拡張機能の存在を仮定せず、それらをいっさい使用しない。

以上で挙げたような対策によって、多くの UNIX で広く用いられている wu-ftpd[163] などの ftpd に対して NFS - FTP プロトコル変換を適用することが可能となった。しかしながら、このような対策にはそれぞれ問題点が残されている。

1. ls -l 形式をサポートしていない ftpd に対して、NFS - FTP 変換を適用できない。

2. ネットワークおよびサーバにおいて発生する様々な障害がすべて I/O error として表現されてしまい、ユーザあるいは csd 管理者がエラー状況を正確に把握できない。

3. NFS ではパーミッション 0444 として表現されているファイルが実際には読み出すことができないという奇妙な振舞いが生じる。

4. hard link されているファイルが多く存在するとき、キャッシュヒット率が著しく低下する。

5. wu-ftpd などの広く用いられている ftpd で採用されている拡張機能を有効に利用できない。

2.3.2 ネットワークアーキテクチャ

WWFS 92 モデルを実装し、実際にインターネット上で日常的に用いることができるシステムを構築していく過程で、前節で述べたようなプロトコルアーキテクチャの問題と並行して、解決しなければならない技術的障害が幾つか発生した。
それらは以下のような四つの問題に大別できる。

1. 広域ネットワークにおける構成要素のエラー率の高さからくる問題、
2. csd の大規模な運用からくる問題、
3. 現在の長期的キャッシュ手法と応用分野の不一致からくる問題、
4. 企業がインターネットに接続するときのセキュリティ要求からくる問題。

本節ではこうした問題に対する解決策や回避手法について論じる。

広域ネットワークにおける要求

広域ネットワークにおけるネットワークサービスは、他の様々なネットワークサービスに依存し、多層構造をなしており、それらの一部が動作しなかった場合、アプリケーション層まで詳細なエラー情報が伝えられることは稀である。このような理由から、csd では複数のネットワークサービス構成要素について積極的に障害を検出し、アプリケーション層だけで得られる抽象化されたエラー情報のほかに、詳細なエラー情報についても収集を行なっている。具体的には、csd において以下のようなエラー収集を行なっている。

- ICMP パケットの監視
  ネットワーク層における障害状況を正確に把握し、迅速なサーバ切替えを行なうために ICMP パケットを監視する。
- FTP コネクションの監視
  ftpd の依存するネットワークサービスが機能しなくなり、ftpd が応答しなくなった時のために、FTP コネクションを監視し、タイマを用いて応答していない FTP コネクションを積極的に検出する。

このようにしてエラー情報を収集することにより、ネットワーク層およびアプリケーション層で生じた障害を詳細に知ることができ、効果的なエラー対策が可能となった。

現在の csd アーキテクチャの問題点

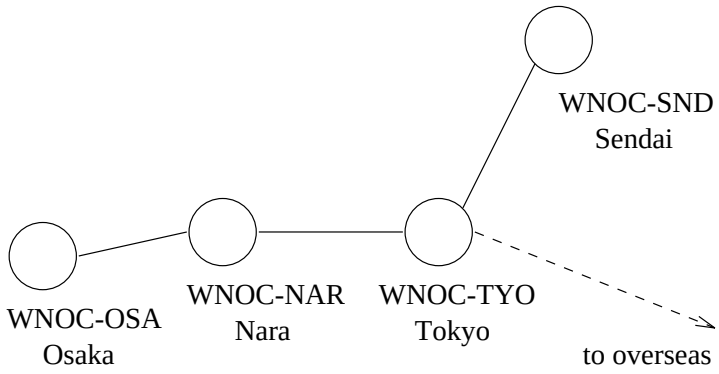

図 2.2: WIDE Internet のようなトポロジをもつ広域ネットワーク上で WWFS を広範囲にわたって運用した場合、海外から転送されたファイルをいったん東京でキャッシュし、各ノードに対しては東京を経由して転送したほうが、それぞれのノードから直接海外にアクセスした場合に比べて 1/4 のデータ転送量ですむ。

WWFS 92 が普及するにつれて、csd のより広範囲/大規模な運用形態における問題点が明らかになってきた。

csd を図 2.2 のようなトポロジをもつ広域ネットワーク上で広範囲にわたって運用した場合、海外から転送されたファイルをいったん東京でキャッシュし、各ノードに対しては東京を経由して転送したほうが、それぞれのノードから直接海外にアクセスした場合に比べて 1/4 のデータ転送量ですむ。

また、大阪大学や慶應義塾大学などでは、近隣に位置する複数のサイトで csd が運用されており、それらの間での効率的なキャッシュ共有が望まれている。これに対する解決策として、これら複数の csd を図 2.3 のように完全グラフ状に結合し、それらの間で何らかのキャッシュ共有プロトコルを用いて効率化をはかることができると考えられる。また、このような水平型のサーバ協調アーキテクチャでは、マスタ・スレーブ型と比べてキャッシュミス時のアクセス遅延を改善でき、マスタサーバで必要とされるような大規模なディスクを必要としないという利点がある。

現在の長期的キャッシュ戦略の問題点

現在 csd で採用しているキャッシュ戦略は、「疎な」情報の共有を前提とした単純なものであるが、いくつかの FTP の応用分野ではこれとは異なるキャッシュ戦略を用いる必要があり、問題となっている。

具体的には、衛星から転送される気象データ、The Voice of America[164] などのリアルタイム性の高いニュースソース、ホームディレクトリやプロジェクト共同作業ディレクトリなどといった比較的変化の早いファイル群を csd を使ってアクセスするときに現在の csd で用いられている長期的キャッシュ戦略を用いた場合、キャッシュの一貫性がと

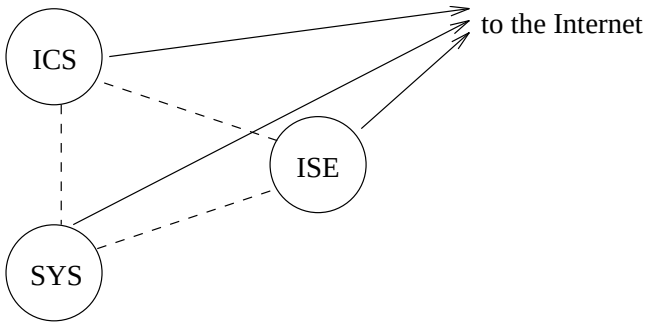

図 2.3: 大阪大学では基礎工学部 情報工学科 (ICS)、システム工学科 (SYS)、工学部 情報システム工学科 (ISE) において csd が実験運用されている。このような場合、これらを図のように完全グラフ状に結合し、それらの間で何らかのキャッシュ共有プロトコルを用いることで広域ネットワークへのアクセスを効率化でき、同時に階層化キャッシュに比べてアクセス遅延を改善できると考えられる。

れなくなるという問題がある。

現在 csd で用いているキャッシュ戦略はアクセス時間の最小化を目標とし、キャッシュの一貫性をある程度犠牲にしている。これに対し、キャッシュ戦略をボリュームごとに変更できるようにし、キャッシュの一貫性とアクセス遅延のバランスを管理者もしくはユーザが選べるようにするという改善策が考えられる。

企業ネットワークにおける要求

企業ネットワークにおいて WWFS のようなネットワークサービスを運用する場合、外部からの侵入や情報の漏洩といったセキュリティ上の問題を引き起こす可能性を最小限度に抑えることが要求される。通常、企業ネットワークではこうした目標を実現するため、インターネットと企業内ネットワークの接点を一台のマシンにとどめ、情報の中継を行なう以外の、不必要な機能を停止するといった対策がとられる。こういったセキュリティ技術、およびそれを実装したマシンなどを総称して一般に firewall と呼ばれているが、firewall 周辺のネットワークサービス要素は厳しいセキュリティ基準を満たすために特殊な設定がなされており、WWFS のような firewall 上で動作する情報の中継機能をもつソフトウェアは、そのような特殊な環境においても、環境の差異を吸収して動作することが求められる。

firewall は、複数のインタフェースを持っているにもかかわらず IP 層におけるパケットフォワーディングを禁止していることが多く、そのような環境で動作するネットワークソフトウェアを作成する際は特に注意が必要である。peer entity に対してネットワークアドレスを伝える場合、firewall の内側・外側を区別し、peer entity から到達できるインタフェースを正しく選び、そのアドレスを伝える必要がある。例えば firewall の外

側と FTP で接続しているときに、PORT コマンドにおいて、引数に内側のアドレスを指定するとファイルやディレクトリを転送できないので注意が必要である。また、BIND 4.9[152] を用いている場合、複数ある DNS A レコードからランダムに一つが返されることがあり、そのアドレスに対して到達できないことがあるので注意が必要である。これに対して、Berkeley sendmail release 8[165] で行なわれているように、すべてのアドレスに対して接続を試みる、あるいは、IP forwarding を禁止しているマシンについては shuffling A record 機能の利用を禁止する、といった解決策が考えられる。

また企業ネットワークでは、ホスト設定情報を隠蔽し、外部からの侵入を防ぐため、内側と外側のネームサーバを分離し、企業内部のホスト設定情報を内向きネームサーバ、インターネットに直接接続しているホストに関する設定情報を外向きネームサーバに記述するといったことが行なわれる場合がある。こうした環境で、企業内部から firewall マシン上で提供されているサービスを利用する場合、ホスト名を指定することができないため、各種の設定ファイルについて、ホスト名だけでなく IP アドレスによる指定もできるようにしておく必要がある。

さらに、企業ネットワークではパケットフィルタリング技術を用いて外部からの TCP コネクションを受け付けない、セキュリティ的に問題となる可能性の高い portmap, rexec などのポートに対するアクセス要求を通さない、といったセキュリティ基準を実装していることが多い。こうした技術を実装した場合、外部の peer entity に対して TCP コネクションの開始を要求する FTP のようなプロトコルは、外部からの侵入と区別できないため望ましくない。したがって、パケットフィルタリング技術の観点から、FTP のようなプロトコルにおけるデータ転送においてもクライアントが TCP コネクションを開始することが要請される。

2.4　新たな応用分野・利用形態 — WWFS 94

インターネットにおけるユーザ層の爆発的な拡大、および先進的なアプリケーションによる新たな応用などにより、WWFS は単一クラスタにおける「疎な」情報の共有という当初の目的を離れ、より多様な利用形態を考慮し、WWFS の広範囲な利用を前提とした情報配送システムとしての役割を期待されるようになった。

WWFS の新たな利用形態としては、複数のクラスタからなる大規模なネットワーク環境における「密な」情報の共有や、広域網におけるデータ転送を効率化するためのマルチレベルキャッシュ、大規模な科学技術データの効率的な取り扱い、画像や音声データなどのサービス品質などを考慮した情報配送系、などが挙げられる。

本章では、こうした新たな応用分野において効率的な情報配送を行なうためのメカニズムとして WWFS を捉え、ユーザからの様々な要求を整理し、それをもとに WWFS 92 モデルに対して改良を行なう。以下、こうして得られた改良モデルを WWFS 94 と呼ぶ。

また本章では、WWFS 94 モデルにおけるプロトコルに対する要求を明らかにし、既存の主な情報転送プロトコルをこうした評価基準にあてはめ、WWFS 94 モデルとの親和性について論じる。

2.4.1 新たな応用分野からの要求

今日のインターネットでは、The Voice of America, 気象データなどの比較的リアルタイム性の高いデータが Mosaic, Gopher などといったエンドユーザ向けツールを用いて日常的に利用されている。大阪大学大型計算機センターにおいて、1993 年 11 月 12 日から 1994 年 1 月 22 日の約二カ月半にわたって測定した統計データでは、Gopher によるアクセスのうち 78% は気象データに対するものであった。インターネットでは、今後もこうした実時間性の高いメディアとしての性格をもつ情報が増えてくると予想され、こうした情報についても WWFS 92 で成功をみたような効率的なキャッシュ手法が有効であると考えられる。このような理由から、広域ネットワークにおいて実時間性の高い情報を効率的にキャッシュできるようなシステムおよびプロトコルが求められている。

