グループコミュニケーションを支援する
情報共有システムの設計と実装

卒業論文 1998年度 (平成10年度)

指導教員

徳田 英幸
村井 純
楠本 博之
中村 修

慶應義塾大学 環境情報学部

石井 徹

卒業論文要旨 1998年度(平成10年度)

本研究では、移動計算機環境における回線の切断・高遅延状態が頻発するような状況でも、利用者が回線の状況を意識せずに情報共有の行えるシステムを設計・実装した。

移動計算機環境で情報共有を行う際、不安定な通信環境を想定していない情報共有システムでは、不測の回線切断によって情報を管理するサーバとの通信が不可能になる。サーバ集中型の情報共有システムでは、情報をサーバ上で直接更新するため、サーバとの通信が不可能になることで、システムが動作不能になる。こうした問題を解決するためには、予め移動計算機環境を想定した情報共有システムが必要となる。このシステムでは、回線の切断や、情報を管理するサーバホスト上の障害を回避する必要がある。複数ホスト上で同様の情報を保持して共有する分散型の情報共有システムでは、一部のホストや通信回線に障害が発生しても、システム全体への障害伝播を回避することができる。本研究では、移動計算機環境という、回線の状態が不安定な状況でのシステム動作を想定する。そこで、本研究では分散型の情報共有システムを採用してシステムの設計を行った。

多くのオペレーティングシステムでは情報をファイルに抽象化している。そこで、ファイルを共有することができれば、オペレーティングシステム上での情報の管理が容易になる。こうした理由から本研究では、ファイルを媒介とした情報共有システムとして、システムの設計を行った。

そして、一般的なオペレーティングシステムでの動作、導入の容易なシステム、という二つの目標を掲げて実装を行った。本研究で提供したシステムにより、移動計算機環境という特殊な環境での、円滑なファイル共有が可能となった。

慶應義塾大学 環境情報学部

石井 徹

Abstract of Bachelor Thesis Academic Year 1998

In this paper, the system that users can share information one another on any status of network, which has congestion, high delay in mobile computing environment is designed and implemented.

To share information in the mobile computing environment, the information sharing system that does not suppose communication link is unstable can not communicate servers that manage information due to unpredictable shutdown of link. The information sharing system in a centralized server scheme, every information is updated on the server. In case that client can not communicate with the server, entire of system must terminate. To solve such problems, the information sharing system, which is designed for the mobile computing environment, is required. It is necessary for such system to avoid any trouble caused on the server host that manage information and status of link.

The distributed system that keeps and shares similar information can avoid whole system failure, if several hosts or links down. In this research, the system that adopts the distributed information sharing system is adopted.

In many operating systems, information is handled as files, and this abstraction makes it easy to manage information. The information sharing system designed in this paper also uses files as abstraction of information.

The implementation goals are: a) the system will run on major operating systems, b) the installation of the system will be easy. The system makes it possible to sharing files smoothly, under special circumstance of mobile computing.

Keio University,

The Faculty of Environmental Information

Tohru Ishii

目 次

図 目 次

表 目 次

第1章 序論

1.1 背景

現在、計算機の小型化・軽量化が進み、様々な状況において利用者が計算機を携帯し、移動中・移動先で使用することができる。移動中や移動先で他の計算機とファイルのやり取りを行う場合、利用者は固定計算機環境で使用していた現存のファイル共有システムを利用できる。

複数の人間から成るグループ単位でミーティングや作業を行う際、これらのシステムを使用して情報の共有を行うことで、グループ作業を効率よく進めることができる。携帯型計算機の普及に伴い、こうしたシステムを移動中に使用する必要性が高まっている。しかし、計算機が移動することによって、固定計算機環境では想定できなかった次のような問題が浮上する。

例えば、離れた場所から人々が集まり、計算機を持ち寄ってミーティング記録の閲覧や資料の配布など、情報の共有を行おうとする状況を考える。情報共有を行おうとした際、各計算機の、サーバなどの設定が統一されていないと、情報の共有を行うことができない。離れた場所から集まって来た利用者たちが新たに設定を行う手間を考えると、グループの作業効率は落ちてしまう。こうした際に、持ち寄った計算機がすぐさま協調して情報共有を行うことができれば、利用者は設定の手間から解放され、円滑なグループ作業を行うことができる。

一方で、計算機をネットワークに接続する際には、状況に応じて様々な手段が用いられる。この際、移動計算機環境では、主に無線 LAN や携帯電話などを使用する。しかし、これらの通信メディアは固定計算機環境のものと比べて低速で高遅延であり、また、全く通信を行うことができない状況もあり得る。固定計算機環境では、こうした不安定な通信回線を想定していないことが多い。そのため、移動計算機環境で既存のファイル共有システムをそのまま使用した場合、長時間にわたる動作の停止や、システム自体の終了を招くことがある。