また、そのような共有度の高い情報に対しては、複数の地理的に離れたサイトからのアクセスが観測されており、こうした情報を階層化キャッシュなどの手法を用いて広域ネットワーク上で効率的に転送することが重要であると考えられる。例えば、end - end でキャッシュをまったく行なわない状況において、400Mbytes の MPEG 動画ファイルを 100 台のクライアントから転送した場合、総データ転送量は 40Gbytes、クライアント 10 台を単位とする 10 クラスタにおいて各々キャッシュを行なった場合の総データ転送量は 4Gbytes、クラスタを階層化し、サーバへのアクセスを 1 クラスタに集約するような階層化キャッシュを行なった場合の総データ転送量は 400Mbytes であり、伝送路として OC-12 (622Mbps) の回線を用いた場合でも、それぞれの理想的な情報転送時間は順に 514 秒、51 秒、5 秒となり、劇的な改善がみられることがわかる。また、画像や音声などの大規模データの普及にともない、このようにして重複したアクセスを集約し、トラヒックを圧縮することで、例えば日米間の国際リンクなどのコストの高い回線を有効利用することができる技術が求められている。

さらに、そのような画像や音声などの大規模データをアクセスする場合、求めるデータを見つけるためにいくつかのデータを再生してみることが多い。広域ネットワークでは、このようなナビゲーションにかかる時間を短縮するため、データの品質を落して再生することが一般的になると予想される。例えば、528Kbytes の 1024x768 ピクセル、256 色の GIF イメージを ImageMagick[166] を用いて 512x384 ピクセル、32 色に変換すると 64Kbytes となり、ファイル転送時間が約 1/8 に短縮される。今日のインターネットでは、こうした大規模データのナビゲーションをサポートできるようなプロトコルが求められている。

また、インターネットの普及によって遠隔地の研究機関との共同研究はますます盛んになってきており、プロジェクトの共同作業に用いるデータやホームディレクトリなどを WWFS を用いて共有する試みがなされている。しかし現在の FTP プロトコルはパスワードを暗号化せずに送るなどのセキュリティ上の問題点があるため、機密を含むデータの共有などには適していない。したがって、公開鍵暗号などのセキュリティ技術を応用したファイル転送プロトコルが求められている。

2.4.2　WWFS 92 に対する改良: WWFS 94

前節で述べたような、新たな応用範囲・利用形態の広がりによって、WWFS 92 はその設計当初に予定していたような、インターネット上で公開された「疎な」ファイルの単一クラスタ内部での共有という目的を逸脱し、公開できないファイルへのアクセス、「密な」ファイルの共有、複数クラスタにまたがる、より大規模かつ広範囲な運用などといった目的にも応用されはじめている。こうした現状をふまえて、本研究では WWFS 92 モデルに対して以下のような改良をおこなった。以下、このモデルを WWFS 94 と呼ぶ。WWFS 94 における改良は大きく分けて、サーバ系の構成を支援する機能と、WWFS において扱える情報の範囲を広げるための機能の二つからなる。

サーバ系の構成を支援し、利用形態を広げるための拡張は以下の通りである。

- サーバ間におけるキャッシュ共有 (csd interconnect)
  単一の csd のみからなる独立動作だけでなく、複数の csd が高速ネットワークの周辺に配置されているような状況における csd 間のキャッシュ共有をサポートし、ボトルネックへの負荷集中をさらに軽減することを目標とする。

- 複数 csd の中継による階層化キャッシュ (csd relay)
  クライアント ― csd ― ファイルサーバという単一階層のファイルアクセスだけでなく、複数の csd を介在するような多階層にまたがるファイルアクセスもサポートし、広域ネットワークにおいて複数のクラスタが存在するような状況におけるボトルネックへの負荷集中をさらに軽減することを目標とする。

- ファイルシステムの一貫性制御
  更新があまり行なわれないファイルだけでなく、更新が頻繁に行なわれるような、ニュース、ログファイル、共同作業ファイルなども対象とし、そこで必要とされるファイルシステム/キャッシュの一貫性制御を行なうことを目標とする。

扱うことのできる情報を拡張し、WWFS の応用範囲を広げるための拡張は以下の通りである。

- ファイル転送におけるセキュリティ
  インターネット上の公開ファイルだけでなく、ホームディレクトリ、共同作業ファイルなども対象とし、広域ネットワークにおける盗聴によるデータの漏洩を未然に防止できるようなセキュリティ技術をファイル転送に応用することを目標とする。

- ファイル属性の一般化
  ファイル名、ファイル長、生成時刻などのあらかじめ定められたファイル属性だけでなく、subject, resolution, content-type, path といった、データの内容やキャッシュ管理に用いることができる情報などもファイル属性として統一的に扱うことを目標とする。

	static	dynamic
tagged	directory, tar.Z, dbm ASN.1, music CD Laserdisc	WAIS/Archie result CG rendering server digital radio/TV
non-tagged	text, bitmap raw audio/video file	block/character device socket, named pipe

表 2.10: WWFS 94 ではファイルタイプを一般化し、動的な/静的なファイル、タグ付き/タグなしファイルという二つの直交した分類を行なう。ここでは、こうして得られた四通りのファイルタイプにおける様々なデータの区分を例によって示した。

- 動的なファイル生成
  内容が固定された静的なファイルだけでなく、与えたパラメータによって内容が変化するようなファイルも対象とし、動的に生成されるデータを統一的に扱うことを目標とする。

- ファイルタイプの一般化
  ファイル全体のみを単位として扱うのではなく、ファイルの一部をタグ付けされたバイト列として扱い、タグを単位とするより粒度の細かいファイルアクセスを統一的に提供することを目標とする。

上で述べたような、動的なファイル、タグ付けされたファイルなどのファイルタイプの一般化により、四通りのファイルタイプが定義される。この分類では、表 2.10 に示すように様々なデータを扱うことができる。動的なファイル生成によって、情報検索とのインタフェース、デバイスやソケットなどとのインタフェースや、前節で述べたような大規模データのナビゲーションを WWFS 94 で扱うことができる。また、タグ付けされたファイルによって、tar-compress されたファイルの一部や、dbm などの構造化されたデータなどを WWFS 94 で効率的に扱うことができる。

2.4.3 プロトコルに対する要求

前節で述べたような多様な利用形態のための拡張モデルを実現するためには、その基礎となるプロトコルに対して様々な付加機能や柔軟性が要求される。まず、csd interconnect によるキャッシュ共有を行なうためには、サーバ間プロトコルにおいて各々のファイルに関するキャッシュ状態を交換できることが必要である。また、csd relay による階層化キャッシュを実現するために、サーバ間プロトコルにおいてエラー発生時の状態遷移を明確に定めることが重要である。さらに、異なるマシン上にある複数のファイルシステムおよびキャッシュの一貫性を保つために、一貫性制御を支援するようなファイル操作 (ロック、同時書き込みの解消) を行なうための枠組が重要である。また、ネットワーク障害か

	NFS	FTP	NNTP	HTTP	Gopher
サーバ間の協調動作	×	×	○	×	×
一貫性制御	×	×	○	×	×
セキュリティ拡張	○	○	△	○	△
ファイル属性の一般化	×	△	△	△	×
ファイルの動的な生成	×	△	×	○	○
タグ付きファイル	×	×	×	△	×

表 2.11: WWFS 94 の評価基準にもとづいて、インターネットで広く用いられている情報転送プロトコルの特性を比較した。ここで○印はプロトコルが本来サポートしているか、既にプロトコルの拡張が行なわれていることを表し、×印はプロトコルが本来サポートしておらず、プロトコルに対する拡張も行なわれていないことを表す。△印はそのような機能の非互換な実装が存在するか、あるいは容易に実装が可能であることを表す。

ら復帰したファイルシステムに対して、非常に大きなプロトコル/CPU オーバーヘッドなしに最近の変更を反映するために、NNTP[167] における NEWNEWS コマンドのような、指定された時間に遡って、その時点からあとに到着したファイルの一覧を得るような操作がプロトコルにおいて定義されていることが望ましい。

また、応用範囲を広げるための拡張モデルを実現するためにも、その基礎となるプロトコルに対して様々な柔軟性が要求され、それと同時に、多様な情報を統一的に扱うことが要求される。まず、ファイル転送におけるセキュリティを保証するため、終端ノード間や終端ノード・中間ノード間における認証を行なうための枠組を用意できることが望ましい。次に、前節で述べたような一般化されたファイル属性を統一的に扱えることが必要である。さらに、動的に生成されたファイルやタグが付けられたファイルなどを扱えることもプロトコルに対する基本的要件である。

2.4.4　既存のプロトコルの特性比較

WWFS 94 では、前節で論じたような特性をみたすプロトコルが求められているが、既存のプロトコルをそのまま用いた場合こうした要求をみたすことができず、プロトコルの拡張や新たなセマンティクスの定義といった何らかの改善が必要となる。その結果として、インターネットで広く用いられている実装との間における相互運用性が問題となる。

ここでは、インターネットで広く用いられている NFS, FTP, NNTP, HTTP, Gopher の五つのプロトコルに焦点をあて、その各々について、WWFS 94 の評価基準にもとづいてプロトコルの特性を比較し、表 2.11 にまとめた。

この表からもわかるように、既存のプロトコルは WWFS 94 が要求するプロトコル特性を満たしておらず、新たなプロトコルの設計・開発が必要であると考えられる。

2.5 まとめ

本研究では、まず 91 年度に DFS WG において提案した WWFS 92 モデルの有効性を確認するため、このモデルに基づいた中間サーバ csd を奈良先端科学技術大学院大学、オムロン株式会社、WIDE プロジェクトにおいて運用を行なった。そして、それぞれについて通常の FTP アクセスとの利用頻度の比較、mirror と WWFS アクセスを比較した場合のトラヒック改善率などを明らかにした。次に、インターネットにおけるより大規模な運用をめざした普及活動について述べた。

さらにこうした運用・評価を通して明らかになった問題点、およびそこでとった対処方法をプロトコルアーキテクチャ、ネットワークアーキテクチャの二つの側面から論じた。また、こうした運用の結果、Mule, Mosaic, Chimera, freeWAIS などの先進的なアプリケーションによる新たな応用などの成果が得られ、その結果として WWFS ユーザが拡大し、より大規模な運用や、さらに広い応用範囲を考慮した新たなモデルが必要となった。

本研究ではこうした問題点および要求を詳細に分析し、WWFS 92 の改良モデル WWFS 94 を提案し、そこで求められるプロトコル特性を明らかにした。さらに、WWFS 94 が要求するプロトコル特性を満たし、高い記述能力と新しい技術への柔軟性をもつプロトコル ITP を開発した。ITP については次章で詳細に述べる。

第 3 章

情報転送プロトコル ITP

前章で論じたように、既存の主な情報転送プロトコルは WWFS 94 に代表されるような次世代の情報配送システムが要求するプロトコル特性を満たしておらず、新たなプロトコルの設計・開発が望まれる。また、FTP や NNTP などのプロトコルがインターネットにおいて広く採用されてから約十年が経とうとしており、この間にみられた関連技術の進歩をとりいれた新たなプロトコルの設計・開発が待望されている。本研究ではこうした背景のもとに、ITP (Information Transfer Protocol) を設計し、簡単な ITP サーバおよびクライアントを実装した。

本研究では ITP の設計にあたり、近年の情報理論、計算機アーキテクチャ関連技術およびインターネット関連技術の進歩に焦点をあて、このような技術の進歩を積極的に採り入れることができるような枠組みを提供することを目標とした。とりわけ伝送路の高速化にともなうプロトコルの高速化、トランスポート/データリンク層技術の進歩、サーバ実装技術の進歩、セキュリティ技術の進歩、プロトコルの多言語化などは次世代インターネットの基盤技術として重要な役割を果たすと考えられ、そのような進歩を柔軟に採り入れることができるようなプロトコル、およびプロトコル拡張のためのメカニズムが特に必要とされている。

本章では、このような目標をもつ ITP について、まずそれを構成する要素技術について概観する。次に、そのような様々な要素技術を将来にわたって統一的に扱うための枠組であるコマンド、属性、環境といった概念を導入し、この枠組における多様な情報システムの統一的な扱いについて論じる。最後に、ITP を適用することができる応用分野、および既存のプロトコルから ITP へのプロトコル変換について考察する。

3.1 要素技術

ITP を構成する要素技術は、情報転送プロトコルの高速化、効率化のための技術と、プロトコル拡張のための枠組を提供する技術の二つに大別される。この節ではそれらについて概観し、ITP がそれらをどのように実現しているかについて簡単に述べる。

3.1.1 情報アクセスの高速化、効率化のための技術

広域ネットワークやギガビットネットワークでは、TCP の three-way handshake などのコネクション確立時の RTT が情報転送プロトコルの主なオーバーヘッドである。こうした RTT オーバーヘッドを最小化するための技術として、csd や Berkeley sendmail release 8 などにみられるような connection caching の実装技術や、nntplink[168] において採用されているような定常的なコネクションを維持する実装技術などがある。ITP では単一コネクションにおいて複数のコマンド列をサポートすることでコネクション確立時の RTT オーバーヘッドの影響を最小化することができる。また、このようにして定常的なコネクションを張った場合、ネットワーク障害によってコネクションが切断されたときの障害回復が問題となるが、ITP ではプロトコル状態マシンのすべての状態においてコネクション閉鎖時の状態遷移を定めることで、こうした実装技術の採用が容易になると考えられる。

また、そのほかの RTT オーバーヘッドを減らす手法として、Prospero Filesystem の ARDP[169] (Asynchronous Reliable Delivery Protocol) で用いられている lightweight connection、IETF で提案されている TMux[170] (tranport multiplexing) といったものがある。ITP ではネゴシエーションに基づいてデータ転送プロトコルを変更できるため、こうした技術の採用も容易である。

このほか、コマンドの粒度を大きくとることで、複数のファイル属性やファイルを一度に転送し、command – response サイクルにおける RTT を最小化することができる。ITP では任意のファイル/ファイル属性の一括転送をサポートしているため、そのようなプロトコルオーバーヘッドを最小化する手法を用いることもできる。

また、動画や音声ファイルのプレビュー操作や、文書ファイルの全文検索をサーバ側でサポートすることで、全データを peer entity 間で転送することなくデータの目的とする部分へのアクセスを行なうことができ、データ転送の効率化をはかることができる。ITP ではタグを単位とした粒度の細かい情報アクセスや、サーバ側での動的なファイル生成をサポートしており、そのような情報の検索やプレビュー操作を行なう場合に効率的な情報アクセスを行なうことができる。

また、UNIX ファイルシステムで用いられているような一意なパス名、WAIS の Object ID[171]、ネットワークニュースの Message ID[167] といった情報の一意な名前づけを行なう手法によって、情報を効率的にキャッシュすることが可能である。さらに、MD5[138] などのディジタル署名アルゴリズムをファイルに応用することで、ファイル内容に基づいたファイルの同一性の判定をファイル全体にアクセスすることなく行なうことができる。ITP ではこうした様々なファイルの付加情報を属性というかたちで統一的に扱うことができる。

3.1.2 プロトコル拡張のための枠組を提供する技術

従来の情報転送プロトコルは、周辺技術の進歩をとりいれると同時に古いバージョンとの相互運用性を保つことが困難であり、その結果として、データリンク技術の進歩や

セキュリティ技術の採用が立ち後れてしまっている。将来にわたって技術の進歩を採り入れていくためには、コネクションの確立時に TELNET プロトコル [172] にみられるようなネゴシエーションを行ない、異なるプロトコル実装・ファイルシステム・OS・アーキテクチャの間での相互運用性を保証する必要がある。

また、サーバ系の構築を支援し、サーバ間におけるキャッシュ共有やファイルシステムの一貫性制御を行なうために、従来のファイル転送プロトコルのように予め決められた

ファイル名、ファイル長などのファイル属性やカレントディレクトリなどのコンテキストだけを扱うのではなく、そのような系を構築するさいに必要とされる属性を自由に定義でき、扱えることが重要である。ITP では、ファイルの付加情報を属性、セッションの付加情報を環境として統一的に扱い、ネゴシエーションの範囲内において属性・環境を自由に定義できる。

また、新たな認証技術やトランスポート技術、先進的なサーバ実装技術やファイルシステム技術を採用するためには、プロトコル状態マシンが拡張可能であることが重要である。ITP では、ネゴシエーションの範囲内においてプロトコル状態マシンを拡張でき、そのような新しい技術の積極的な導入を支援することができる。

3.2　プロトコルの詳細

本節では、さまざまな要素技術を将来にわたって統一的に扱うための四つの基礎的概念 (コマンド、リプライ、環境およびファイル属性) について述べる。

ITP におけるコマンドは、ファイルの操作、環境の操作、ファイル属性の操作の三つに分類される。

ファイルの操作は以下の四つのコマンドによって構成される。ディレクトリは特殊な形式のファイルとして扱われる。

PUT	files で指定されたファイルに書き込む。
GET	files で指定されたファイルを読み出す。
CREATE	files で指定されたファイルを作る。
DELETE	files で指定されたファイルを消す。

単一コネクション上の環境は、以下の二つのコマンドによって操作される。環境はトランスポート層および ITP 自身のふるまいを決定する。

ESET	指定された環境を設定する。
EGET	指定された環境を読み出す。

ファイル属性の操作は以下の四つのコマンドからなる。そのコマンド実行時にだけ有効な、一時的な環境指定を含んでもよい。

ASET	指定されたファイル属性を設定する。
AGET	指定されたファイル属性を読み出す。

ASETR	query に一致するファイルすべてのファイル属性を設定する。
AGETR	query に一致するファイルすべてのファイル属性を読み出す。

利用可能な環境および属性は以下の四つのあらかじめ名前の定められた環境を用いて取り決められる。

do	指定された属性の使用を要請する。
dont	指定された属性の使用中止を要請する。
will	指定された属性の使用を許可する。
wont	指定された属性の使用を拒否する。

それぞれのコマンドに対して、一つまたは複数のリプライが用意される。リプライは簡単のため三桁の数字によって指定される。一桁目はコマンドが受理されたかどうかを示す。

2	command ok.
3	command ok so far, send the rest of it.
4	command was correct, but couldn't be performed.
5	command unimplemented, incorrect, or serious error.

残りの二桁はメッセージを解釈するエンティティを指定する。

1x	human readable message
0	user
1	statistics
2	network management
3	debug
2x	machine readable message
0	end system
1	intermediate system

任意のコマンドおよび機械可読なリプライに対して、以下のような環境を指定することができる。

files	対象とするファイル
options	機能のリスト
headers	ファイル属性のリスト
format	AGET の出力形式
query	AGETR, ASETR の検索条件
encoding	GET, PUT の出力形式
resolution	イメージの解像度
cwd	現在対象としているディレクトリ
itp-version	ITP のバージョン

transport	使用するトランスポート・プロトコル
flowspec	データストリームの流れ制御

任意のファイル、機械可読なリプライに対して、以下のような属性を指定することができる。

type	ファイルの種別
mode	ファイルのアクセス制御情報
mtime	ファイルの最終変更時刻
size	ファイル長
name	ファイル名
oid	オブジェクト ID
subject	ファイル内容
content-type	ファイル形式
md5	MD5 アルゴリズムにより計算されたシグネチャ
cached	このファイルをキャッシュに持っているホスト名

ITP の プロトコル状態マシンは拡張可能である。ネゴシエーションを行なった peer entities の間では同じ状態機械を共有しているものとする。ITP では各々の機能拡張に対して、そこで必要なプロトコル状態マシンの拡張を明確に定める。

3.3　記述能力

前節で述べたような一般化されたプロトコル体系によって、従来、ファイルシステム、ニュースシステム、データベースシステムなどの形で提供されていた様々な情報システムを統一的に扱うことが可能になった。

この節では、いくつかのファイル属性の一般化について述べたあと、そのような様々な情報システムが提供するセマンティクスの ITP における扱いについて、例を中心として述べる。

3.3.1　ネゴシエーションに基づくプロトコル拡張

前節で述べたようなネゴシエーションを基本としたプロトコル拡張のための枠組によって、ITP ではファイル属性および環境を容易に拡張することができる。ITP では通常、TCP コネクションを確立し、プロトコル状態マシンの初期状態を responder から得た直後に、initiater が do 環境を設定することでネゴシエーションを開始する。

```
ESET do="auth, ipsec, kerberos, md5, mime, base64, iso2022int, unicode"
220-will="auth, md5, mime, base64, iso2022int"
220 wont="kerberos, ipsec, unicode"
```

この例では Kerberos, IP security, Unicode のプロトコル拡張オプションが拒否され、その他の MD5, ISO2022-INT などのオプションが受理された。このようにして利用可能な

プロトコル拡張を決定した後、クライアントのプロトコル状態マシンはその他のコマンドを発行できる状態に遷移する。

以上で述べたようなネゴシエーションを行なった場合、TELNET (rfc 854) で指摘されているように、acknowledgement loop が起きるという問題点がある。これに対して、TELNET では以下のような規則が一般的に適用されており、ITP においてもこれを採用する。

- 現在使用しているオプションを相手に通知する目的のみにおいてネゴシエーションを開始してはならず、オプションの状態を変更する要求しか発行しない、
- すでに採用したオプションについて相手が要求した場合、それに対する ack を送ってはならない、
- オプションは、command byte stream 中において、その存在する位置から後でのみ有効にしなければならない。

3.3.2 様々なセマンティクスの統一的な扱い

ITP では UNIX にない様々なプロトコルのセマンティクス、例えば ISAM や NNTP の expiration date、ISAM の secondary key、record/file locking などをファイル属性という形で扱える。また、サーバ系を構成するために必要な情報もファイル属性として定義することができ、以下のようなインタラクションによってキャッシュの共有が可能となる。

```
AGET files="/foo/bar.c" headers="cached"
220 name="/foo/bar.c" cached="fsc1"

ASET files="/foo/bar2.c" cached="fsa2"
220 state=2
```

上の例ではファイル /foo/bar.c をキャッシュしているホスト名を/foo/bar.c のファイル属性 cached によって得ている。また、下の例では /foo/bar2.c をキャッシュしているホスト情報を交換している。ここでは ack として responder 側のプロトコル状態マシンを返している。

3.3.3 タグを用いた細粒度アクセス

ITP では、offset によって指定されたファイルの一部や、tar-compress されたアーカイブの一部、dbm[173] ファイルの一部などを、タグを指定したファイルアクセスという形で効率的にアクセスできる。これにより、tar-compress ファイルの一部を tag=”tar.Z <filename>”、任意のファイルの 1024 バイトブロックを tag=”block-1024 <block-number>” として指定でき、以下のようなアクセスが可能となっている。

```
AGET files="foo.tar.Z" tag-scheme="tar.Z"
220-name="foo.tar.Z" tag="tar.Z README" size=857 mtime=...
220-name="foo.tar.Z" tag="tar.Z doc/manual.ps" size=186314 mtime=...
220 name="foo.tar.Z" tag="tar.Z doc/manual.tex" size=45260 mtime=...
```

また、ファイルの一部についてのみ MD5 の値を計算することができるような実装では、1024 バイトブロックごとに計算した MD5 の値によってファイルの一部について同一性を確かめることができる。

```
AGET files="foo.tar.Z" tag-scheme="block-1024" headers="MD5, name"
220-name="foo.tar.Z" tag="block-1024 0" md5="kad9khaiuq7yivx"
220-name="foo.tar.Z" tag="block-1024 1" md5="poiuewr6zcnvwiu"
220 name="foo.tar.Z" tag="block-1024 2" md5="qz8yq32naxadsfh"
```

また、前述の cached 属性をタグ付けされた部分ごとに設定できる場合、以下のような粒度の細かいキャッシュ共有も可能であると考えられる。

```
AGET files="foo.tar.Z" tag-scheme="tar.Z" headers="cached, size"
220-name="foo.tar.Z" tag="tar.Z README" cached="fsa1" size=857
220 name="foo.tar.Z"
    tag="tar.Z doc/manual.ps" cached="fsb2" size=186314
```

3.3.4　Object ID の一般化

ITP では、これまで特定のファイル、メッセージなどを情報システム内部において一意に識別するために用いられてきた内部表現を、情報の一意性を保証するための外部表現としても用いる。これにより、NNTP のメッセージをファイル属性 oid="nntp <message-id>"、WAIS のオブジェクト識別子をファイル属性 oid="z39.50 <oid>" として指定できる。このような識別子によって、UNIX ファイルシステムに見られるような木構造をなす名前空間だけでなく、DAG やメッシュなどのより柔軟なトポロジーに基づく名前空間についても効率的なキャッシュ手法を適用することができる。

3.3.5　多言語環境におけるパス名

英語圏以外においてインターネットが発展するためには、アジア、アフリカ、ヨーロッパなどの様々な言語を用いた情報アクセスが円滑に行なえることが重要である。したがって、ITP では多言語環境におけるパス名の交換形式を必要としているが、既存のファイルシステムでは単一のコード体系におけるパス名の多言語化しか行なっておらず、こうした枠組を新たに設計する必要があると考えられる。