また、多くの分散型情報共有システムは、単一のサーバが情報を保有し、複数のクライアントがサーバに接続することで情報の取得・送信を行う構成になっている。不安定な通信環境でこれらのシステムを用い、情報の共有を行った場合、サーバへの経路、もしくはサーバ自体に障害が発生した時に、クライアントはサービスを受けることができない。この結果、システム全体の動作に悪影響が及ぶ可能性がある。

既存の通信システムは元来、固定型計算機環境において開発されたものである。これらのシステムは通信回線の状態が良好であることを前提として設計されており、不安定な通信状況に対応することができない。移動体通信においてファイルの送受信などを行うシステムを構築する際に

は、こうした通信環境を予め想定することが必要となる。

1.2　本研究の目的

このように、移動計算機環境では、複数の利用者から成るグループで携帯型計算機を使用し、情報共有を行うケースが存在する。移動計算機環境という特殊な環境では、従来の情報共有システムをそのまま使用することができない。そのため、移動計算機環境を考慮した情報共有システムが必要となる。

そこで本研究では、移動計算機環境という特殊な状況下での不安定な通信状態や、計算機の移動に伴う回線の切断を想定する。こうした環境に適した情報共有システムを提供する。このシステムの構築によって、グループ内で行う情報共有を支援することを 本研究の目的とする。

1.3　本論文の構成

第 2 章では、移動環境に適した情報共有システム設計のため、既存の情報共有システムを比較検討する。その結果から、移動計算機環境で情報共有を行う際に必要となる機能について考察する。

第 3 章で移動体を考慮した情報共有システムのための最適なモデルを考察し、システム設計に続くための指針を立てる。このモデルに沿った設計を第 4 章で行う。ここではシステムの概要と主な機能について説明する。

第 5 章では実際の実装について述べる。システムの内部動作・モジュール構成について解説する。そして、本研究で実装したシステムの評価を第 6 章で行う。この結果を考察し、第 7 章で本研究のまとめを行う。

第2章　情報共有システム

本章では、移動計算機環境に適した情報共有システムについて考察する。

まず、既存の情報共有システムについて調査を行う。この調査結果のもと、移動計算機環境で情報情報共有を行う際に必要となる機能について考察する。

2.1　グループコミュニケーション

本論文ではグループコミュニケーションという言葉を、「複数人の間で、計算機を利用して行う情報通信」と定義する。

本研究では、グループコミュニケーションのひとつの形として、携帯型計算機を持ち寄って構成されるネットワークでの情報共有を選ぶ。

すでに前章で述べたように、移動計算機環境における情報共有は、不安定な通信環境を想定していない現存のシステムを利用することが多い。しかし、現存のシステムでは、移動計算機環境で生じる不安定な通信状況や、不測の回線切断を支援していないものが多い。そこで、本論文ではグループコミュニケーションを行う環境を、移動計算機環境に限定する。そして、この環境における情報共有を支援するシステムを構築するための研究を行う。

2.2　ファイルを媒介とした情報共有

オペレーティングシステムが情報を扱う際、その多くはハードディスクやフロッピーディスクを使用して情報を保存する。情報をファイルに抽象化する事によって、利用者が情報の複製や編集を行うことが容易になる。また、現在のオペレーティングシステムの多くは情報をファイルに抽象化して管理しているため、アプリケーション上での管理も容易である。

このような理由から、本論文で提供するシステムでは、ファイルを媒介として情報の共有・管理を行う。

2.3 既存のファイル共有システム

本節では既存のファイル共有システムに注目し、その長所・短所を考察する。

2.3.1 NFS

NFS(Network File System) [1] は、Sun Microsystems Inc. で開発された共有ファイルシステムである。設計は、クライアントサーバモデルに基いている。ファイル更新・削除の際はファイルサーバに直接接続し、サーバ上のファイルを直接更新する。即時的なファイル更新を行うことができ、安定した回線では高速なファイル操作環境を提供する。

ファイルアクセス時にはサーバに接続する必要があるため、低速な回線での応答性の悪さが欠点として挙げられる。同様の理由から、回線の切断時には全く利用することができない。クライアントサーバモデルを用いているため、サーバに障害が発生すると、サービスを受けるクライアントの動作は全て停止する。