ITP では rfc 1521[174] の 5.2 節にならい、任意のコード系を base64 alphabet にエンコードする。しかし、rfc 1521 で提案されている base64 alphabet はファイルシステムにおけるメタ文字 / を用いているため、/ を用いない base64 alphabet を新たに提案する必要がある。

こうした多言語環境への対応として、Plan9 では FSS-UTF[175] (File System Safe UCS Transformation Form) と呼ばれる Unicode を基本とした Unicode のエンコーディングを用いているが、Unicode は漢字を基本としたアジア圏の言語を同時に扱うことができないという問題があり、アジア圏における FSS-UTF の利用は望ましくないと考えられる。

こうした背景から、ITP では rfc 1522[176] にならい、(charset, encoding, encoded-text) の三つ組を基本として多言語パス名の表現を目標とする。さらにその中で、rfc 1554[177]

で定義された Compound Text を基本とした、アジア圏の言語を同時に扱うことができるコード体系による多言語ファイル名のエンコーディングを広く用いることを目標とする。

3.3.6 アクセス制御情報の統一的な扱い

ITP ではアクセス制御情報をグループごとに定義された (capability, value) の対として捉え、その最も冗長な g エンコーディングと UNIX との親和性の高い unix エンコーディングを定義した。これによって、以下のように異なる OS 間においてアクセス制御情報を交換することが可能であると考えられる。

```
ASET files="foo.txt"
     mode="g owner read=yes write=yes, g group-1 read=yes"
```

また、UNIX などの広く用いられている OS 間では、FTP と同様に簡潔な表現が可能である。

```
ASET files="foo.txt" mode="unix 0755"
```

さらに、アクセス制御情報の g エンコーディングによって、利用可能なトランスポートプロトコルや、書き込み操作の種類といった高次のセマンティクスをファイルごとに記述することができる。例えば、二種類の QoS (Quality of Service) によって利用可能なトランスポートプロトコルが異なる場合、

```
ASET files="speech.au" mode="g QoS-1 TCP=yes ATM=no, g QoS-2 ATM=yes"
```

また、管理目的で用いるログなどに対して、ログを行なうプロセスをグループ logger、ログを参照する管理者をグループ admin に分類し、logger からは追加のみ、管理者は読み出しのみに制限するといったセマンティクスを記述できる。

```
ASET files="admin.log"
     mode="g logger append=yes read=no, g admin append=no read=yes"
```

3.4 応用分野

前節で述べたように ITP は高い記述能力と新しい技術への柔軟性をもち、様々な目的に応用することが可能である。この節ではこうした応用分野における ITP の位置づけと、既存のプロトコルからのプロトコル変換について考察する。

3.4.1 ファイル転送への応用

ここでは現在インターネットで最も広く使われているファイル転送プロトコル FTP からのプロトコル変換について、例を中心として述べる。

ITP では responder が state 0 であるとき、環境 user, pass を設定することで anonymous ftp login に相当する操作を行ない、コマンドを受理可能な state 2 に遷移する。

```
220-state=0
210 welcome to itp server.  enter user and passwd.

ESET user="anonymous" pass="youki@wide.ad.jp"
220-state=2
210 anonymous access ok.
```

responder 側が公開ファイルのみを扱い、クライアントの認証を必要としない場合は state 2 を返し、login フェーズを省略することができる。

```
220-state=2
210 welcome to itp server.  access ok.
```

カレントディレクトリの変更は、環境 cwd の設定によって行なう。

```
ESET cwd="/VOA/radio/newswire/sat"
```

同様にして、環境 cwd を得ることでカレントディレクトリを知る。

```
EGET cwd
220 cwd="/VOA/radio/newswire/sat"
```

ディレクトリを明示的に指定してファイル一覧を得たい場合、環境 cwd を一時的に設定し、環境 files を指定せずに AGET コマンドを発行する。

```
AGET cwd="/VOA/radio/newswire/fri"
```

ファイル一覧における出力形式は、環境 headers によって変更できる。

```
AGET headers="mode, link, owner, group, mtime, name"
```

ファイルを得る場合は、GET コマンドの initiater がデータコネクションを開始する。responder は GET コマンドを受けとると、接続アドレス形式、アドレス、ポートを返すとともに状態遷移を通知して接続を待つ。

```
GET files="foo"
220-port="tcp ip4 133.1.4.10 2097" state=3
210-accepting data connection.
220-state=2
210 file transfer complete.
```

ファイルの一部に追加/変更を行なう場合は、PUT コマンドとともに tag を指定する。

```
PUT files="foo" tag="offset 29384"
220-port="tcp ip4 133.1.4.10 2098" state=4
210-accepting data connection.
220-state=2
210 file transfer complete.
```

responder がファイル書き込みに対して前処理を必要とする場合、コネクション確立時のネゴシエーションに基づいて拡張されたプロトコル状態 (この例では 10) を通知し、ファイルの allocation などの処理を行なう。

```
PUT files="foo"
220-state=10
210 expecting bytes and records.

ESET bytes=93284 records=91
220-port="tcp ipv4 133.1.4.10 2099" state=4
210-allocation successful; accepting data connection.
220-state=2
210 file transfer complete.
```

ファイル名の変更は、files で指定したファイルに対して属性 name を変更することで行なう。

```
ASET files="foo.txt" name="foo.txt.orig"
```

ディレクトリの作成はタイプ d のファイルの作成によって表現する。

```
CREATE files="mydir" mode="unix 0755" type=d
```

responder との接続状態は、環境 alive を読み出すことで確認することができる。

```
EGET alive
220 alive=yes
```

3.4.2 ネットワークニュースへの応用

ITP は Message ID, Subject, From といったヘッダ情報をファイル属性として扱えるため、ITP をネットワークニュースの転送へ応用することも可能である。また、csd において ITP を実装することで、ニュース記事を要求時に配送するようなニュース配送系を構成することが可能となり、FTP でみられたようなトラヒックの劇的な削減が期待される。

3.4.3 CWIS への応用

ITP は、その拡張/縮小可能なプロトコル構成により、PC, Macintosh などの低価格プラットフォームにおける CWIS[178] (Campus Wide Information Systems) のための軽量プロトコルとしての応用も期待されている。また、今後期待されるワークステーション技術およびネットワーク技術の発達により ITP クライアントの処理能力が飛躍的に拡大した場合、プロトコル状態マシンの拡張によって新たなトランスポート/データリンク、クライアントの高速化、セキュリティ、多言語環境といった技術を段階的に導入することが可能であると考えられる。

3.5 ITP の実装

94 年 2 月現在、ITP の実装は perl で記述したプロトタイプのみにとどまっており、タグによる細粒度の情報アクセス、サーバ間の協調動作などを支援する機能が実装されていない。今後、ITP に関して以下のような設計・開発をすすめていく必要があると考える。

- タグをサポートした ITP サーバ、クライアントの構築。
- csd における ITP の実装。
- ITP を用いた協調サーバ系の構成。
- Gopher や FTP などの既存のプロトコルとのゲートウェイの作成。

3.6　まとめと課題

前章で述べた WWFS 94 が要求するプロトコル特性を満たし、高い記述能力と新しい技術への柔軟性をもつプロトコル ITP を開発し、それを構成する要素技術、基礎概念を明らかにしたのち、様々な情報システムの統一的な扱い、および既存のプロトコルからのプロトコル変換について述べた。

しかしながら、今後 ITP に関して多くの課題が残されており、さらなる研究が必要であると考えられる。

1. プロトコル変換技術に関する研究
   従来のプロトコルから ITP への変換手法について研究を行ない、プロトコル変換手法およびそこで必要とされる基礎技術を明らかにする。
2. ITP の応用に関する研究
   ITP を用いることで可能となる新たな応用分野について研究を行ない、そのような応用分野における実験運用を通して ITP の可能性を実証し、ITP の改良を行なう。
3. 協調サーバ系に関する研究
   ITP を用いた協調サーバ系をモデル化し、情報アクセスにおけるサーバ系の有用性を評価する。
4. IETF における標準化活動
   IETF (Internet Engineering Task Force) に対して本研究の成果をフィードバックし、ITP で提案したような高いプロトコル記述能力をもつプロトコルの標準化を目指す。

第 4 章

移動型計算機のためのファイルシステム

従来の分散ファイルシステムは、固定型の計算機とは異なる特性を持つ移動型計算機で利用するには、応答性や可用性の面で問題が多い。

そこで、移動型計算機に適しており、かつ導入が容易な分散ファイルシステムとして Personal File System (PFS) を設計、実装し、その評価をおこなった。このファイルシステムでは、ワーキングセットの明示的な指定によるキャッシュの有効利用と、NFS プロトコルを用いたカーネルとのインターフェースによる導入の容易さを実現した。

4.1 背景

計算機は小型化され、持ち運びが容易な大きさになり、Virtual Internet Protocol (VIP)[179] を用いる事により、計算機の移動を意識する事無く、通信を続ける事が出来るようになった。

当然、移動型計算機においてもファイルサーバ上の、コマンドファイルやユーザのホームディレクトリを共有して、他の計算機と同様の作業を行いたいという要求がでてきた。

しかし、移動型計算機の接続するネットワークは低速な事も多く、移動中の通信断などにより一時的に不通になる事もある。そのため、安定したネットワークを想定している従来の NFS では、応答性や転送速度に問題が生じる。

4.2 分散ファイルシステムの現状

この節では現在の主な分散ファイルシステムの概要と、移動型計算機上で利用する際の問題点を述べる。

4.2.1 NFS

Network File System (NFS) [124] は、ネットワークを介してのファイルの共有を実現する機構であり、異なる種類の計算機でのファイル共有も可能とする事を目的としている。

NFS はプロトコルが機種などから独立し、状態をできるだけ持たないように設計されている上、その仕様が rfc として公開されている為、非常に広く普及している。このため、システム管理者が NFS を導入するのが非常に容易になっている。

しかし、NFS を移動型計算機で使用する場合、次のような問題点がある。

- 呼び出しのオーバーヘッドが大きい
  NFS のプロトコルには状態が無いため、RPC の要求一つ毎に認証情報などがつき、オーバーヘッドが大きい。

  また、一度の RPC 呼出しで転送できるデータ長が 8192 バイトに制限されているので、大きなファイルを転送するには複数の RPC に分割しなければならない。

- ネットワークの遅延による応答性の悪化
  RPC の要求一つ毎に応答を待つので、遅延時間の長いネットワークを通した場合、応答時間が極端に長くなる。

- キャッシュの内容が正しいことを保証できない
  クライアント側のキャッシュの整合性を保証する機構が無いため、キャッシュの内容を長時間保証できず、ファイル操作の度にファイルサーバに内容の確認をしなければならない。

- 不正使用を防げない
  一般的に使われている NFS の実装では、アクセス制御にクライアントのホスト名と、クライアント自身から送られたユーザー情報しか使わない。そのため、他のホストやユーザーになりすまして不正にファイル操作を行なう事ができる。

  また、転送するデータを暗号化しないので、ネットワークの経路上で盗聴される可能性がある。

これらの問題のため NFS は、回線速度や遅延時間などが異なるさまざまなネットワークに接続し、移動による通信断などが起こる、移動型計算機での利用には適さない。

4.2.2　AFS

Andrew File System (AFS) [180] は、カーネギメロン大学で開発された分散ファイルシステムであり、広域ネットワークを通してのファイル共有を考慮して設計されている。
データの暗号化や、クライアントがキャッシュに読み込んでいるファイルが、他のクライアントによって変更された時に、サーバがクライアントに変更を通知するコールバックを行なう事などにより、広域ネットワークでの利用が可能になっている。
しかし、AFS を移動型計算機で利用する場合、次のような問題点がある。

- 通信が断絶すると利用不能になる
  AFS は、クライアントに残っているキャッシュの有効性をサーバからの callback を受けることにより保証しているので、通信が断絶し callback を受け取ることができなくなると、キャッシュの内容を保証できずにファイル操作ができなくなってしまう。

- 動作環境をつくりにくい
  AFS の一部は OS のカーネル内部にあるので、AFS を使うには、カーネルの変更が必要である。このため、OS の一部として供給されるか、カーネルのソースコードに手を入れるかしないと使う事ができない。

  しかし、AFS は NFS ほど一般的でないので、AFS を利用可能な計算機は少なく、AFS の動作環境はつくりにくい。

- *AFS* への移行コストがかかる
  AFS サーバには、現在一般的に使われいている NFS サーバとの互換性はない。したがって、既に NFS で共有されているデータを AFS で共有する場合は、データを全て AFS サーバに移動しなければならない。

  また、従来の NFS クライアントマシンがそのデータを利用するには、 AFS のクライアントを実装しなければならない。

4.2.3 Coda File System

Coda File System [181] は、ネットワーク接続を絶たれても動作するように設計された、移動型計算機のためのネットワークファイルシステムである。Coda は AFS を改良して設計されている。

Coda では、コールバックを行なうなどの AFS の特徴に加えて、移動型計算機で使うために、いくつかの機能が追加されている。

- サーバの複製を作る
- ネットワーク切断時にはキャッシュのみで動作し、再接続後にサーバに反映させる
- あらかじめユーザの指定したファイルをキャッシュに取り込んでおく事ができる。

などである。

ネットワークの切断に対処できるようになってはいるが、Coda にも AFS と同様の問題が残っている。すなわち、

- 動作環境を作りにくい
- Coda ファイルシステムへの移行コストがかかる

である。

4.2.4 従来の分散ファイルシステムのまとめ

従来の分散ファイルシステムを移動型計算機で利用する際の、利点と問題点についてまとめると、次のようになる。

	導入の容易さ	広域での使用	通信断での使用	書き込みの保証
NFS	容易	速度に難あり	不可能	完全
AFS	容易ではない	良好	不可能	完全
Coda	容易ではない	良好	可能	通信断中は保証無し

これらの分散ファイルシステムでは、それぞれの目的を達成するために、ここに示したような問題を容認している。言い換えれば、致命的ではない欠陥を容認することにより、分散ファイルシステムとしての長所を伸ばしている。

4.3　PFS の概要

4.3.1　移動型計算機の特徴

移動型計算機には、固定型の計算機と比較して次のような特徴がある。

固定型の計算機に対して資源が乏しい

小型化のために、移動型計算機は固定型計算機よりも速度が遅く、少ないメモリしか持たず、ディスク容量も小さい。

通信路が不安定

移動型計算機は、さまざまな回線速度/遅延時間のネットワークに接続する。例えば、大学内の Local Area Network (LAN) に接続された計算機は、大学内のの計算機と数ミリ秒の遅延で通信を行なう事ができるが、アメリカに移動した計算機と大学内の計算機との通信は、広域ネットワークを介するため遅延時間が通常数秒に達する。また、無線などによる接続では一時的に通信が途切れる事もある。

信頼のおけないネットワークも通る

移動型計算機が利用するネットワークでは、回線の盗聴やなりすましが起きる可能性もある。

通信が絶たれた状態でも動作する

移動型計算機は、移動中などで通信路が確保できない事があり、その間も動作する。

異なるサブネットワークと接続する

移動型計算機は移動先のサブネットワークと接続する事があり、その場合ネットワーク上の位置も変化する。

個人で使うことが多い

移動型計算機は、その大きさや機能から、1 台を 1 ユーザが占有することが多い。

4.3.2 PFS に必要な機能

移動型計算機のための分散ファイルシステムにとって、重要で実現しなければならないことと、容認してもよいことを考える。

実現しなければならないこと

- 既存の分散ファイルシステムの内容を参照できる
  共有すべきファイルの多くは、既に既存の分散ファイルシステム上で共有されている。したがって、これらのファイルを既存のファイルシステムと共有する必要がある。
- 遅延の大きなネットワークを使用している時でも良好な応答性が確保できる
  移動型計算機とファイルサーバとの間のネットワークの遅延が大きいことが考えられるので、ネットワークの遅延が大きくても応答性を悪化させない機構が必要である。
- 通信が突然途絶えても作業を続行できる
  移動型計算機は移動などでネットワークとの接続を突然絶たれることがある。また、接続を絶たれた状態で動作することもあるので、ファイルサーバとの通信が絶たれても必要な作業を行なうことのできる機構が必要である。
- ユーザーにとって、本来のファイルシステムと同じ操作ができる
  従来のアプリケーションをそのまま使うには、同じ操作ができることが必要である。

実現することが望ましいこと

- クライアント側の資源を大量に消費しない
  一般に移動型計算機は資源が乏しいので、資源を大量に消費しない事が望ましい。
- 容易に導入できる
  導入が容易である事が望ましい。

容認してもよいこと

- 全てのファイルに書き込みできなくてもよい
  1 ユーザでの使用が多いので、書き込みが出来るのは使用中のユーザのホームディレクトリ程度でよい。
- ファイルの一貫性の保持が完全でなくてもよい
  ファイルサーバ上のファイルとの一貫性は極力取るべきだが、個人的なファイルの書き換えに関しては一貫性が保持出来なくなったときでも、ある程度修復が可能なので、自動的な保持が不可能な場合にはユーザにまかせてもよい。

PFS では、ここで述べた機能を実現する必要がある。

4.3.3　PFS の特徴

移動型計算機で利用するために必要な機能を満たすために、PFS に次のような特徴を持たせる。

- クライアントカーネルとのインターフェースとして NFS プロトコルを利用する。
  カーネルからファイルシステムを参照するのに最も容易で汎用性がある方法は、NFS プロトコルでの通信をカーネルと行なう事である。そこで、PFS では、導入を容易にするため、NFS プロトコルを利用する事にする。

- 積極的にキャッシュを利用する。
  通信量を減らし、通信が途絶えても利用できるためには、必要なファイルの内容がクライアント側に存在する必要がある。しかし、移動型計算機の資源は少ないので、必要なファイルのみをキャッシュ上に持つ事により、これを実現する。

- キャッシュに取り込むファイルをユーザが明示的に指定できる。
  計算機を利用するユーザの作業に必要なファイルを明示的に指定できる機構を用意し、十分な指定ができれば、作業に必要なファイルを必ずキャッシュ上に用意する事ができ、キャッシュミスを減らす事ができる。

- ホストの移動透過性は、VIP を利用する事により保証する。
  移動透過性の保証に既存の VIP を使う事により、PFS 自体の処理を単純にできる。

- ファイルの一貫性保持の失敗をある程度容認する。
  ファイルの一貫性の保持の失敗をある程度容認し、失敗したときにはユーザに対処を求める事により、他のファイルシステムとの共存を容易にする。

4.4　実装

4.4.1　PFS の構造

PFS はクライアントカーネルにとってはユーザーレベルで動作する NFS サーバとして働く。

しかし、先に述べたように NFS プロトコルをインターネット上で使うには問題が多いので、 PFS の構成要素を、クライアントマシン上でキャッシュを元に NFS サーバとして動作する Cache Server (CS) と、ファイルサーバマシン上のファイルを CS に提供する Master Server (MS) の 2 つに分割する (図 4.1)。

これは、NFS プロトコルが公開されており利用しやすい事と、非常に普及しておりほぼ全ての UNIX オペレーティングシステムで動作している事から決定した。

また、CS と MS の間の通信には、信頼性が高く通信の連続性が保証される TCP を利用し、移動透過性を後述する VIP を使う事により保証する。この通信のアプリケーション層としてのプロトコルには独自の Cache Update Protocol (CUP) を用いる。VIP を利

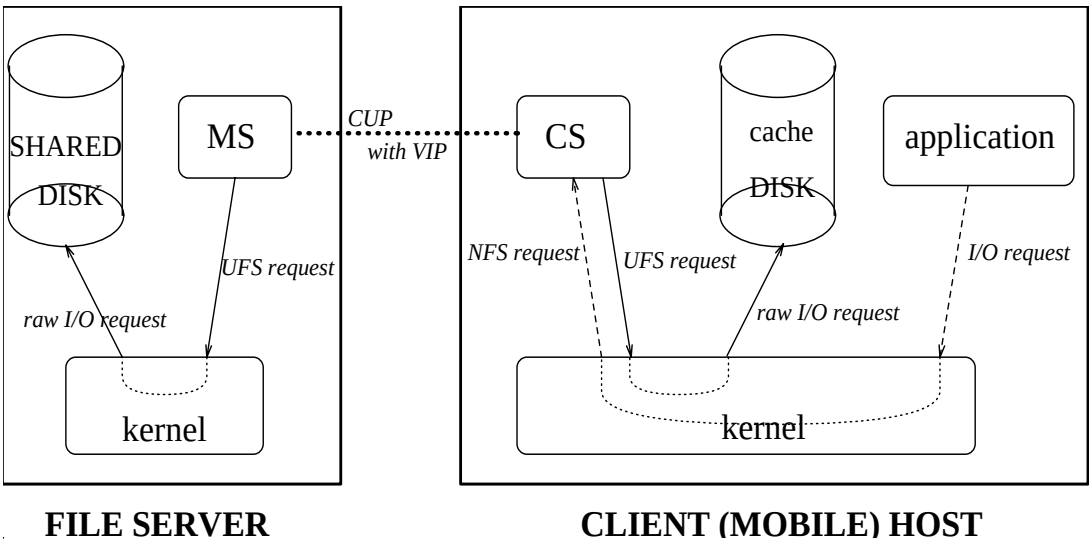

図 4.1: PFS の構造

用することにより、アプリケーション層に存在する CUP は移動に関する情報を扱う必要がなくなり、処理が簡素化される。

4.4.2 CS の動作

CS は、クライアントマシン上で動作し、キャッシュディスク上のファイルを使ってクライアントカーネルからの NFS 要求に答える。キャッシュ上に無いファイルに関する要求が来た場合には、NFS 要求を待たせておいて、MS に対してそのファイルの転送を要求し、キャッシュ上にファイルの用意ができてから NFS 要求に対する返答を行なう。この機構により、たとえキャッシュに存在しないファイルに対しても、最初の操作に時間がかかるだけで透過的に操作できる。

さらに、通信断に備え、キャッシュにユーザの指定した必要なファイルをあらかじめ用意しておく機構を用意する。これにより、作業に必ず必要なファイルのみを選択的にキャッシュに保持することができ、キャッシュミスを減らすことが期待できる。

ファイルサーバとのファイルの整合性を保つために、ファイルのアトリビュートを比較することによる整合性チェックを行なう。ファイルに対する操作毎にチェックを行ない、操作中のファイルが極力最新の物になるようにする。また、定期的にキャッシュされている全ファイルのチェックも行なう。

ファイルの変更など書き込みも、キャッシュに対して行なわれ、その後ファイルサーバに反映される。

この方法では、読み込み時間の遅延等によりファイルサーバとの不整合の起きる可能性がある。しかし、ユーザのホームディレクトリの書き込みを行なうのはユーザ自身であることがほとんどなので、書き込みを行なえるのをクライアントマシンを利用してい

るユーザのホームディレクトリのみであると限定すれば、不整合が問題になりにくいと予測される。また、不整合が起きたとしても、ファイルの修復に必要な情報を提供すれば、多くの場合ユーザ自身またはプログラムによる補正が可能である。不整合が問題になりにくいことは、AFS, Coda に関する論文 [182] [183] でも述べられている。

4.4.3　MS の動作

MS はファイルサーバ上で動作し、CS から要求のあったファイル操作を行なう。

また、CS に提供したファイルを記録しておき、CS からの要求に応じて CS へ転送した時のアトリビュートと、現在のアトリビュートを比較し、変更が行なわれていれば CS にそれを通知する機構を持つ。

4.4.4　実際のプログラム

実装は、SONY NEWS NEWSOS-4.2C 上で行った。RPC の呼び出しインターフェースと XDR に従った符号化と復号化には、OS に付属の `rpcgen` コマンドによって自動生成したプログラムを利用した。

NFS プロトコルの RPC 言語による記述ファイルには Sun Microsystems Inc. が配布している `rpcsrc.4.0` 中の `nfs_prot.x` を利用した。

4.5　評価

4.5.1　速度

評価には、クライアントとして SONY NEWS NWS-3250、サーバとして SONY NEWS NWS-1750、通信路に 9600bps の SLIP 回線を用いた。

図 4.2 は、20 個のエントリを持つディレクトリを `ls -l` コマンドで周期的に読み出すのにかかった時間を、`time` コマンドで測定し、NFS と比較したものである。読みだし周期には 1 秒と 60 秒を用いた。

PFS、NFS 共に初回はファイルサーバからの読み込みに時間がかかるが、PFS では以後の読み出しは高速である。それに対し、NFS では周期が 60 秒ほどになるとキャッシュの効果が無くなり、毎回低速な読み出しになっている。