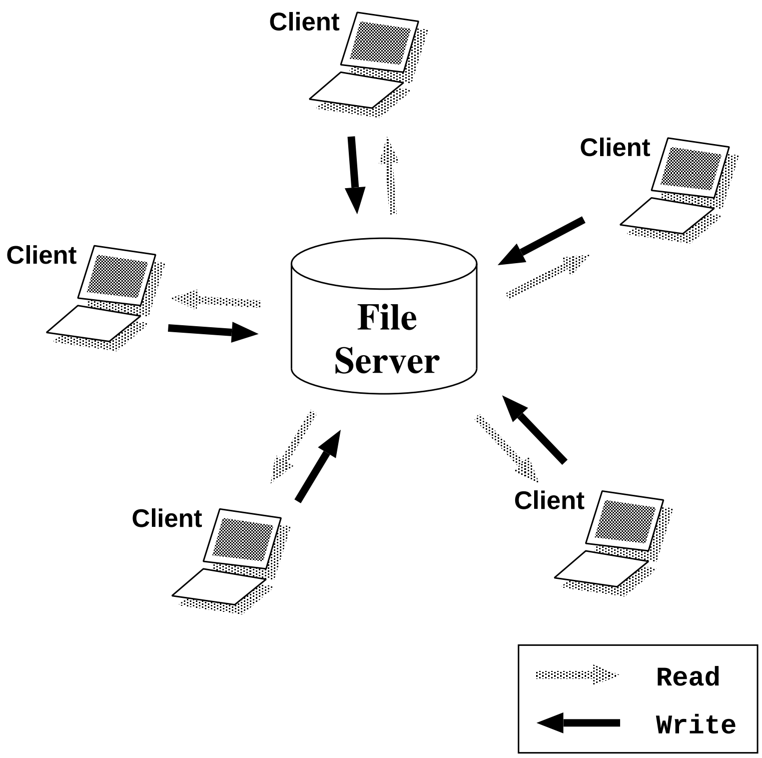

図 2.1: NFS の動作モデル

2.3.2　AFS

AFS(Andrew File System) [2] はカーネギーメロン大学で開発された分散ファイルシステムである。AFS も、NFS と同様にクライアントサーバモデルに基いて設計されている。

AFS の特徴として、大容量のキャッシュを管理する機能が挙げられる。このキャッシュを利用して、図 2.2 で示しているように、サーバとの通信が不可能になってもファイル操作を行うことができる。また、通信回線の状態が多少不安定でも、利用者に回線状況を意識させることを最小限に抑えることができる。

しかし、回線の切断が長時間に及ぶと、クライアント上のキャッシュの整合性が確保できなくなる。このため、最終的にはファイルアクセスが不可能になってしまう。

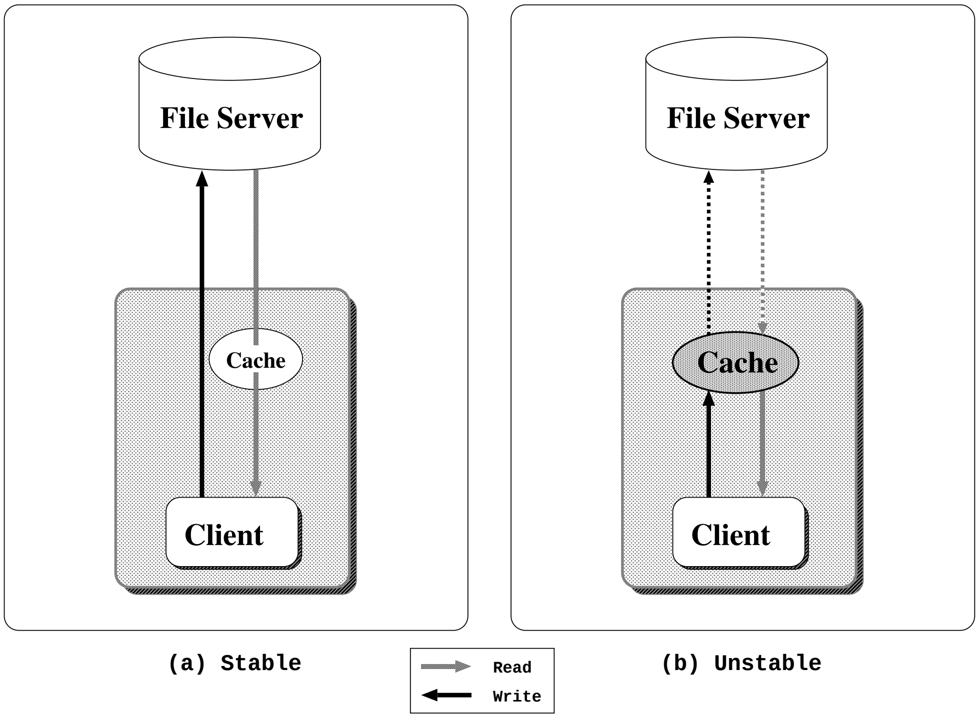

図 2.2: AFS の動作モデル

2.3.3　Coda

Coda [3] はカーネギメロン大学で開発されたファイルシステムである。Coda は移動計算機環

境での使用を前提として設計されており、回線切断時の動作に優れている。Coda は接続状態と切断状態の二つの状態を持つ。通常使用時には接続状態でサーバとの通信を行う (図 2.3 a) が、回線が切断されると切断状態に移行する。切断状態では、ディスクのアクセスを全てキャッシュに対して行う (図 2.3 b)。この機能により、回線の状況を意識せず、利用者は透過的なファイルアクセスを行うことができる。