このように低速な回線でも良好な応答性が確認できた。

4.5.2　必要な機能の実現状況

ここでは、必要であるとした機能が、どのように実現されたかについて述べる。

- 従来の NFS と共存できる
  MS はファイルサーバマシン上では 1 アプリケーションとして動作し、CS が動作し

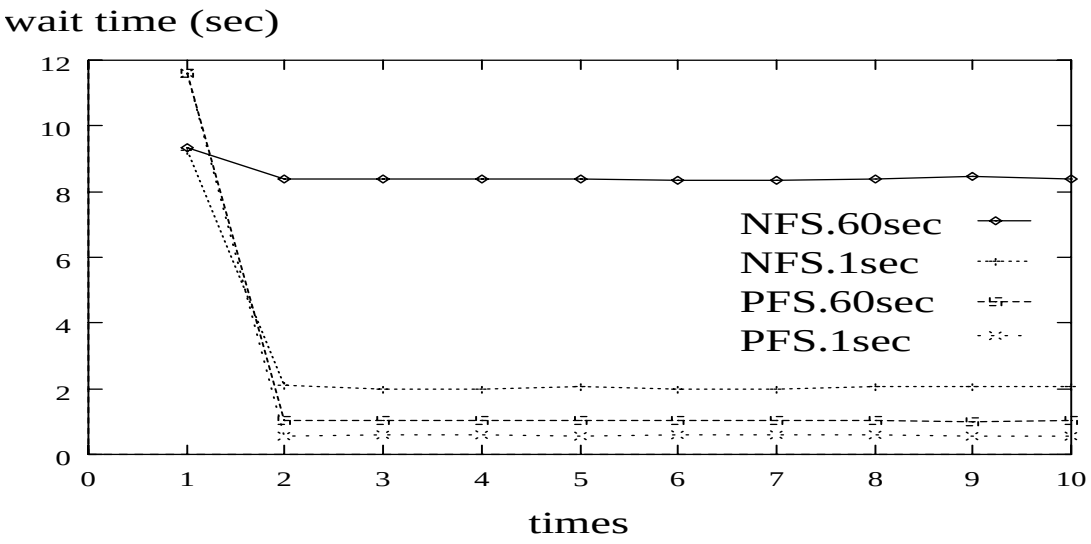

図 4.2: NFS と PFS の速度

ているクライアントマシン以外には影響を与えないため、従来の NFS との共存が可能となった。

- 遅延の大きなネットワークを使用している時でも良好な応答性が確保できる
  ネットワークの遅延が応答性に影響を与えるのはキャッシュに存在しないファイルやディレクトリを読み込む時だけであり、普段は良好な応答性が確保できた。

- 通信が突然途絶えても作業を続行できる
  必要なファイルはキャッシュディスク上に存在し、通信が途絶えてもキャッシュ上の情報だけで作業を続行できるようになった。

- ネットワーク上を移動してもファイル操作を継続できる
  VIP を利用する事により、クライアントの物理的位置が変化しても論理的な位置は変化しないので、通信の連続性が保たれ、ネットワーク上を移動してもファイル操作が継続できた。

- ユーザーにとって、本来のファイルシステムと同じ操作ができる
  ユーザにとっては、従来と同様の NFS サーバとしてふるまうので、従来の NFS と全く同じ操作ができるようになった。

- クライアント側の資源を大量に消費しない
  クライアントでは、キャッシュ上に必要なファイルのみを保持すればよくなり、最低限のディスク容量で動作するようになった。

- サーバとキャッシュの一貫性を保つ
  通信が可能な間は常にキャッシュの一貫性保持を行ない、通信が一時的に途絶えて

も通信復旧後に一貫性保持を行なうので、サーバとキャッシュの一貫性をある程度保つ事ができるようになった。

- 作業に必要なファイルをあらかじめ指定できる
  WST に作業に必要なファイルを登録する事により、作業に必要なファイルを指定しておく事ができるようになった。

- 容易に導入できる
  全てアプリケーション層で実現されており、クライアント側で NFS が利用できれば導入が可能となった。

このように、移動型計算機にとって必要とされた機能を満たす分散ファイルシステムを構築することができた。

4.6　今後の課題

- キャッシュするワーキングセットの決定
- CS コントロール用アプリケーションの作成
- ファイル整合性保持失敗時の処理の改良
- 通信量の削減
- データの暗号化

などが今後の課題である。

第 5 章

情報資源発見

インターネットにおいては異機種の計算機同士があらかじめ定められた相互接続の為のプロトコルを使用することによって情報資源にアクセスすることができる。情報資源をアクセスする為のプロトコルは幾つか存在するが現在もっとも広く利用されているのに FTP と呼ばれるファイル転送プロトコルがある。

FTP を使って情報共有を行う方法として anonymous FTP(匿名 FTP) がある。anonymous FTP では anonymous というユーザ名を使用することで認証の為のパスワードを入力すること無く誰でも遠隔地の計算機にログインして、そこに蓄積されているファイルにアクセスすることが可能である。

現在 anonymous FTP のサービスを行っている計算機は世界各国に数多く存在し利用者が参照できる情報は数百 GB から数千 GB にものぼる。利用者はこのような多量の情報資源にアクセスすることが可能であるにもかかわらず、その量があまりにも莫大であるために、どこにどのような情報資源が公開されているかということを知ることは極めて困難である。

そこで、ネットワークの広域化および高速化に伴い、情報資源を効率良く発見する為の仕組みがさらに重要になると考えられる。本報告では Internet 上に分散して存在する情報資源に着目し、情報資源を効率良く発見するためのシステムに関して述べる。

5.1 広域ネットワーク環境における情報資源共有

本節では広域ネットワーク環境における情報資源共有の幾つかの手法および現在標準化が勧められている情報資源に対する名前付けに関して述べる。

5.1.1 インターネットに環境における情報資源

インターネットに接続されている全ての計算機は対等な関係で相互に通信を行うことが出来る。ある計算機からみてインターネットに接続されている計算機は机上の計算機も、スーパコンピュータも IP アドレスという同じ形式のアドレスを持つ計算機として同様に見える。(図 5.1)

インターネット環境においてはある計算機は他の計算機から提供される情報資源を利用できると共に他の計算機に対して情報資源の提供を行うことができる。

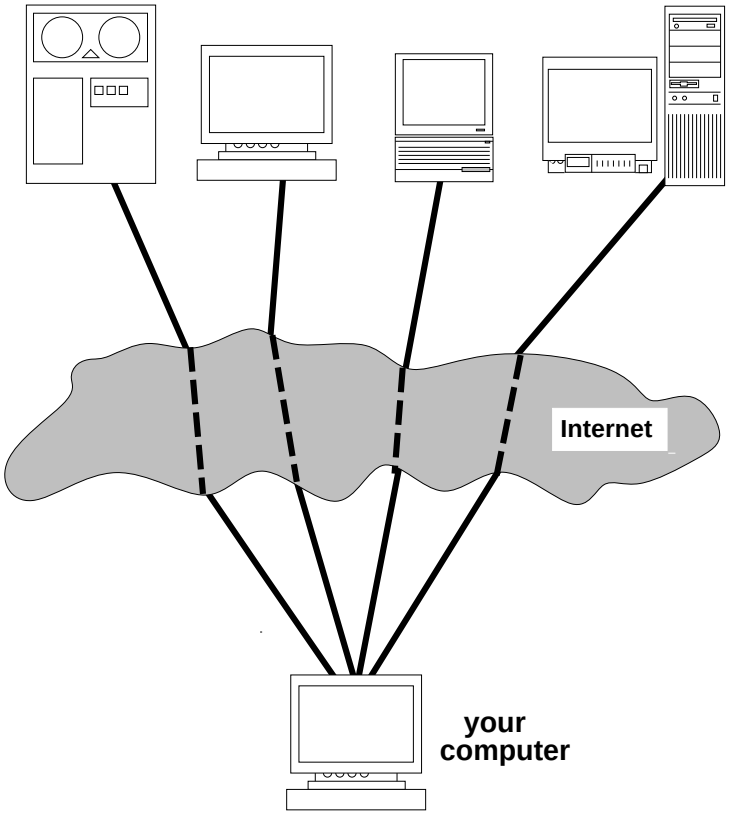

図 5.1: インターネットの概念モデル

5.1.2　情報資源の単位

インターネットを通じて利用可能である情報資源を次の 3 つに分類する。以下に粒度の小さい順に列挙する。

ファイル

可変長バイト列からなるデータ。(粒度: 小)

ボリューム

情報資源をある目的、思想に基づいて集めたもの。例として、anonymous FTP のディレクトリが挙げられる。(粒度: 中〜大)

情報資源データベース

情報資源に関するポインタを蓄積しておき、キーワードによる検索もしくはメニューの使用により、利用者に情報資源のポインタまたは情報資源を提供する為のシステム。例として archie[184],gopher [185],WAIS[186], WWW(World Wide Web)[187] 等がある。(粒度: 大)

5.1.3 情報資源の利用

現在広域ネットワーク環境において情報資源共有に使用されている主要システムに関して述べる。

FTP(File Transfer Protocol)

FTP[115] はその名が示す通りファイルを転送するためのプロトコルである。 FTP は任意の地点にあるインターネットに接続された計算機間でファイル (任意長のバイト列) 転送を可能にする。一般に FTP を使用する際にはログイン手続きが必要であるが、それを省くことで情報資源を公開し共有する方法の一つとして使用できる。これが anonymousFTP(匿名 FTP) と呼ばれるサービスである。現在 anonymous FTP はインターネットにおいて最も広く使われている情報資源共有の方法である。

NetNews: NNTP(Network News Transfer Protocol))

インターネットには NetNews と呼ばれる分散型の掲示板システムがある。この掲示版システムは USENET と呼ばれる論理的ネットワーク網によって成りたっている。メッセージはニュースサーバ間のバケツリレー式転送によって運ばれている。メッセージはニュースサーバに蓄積されユーザはニュースサーバにアクセスしてメッセージを読むことができる。インターネット環境においてはニュースサーバ間のメッセージの中継およびユーザがニュースサーバにアクセスし記事を参照するために NNTP[167] というプロトコルを用いている。

gopher

gopher[185] はメニューベースの情報資源を拾い読みする為のシステムである。システム形態はクライアントサーバ型、かつサーバ同士をリンクで結んだ分散型のシステムである。ユーザは gopher クライアントを使用して gopher サーバに接続する。サーバから送られる階層構造を持つメニューをたどることにより目的とする資源を得ることができる。

WAIS(Wide Area Information Service)

WAIS[186] はサーバ毎に蓄積されている文書ファイルに関しての効率的な検索を行うための索引を自動生成し、利用者からの検索要求に従って文書ファイルを検索し提供するシステムである。

WWW(World Wide Web)

WWW[187] はハイパーテキストと呼ばれる技術をベースにした比較的新しいサービスである。WWW はクライアントサーバ型のシステムでその間の通信には HTTP[188] を使用する。クライアントはサーバにアクセスしてハイパーテキスト形式のファイルを読み出す。このハイパーテキスト形式のファイルは HTML(Hyper Text Markup Language) [189] で記述されている。HTML では文書の中に URL[190] を使用して情報資源へのポインタを記述することができる。

5.1.4　情報資源の表現

広域ネットワーク上には種々の情報資源が存在し、それらは種々のプロトコルを用いてアクセスされる。現在プロトコル毎に異った表現形式が使用されていることから、広域ネットワーク環境で情報資源を統一の取れた形式で表現することができない。従って種々の情報資源に対して統一の取れた名前付けを行う必要がある。

現在 IETF(Internet Engineering Task Force) で標準化が進められている名前付けは URL(Uniform Resource Locators)[190] と呼ばれるやり方である。この節では URL に関して述べる。

URL の目標

情報資源を表すために、以下の 4 つの条件を満たすような名前付けを行う。

- 名前は情報資源を表すのに十分な長さを持つ。
- 名前の構文は新しいプロトコルや新しいディレクトリ技術が現れた際に用意に適合できる。
- 名前はその情報資源をアクセスするための十分な情報を含む。
- 名前は一意である。

URL の構文

URL はまず初めにはプロトコルを表す接頭語で始まる。その後にコロン ":" に続いてオブジェクトの名前が来る。オブジェクトの名前はプロトコル名に依存する。オブジェクト名が "//" で始まるとき "//" の後より次に現れる "/" までは、ユーザ名 (オプション)、ホストが持つインターネットアドレスおよびポート番号から成る。ユーザ名 (オプション) がある時にはユーザ名とホスト名の間に "@" が挿入される。ホスト名は FQDN(Fully Qualified Domain Name) もしくは 4 つの数字をドット "." で連結したフォーマットで表現される IP アドレスである。ポート番号はコロンに続く数字であらわす。ポート番号を省略した時にはプロトコル毎に決められているデフォルトの値が仮定される。 "/" のあとはパスが続く。これはプロトコル毎に定められる情報資源を示す "パス" である。

URL で表現できる情報資源

幾つかの標準化されたプロトコルでアクセスできるオブジェクトは URL を使用して表現できる。ここにその例を示す。

FTP

FTP でアクセス可能な情報資源の URL は ftp: で始まる。ポート番号が与えられなかった際はデフォルトのポート番号 (20 番) が仮定される。ユーザ名が URL の構文中に含まれない場合は anonymous FTP が仮定され、ユーザ名は

“anonymous”,“password” は利用者の Email アドレスが使用される。その後にファイルのパス名が続く.

(例) anonymous FTP サーバ ftp.cse.kyutech.ac.jp に格納されているファイル
/pub/misc/README は、

ftp://ftp.cse.kyutech.ac.jp/pub/misc/README

で表現される。

HTTP

HTTP でアクセス可能な情報資源の URL は http:で始まる。その後に情報資源を特定するパス名が続く。

(例) http サーバ kiwi.cse.kyutech.ac.jp が提供しているハイパーテキスト oielab/index.html は、

http://kiwi.cse.kyutech.ac.jp/oielab/index.html

で表現される。

gopher

gopher プロトコルでアクセス可能な情報資源の URL は gopher: で始まる。パス部の最初の 1 文字は gopher プロトコルで使用される “type” フィールドを示す。その後に gopher プロトコルにおいて使用される情報資源を示す為の “selector string” が続く。

(例) gopher サーバ gopher.ncc.go.jp が提供している type=0 および selector string=“/INFO/net-info/trouble-db/0README” で特定出来る情報資源は、

gopher://gopher.ncc.go.jp/0/INFO/net-info/trouble-db/0README

で表現される。

URL を使用することで異なるプロトコルでアクセス出来る情報資源の名前を統一された構文で記述することが可能になる。また URL は情報資源に対するアクセス方法 (情報資源をサービスしているホスト名、情報資源のパス名) も記述しているので URL を参照するのみで情報資源をアクセスする為に必要な情報を得ることが出来る。

5.2 広域ネットワーク環境における情報資源発見

この節では広域ネットワーク環境における情報資源発見に関して述べる。関連する研究に関して述べながら未解決の問題を明らかにする。

5.2.1　情報資源発見の手順

利用者が情報資源発見を行う手順は情報資源の「認識」そして「発見」と言う 2 つのフェーズに分けることが出来る。以下にこの 2 つのフェーズに関して述べる。

情報資源の認識

情報資源を得る為にはまず第一に利用者がどのような情報を欲しているかを明確にし利用者が得ようとする情報資源を特定しなければならない.

情報資源の発見

既に利用者が必要とする情報資源を認識し、自分の欲する情報資源がインターネット上に存在していることがわかっているのであれば、その情報資源を示す名前をキーにして情報資源のポインタを見つけ出すことで利用者は情報資源に対してアクセスすることが可能となる。

インターネットにおいては、アクセスの集中を防ぎ回線の利用効率を高めるために、情報資源を複製して分散配置することがしばしば行なわれている。ゆえに、情報資源を発見する段階で利用者は複数のポインタを得るかも知れない。多くの人が必要とする情報資源は分散配置されることが多いので、このように複数のポインタを得た場合には利用者から見て情報資源へのアクセス際に必要とされる計算機資源またはネットワーク資源のコストを最も小さくするような情報資源を選択することにより最小限のコストで目的とする情報資源を手に入れることが可能となる。

5.2.2　従来の研究

この節では情報資源発見に関連するこれまで行われてきた研究の幾つかに関して述べる。

archie

archie[184] はインターネットにおいて anonymous FTP で公開されている情報資源のポインタデータベースを自動で構築および管理するためのシステムである。archie は McGill 大学の Alan Emtage, Peter Deutsch らによって作成された。

archie サーバは次にあげる 3 つの部分より構成される。それらは DGC (Data Gathering Component), DMC(Database Maintainance Component) および UAC(User Access Component) である。DGC は予め登録された anonymous FTP サーバからそれぞれのサーバのファイル名リストファイル (UNIX コマンド “ls -lR” を使用して得られるもの) を定期的に収集し、それを raw listing file という形式で格納する。DGC が格納した raw listing file は DMC によって一貫性が確認された後データベース化される。そのデータベースは、UAC によって参照される。利用者は 3 種類のユーザインターフェース (telnet インターフェース, Email インターフェース, Prospero[169] クライアントインターフェース) のいずれかを介して UAC に対して

し問い合わせを行う。問い合わせは名前またはパターンでもって行われる。問い合わせに対して UAC はその名前またはパターンにマッチするファイル名を検索し、それを持つ情報資源のポインタを出力する。

以上が archie の概要である。archie には次に挙げる問題点がある。

- archie が取り扱う情報資源の単位は粒度の小さいファイルである。
- ファイルに関する情報として `ls -lR` で得られるわずかな情報しか保持していないが為に利用者が情報資源を特定する場合の情報が不足する。
- 集中型のデータベースであるのでアクセスの集中を招くこととなる。
- 集中型のデータベースであることからデータの集中を招くこととなる。

WAIS

情報資源発見のためのシステムは WAIS を用いても構築することが可能である。

WAIS を情報資源発見に利用するためには、まず情報資源に関する情報をドキュメントとして作成しておく必要がある。そして、WAIS サーバの機能を使ってそれらを検索することで資源発見のシステムを実現することができる。

WAIS の例として directory-of-servers というデータベースが、`quake.think.com` にて公開されている。これはインターネットからアクセスすることができる WAIS サーバのデータベースである。これは各 WAIS サーバに関して WAIS データベースにアクセスする情報としてホスト名、IP アドレス、TCP ポート番号、データベース名、データベースに関する説明等の項目をファイルにしておき key を与えることによってファイルを検索し、そのリストを返すことができる、また要求に応じてそのファイルを提供している。

もう一つは anonymous FTP アーカイブを検索する目的で現在利用可能な WAIS サーバの例である、wais.oit.unc.edu で稼働している WAIS サーバが挙げられる。このサーバでは wuarchive.wustl.edu で提供しているアーカイブの情報を”ls -l” の形式でドキュメントとして格納している。利用者はファイル名 (またはその一部分) をキーとして与えることにより目的とするファイルのパスを検索することができる。

また、USENET のニュースグループでアーカイブに関する情報提供に使用されている comp.archives に投稿された記事を集めてワード検索を可能にしている WAIS サーバが archive.orst.edu において稼働している。

WAIS を情報資源発見システムと使用するとき、それは以下の特徴を持つ。

- 情報資源発見システムとして有用に利用できるか否かは、WAIS サーバがローカルに持つドキュメントに依存することとなる。
- WAIS サーバは内部的にインデックス作成することで高速なワード検索を可能にしている。
- WAIS は集中型のデータベースシステムでありサーバ間では連係していない。

5.2.3　未解決の問題

これまで既存のる 2 つの情報資源発見機構に関して述べてきた。これらの機構には有用な点もあるが、広域ネットワークが急激に拡大しつつあることなどを考えると解決しなければ問題を幾つか持っている。以下に広域ネットワーク上における情報資源発見に関する問題を示す。

1. 情報資源の粒度

   archie では取り扱う情報資源の単位は粒度の小さいファイルである。急激に拡大をしつつある広域ネットワークにおいて情報資源発見の単位が粒度の小さいファイルであることは、将来システムの破綻をまねくことになると考える。粒度の小さい情報はそれだけ多量に存在することから，蓄積すべきデータ量の増大をもたらし蓄積および検索という処理を行う際に困難を招くことになる。

2. 情報資源認識の為の手がかりの不足

   archie は `ls -lR` で得られる情報しか手がかりがないために、情報資源に関するデータベースを構築しそれを情報資源の認識に利用するさいの情報極めて少ない。

   情報資源の認識を効率良く行う為には有用な索引作りが必要となる。一般に、ファイルの内容、表現形式および格納形式によって最適な索引を作成する方法は異る。

3. 特化された情報資源発見機構

   archie が取り扱うことのできる情報資源は anonymous FTP のファイルのみである。これまで 5.1節で述べたように現在情報資源共有の為には様々なプロトコルが使用されているにも関わらす、それらを全て取り扱えるような情報資源発見機構がない。そこで種々の情報資源を発見する為の機構が必要である。

4. 集中型のシステム

   現在の情報資源発見システムを見てみると集中型のシステムとして構築してあるものがほとんどである。archie, WAIS いずれも集中型のシステムである。集中型のシステム形態を取ることによって、データベース管理および検索の機構が簡単になる。しかしながら、集中型のシステムはデータベースの集中化、アクセスの集中化、データベースの集中化という点が問題となる。インターネットに接続される計算機の数およびインターネットの利用者の数はますます増加しているので、アクセスの集中、データベースの集中はサーバおよびそれに接続する回線に過大な負荷をもたらすこととなり望ましくない。

次節以降は、これらの問題点を解決する為の広域ネットワーク環境における新しい情報資源発見機構に関して述べる。

5.3 HORD の設計

この節ではこれまでに述べてきた問題点を踏まえながら、新しい広域ネットワーク環境における情報資源の発見システム HORD(HOri's Resource Discovery system) の設計に関して述べる。

5.3.1 HORD の設計方針

ここでは HORD の設計に関して述べる。現在使用されている情報資源発見機構の問題点を解決する為に HORD では以下の方針で資源発見機構を実現する。

情報資源の単位

資源発見の単位は次の 2 つとする。

- **ボリューム**
- **情報資源データベース**

5.1.2節であげた分類のうちファイルは資源発見の単位として扱わないこととする。その理由は、粒度の小さい情報資源を単位として取り扱うことは、それを単位とする情報資源の数の増加に伴ってデータベースの量が増加し資源発見時の検索処理が困難になること。また、予め何らかの意味付けを持ってまとめられた比較的粒度の大きいボリュームおよび情報資源データベースという単位で取り扱うことは有用であると考えるからである。

情報資源の属性

HORD では情報資源に属性を設けその属性を利用することで情報資源の発見を効率よく行えるようにする。情報資源毎に以下に挙げる属性を付加する。

- **名前 (Name)**[文字列]
  情報資源の名前であり URL で表現する。
- **複製元の名前 (Original Name)**[文字列]
  この情報資源が情報資源の複製であった場合複製元の名前を URL で表現する。
- **説明 (Description)**[文字列]
  情報資源に関しての説明を記述する。
- **版番号 (Version)**[32bit 符号無し整数]
  情報資源または情報資源の属性がが改訂された場合に改訂前の情報資源との違いを示す為の整数で、情報資源が改訂されることに増加させる。
- **アクセス制御リスト (Access Control List)**[文字列]
  情報資源のアクセス制御を記述する。次に挙げるフォーマットを使用する。

– アクセス制御はログイン名とホスト名を “@” で連結した文字列で表されるユーザの集合を単位とする。これには簡単なワイルドカード “*” を使用できこれは全ての文字列にマッチする。
– アクセスを許可しない場合ユーザ集合には “!” を前置する。
– アクセス制御の単位は “,”(カンマ) で区切って並べることができる。

(例)

1. 全てのユーザに対してアクセスを許可する場合 *@* と記述する
2. 九州工業大学内の計算機 (ホストアドレス *@*.kyutech.ac.jp) のユーザにのみアクセスを許可する場合 *.kyutech.ac.jp と記述する。
3. 九州工業大学内の情報科学センターの計算機にアドレスを持つユーザの中で電子情報工学科および知能情報工学科の学部生 (ログイン名の先頭が’p’ または’q’ で始まるユーザ) にのみアクセスを許可する場合。
p*@*.isci.kyutech.ac.jp,q*@*.isci.kyutech.ac.jp

5.3.2　情報資源発見機構の構成

従来の広域ネットワーク環境における情報資源発見機構の形態は集中型データベースのモデルが多く見られる。ここでは、集中型データベースに対して、分散型データベースを比較しながら HORD においてどのようなシステム形態を使用するべきであるか考察する。

集中型データベース

広域ネットワーク環境上の集中型データベースの形態は archie,WAIS といった従来の資源発見機構が使用している形態である。集中型データベースには次に挙げる利点がある

- 情報の登録および更新が容易になる。
- 情報が一箇所に集中しているので全数探索が容易になる。

しかしながら、集中型データベースには次に挙げる欠点がある。

- データベースに対するアクセスが集中することから、データベースシステムに高負荷がかかる為に高性能の計算能力が必要であるし、回線にも高負荷がかかる。
- システムがダウンした時、または利用者から集中型システムまでの通信回線のどこかに異常が生じデータベースに対するアクセスが不可能となった場合データベースシステムを利用することができなくなる。
- データベースはその管理人によって集中管理されているので誰もがデータベースの保守を行えるわけではない。

これらの欠点を補う為に、archie においては広域ネットワーク環境上にデータベースの複製を作成し配置することが行われている。これにより、アクセスの集中による負荷の増大およびシステムがダウンしたときの可用性は補われている。しかしながら、複製のそれぞれが同じデータベースの塊を持つことになるのでデータベースの集中を避けることは不可能である。現在の使用されている集中型の情報資源発見機構ではデータベースの集中と言う大きな問題を抱えている。集中型の形態は現在のところうまく機能しているものも広域ネットワークの拡大によるデータベースの増加によって破綻する可能性がある。

分散型データベース

分散型データベースに関して考察してみる。分散型データベースには集中型データベースと比較して次に挙げる利点がある。

- データベースが分散しているので、アクセスも分散され集中型よりもデータベースあたりの負荷が低い。
- データベースの分散化と共にデータベースの管理も分散化させることができ、それぞれのデータベースは独自に管理することが可能になる。
- 集中型と比較してデータベースを分散して配置することで利用者とデータベース間の通信コストは一般的に見て減少することとなる。
- 分散して存在するいずれかのシステムが障害を起こして使用不能になったとしても情報検索の機能全てが停止するのではなく一部分だけの停止のみで済むようにシステムを構築することも可能となる。

このように分散型データベースの使用は幾つかの利点がある。しかしながら、次に挙げる欠点も存在する。

- データが分散して配置されているので、全数検索を行うことが困難となる。
- 分散して配置されているデータに対して検索を行う場合ネットワーク遅延が検索の際の速度に影響を与える。
- データの複製が分散して配置される場合、データの一貫性の問題が生じる。

ここで分散データベースに関する上に挙げた欠点は広域ネットワークにおける資源発見の際にどのように影響するか考えてみる。まず、情報資源の発見の為の情報は一貫性を厳密に要求するものではないことから、一貫性の保証は多少緩くても許される。次に、データベースが分散している為に全数検索が困難になるということであるが、情報資源発見を行う際に毎回全数検索をする必要はないと考える。多くの利用者がそれを欲している情報資源はアクセスの集中化を防ぐ為に予めミラーされていることが多い。よって，利用者から見てある限定された範囲で検索を行ってもほとんどの場合資源を発見することができる。

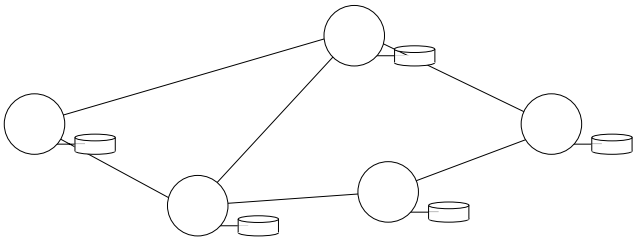

図 5.2: 水平型分散 (網構造)

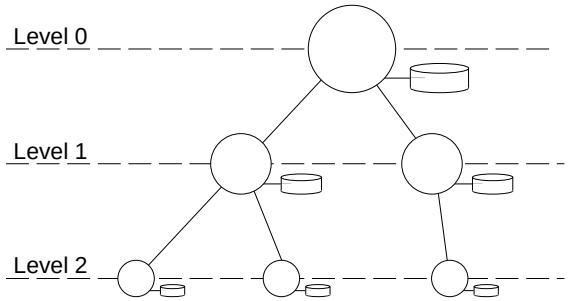

図 5.3: 垂直型分散 (木構造)

従って、広域ネットワーク環境において情報資源発見機構を設計する際には情報資源の特性から厳密な一貫性を要求されることはなく困難を伴う問題とはならないと考える。よって、HORD を分散システムとして構築することにする。

ここで分散システムの形態に関して考えてみる。分散しているデータベース間の関係か網構造をなす水平型分散 (図 5.2) もしくは、データベース間の関係が木構造をなす垂直型分散 (図 5.3) あるいは水平型と垂直型の混合である混合型分散に分けることができる。[191]

広域ネットワーク上の情報資源はある体系に従ってまとめられており従属関係があるというわけではなく、互いに独立な関係であるので HORD では水平型分散の形態で分散システムを構成する。

5.3.3　HORD の構成

これまで述べてきた理由により HORD は水平型の分散データベースとして実現する。HORD は図 5.4に示すように情報資源データベースの管理および他のサーバと協調してデータの検索を行うサーバ部とサーバ部に対して登録、更新、検索等の要求を発するクライアント部から構成されるクライアントサーバ型の構成である。以後、HORD を構成する分散データベースサーバ部はファイルほど粒度の小さくないボリュームと呼ばれる

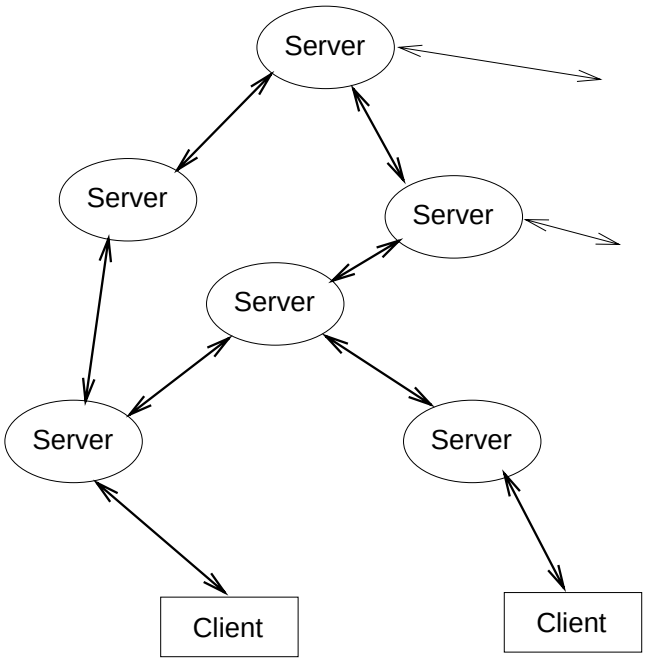

図 5.4: HORD の構成

まとまった情報資源を単位として扱うことから VSD(Volume Server Discovery) と呼ぶ。

5.3.4 データベースの配置

HORD のデータベースサーバ VSD の配置に関して述べる。まず始めに広域ネットワークを複数のエリアに分割する。そして分割されたエリア毎に VSD を配置し VSD にはそのエリア内にある情報資源に関するデータを蓄積する。

5.3.5 情報資源発見アルゴリズム

HORD における情報資源発見の仕組みを述べる。あるエリアに配置されている VSD は他の 1 つ以上の隣接する VSD のアドレスを記録している。HORD は以下のアルゴリズムを使用して検索を行う。

利用者はまず最寄りの VSD に検索要求を送る。検索要求を発行する際に利用者は検索要求パケットの生存時間パラメタ TTL(Time To Live) および 3 つの検索項目 (キーワード、ロケーション、プロトコル) を指定する。検索要求を受けた VSD は検索要求にしたがってローカルに持っているデータベースを検索するとともに、ある単位時間だけ待って予め記録されている隣接 VSD に対して検索要求を転送する。転送の際に利用者最寄りの VSD は検索要求の中に、自分のアドレスを付加する。検索の結果ヒットした場合 VSD は情報資源の情報を利用者最寄りの VSD に送る。この結果ヒットした情報は利用者最寄

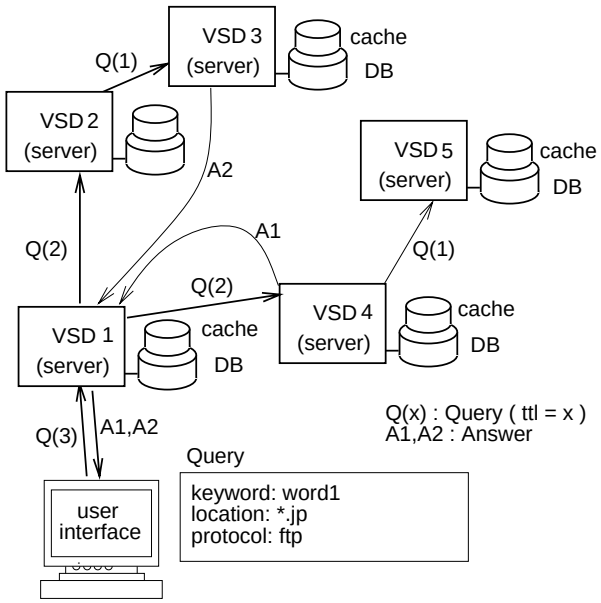

図 5.5: 情報資源発見のプロセス

りの VSD に集まることとなる。

このアルゴリズムで要求を行なうと検索要求が VSD の網を伝わって行くように見える。ここでなんら制限を設けなかった場合は検索要求が永遠に VSD 網を伝播することとなるので生存時間 (TTL: Time To Live) という変数を設けそれを防ぐ。VSD が隣接 VSD に対して検索要求を転送する際に決められた単位時間分の値を TTL から減じてから転送する。その値が 1 より小さくなった場合は転送を行わない。利用者最寄りの VSD では TTL だけ待ったあと、他の VSD から集まった情報を利用者に返す。

検索の結果情報が得られた場合には VSD はその情報をキャッシュする。これによって、一度検索でヒットした情報が再び検索される際の応答速度を改善する。

この動作を例を挙げて説明する。(図 5.5) 利用者は keyword, location, protocol を指定して検索要求を最寄りの VSD1 に送る。このとき TTL=3(初期 TTL) としている。VSD1 でこの要求に従いデータベース検索を行うとともに、隣接 VSD である VSD2 および VSD4 に 1 単位時間だけ待った後 TTL の値を 1 減らし VSD1 のアドレスを付加して転送する。VSD2、VSD4 では TTL=2 の要求を受け取るのでそれから 1 だけ減じ TTL=1 の検索要求を各々の隣接サーバである VSD3 および VSD5 に送る。VSD3 および VSD5 は TTL=1 の検索要求を受け取るが、TTL から 1 を減じた結果が 1 よりも小さくなるので隣接 VSD への検索要求転送は行わない。

各 VSD が検索を行った結果、VSD2 および VSD4 のデータベースで該当する情報が見つかったので、その結果は利用者の隣接サーバである VSD1 に集まる。VSD1 は初期 TTL から算出される時間だけ検索結果が集まるのを待った後、検索結果を利用者に返す。

5.3.6 既存の情報資源発見機構との統合

HORD は従来情報資源発見に使用されている archie,WAIS,gopher などの情報資源発見機構をまた一つの情報資源の塊とみなし取り扱うことができる。よって、既存の情報資源発見機構を統合した情報資源発見を行うことができる。

5.4 HORD の実装

現在 HORD のプロトタイプを Sun ワークステーションの上で開発中である。HORD を構成する VSD および利用者インターフェースは SunRPC[192] を使用して通信を行う。HORD を広域ネットワーク環境において実装および運用するためには、情報資源データベースに入力する情報を収集する機構、情報資源データベースを更新する機構等のの設計開発が必要になると考える。

5.5 おわりに

この節では広域ネットワークにおける情報資源発見機構に関して述べた。この情報資源発見機構 HORD を使用することにより、分散型の情報資源発見機構を構築することが可能であり、今後ますます拡大を続ける広域ネットワーク環境において効率的な情報資源発見が可能になると考える。

しかしながら、将来においてより効率のよい情報資源発見を実現するためには、情報資源共有の方法を見直し情報資源発見を考慮した新しい情報資源共有の枠組を作り上げる必要があり、さらなる研究が必要であると考える。