しかし、Coda ではキャッシュするファイルを予め設定する必要がある。そのため、回線切断後に操作が必要となった、キャッシュを行っていないファイルへのアクセスが不可能である。

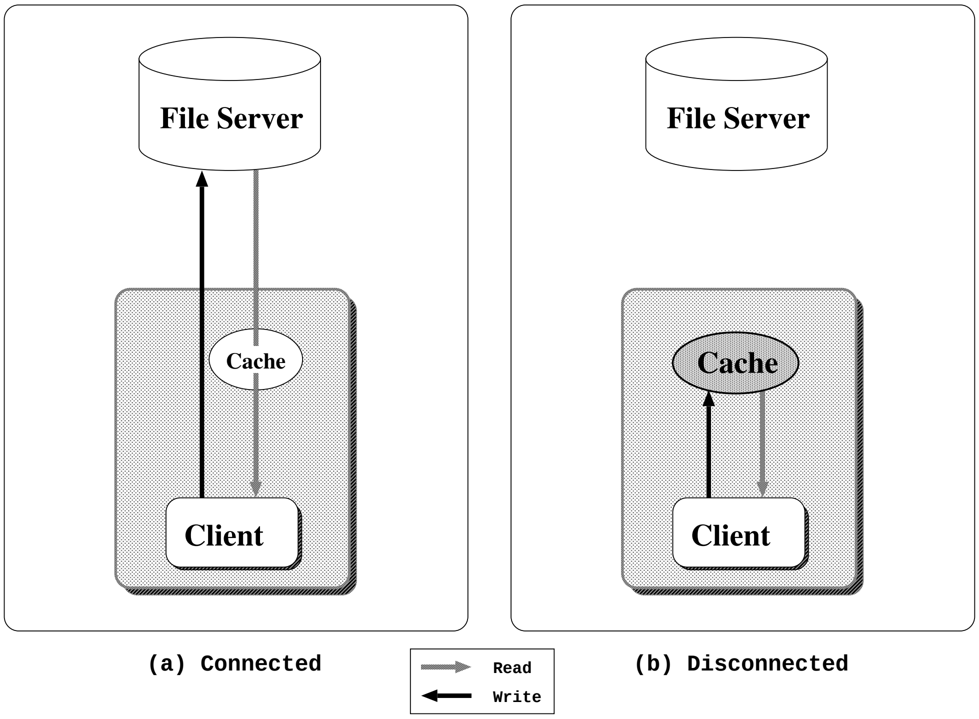

図 2.3: Coda の動作モデル

2.4 既存システムの問題点

前述のように、ファイルの共有を実現したシステムは既に多数存在する。しかし、各システムごとに、その用途は異なる。中でも、移動計算機環境におけるファイルの共有を想定したシステムは少ない。ファイル共有システムの多くは、固定計算機環境での使用を前提として開発されたものである。

通信が不可能な状況での使用を想定したシステムでも、その多くは、ファイルそのものや、その整合性を管理するサーバの存在する、集中型のシステムである。移動する計算機は、ファイルの整合性を確保する為に、予め設定されたサーバに接続しなければならない。

サーバ上にファイルが存在するシステムでは、クライアントがファイルの参照時や更新時、サーバと直接通信し、操作を行う。このようなシステムでは、操作を直接行うというその性質上、回線切断中のファイル操作を実現するものは少ない。回線が切断されると、再び通信可能な状態になるまで、利用者はサーバに存在するファイルに関して、全く操作ができない。

2.5　まとめ

切断時の作業環境を提供するシステムには、キャッシュを用いたものが多い。キャッシュを用いたシステムは、使用目的によっては非常に有効なものである。しかし、キャッシュ技術を使ったシステムはでは、一度操作するファイルを読み込む必要がある。移動計算機環境では、移動中・回線切断中に作業を行う機会が頻繁に存在する。この際、操作するファイルは移動前に操作したものであるとは限らない。そこで、全てのファイルを回線切断時に操作できる環境を提供する必要がある。

上記の問題を考慮したシステムとして、定期的に同期を取ってファイルの共有を行うシステムもある。しかし、現存のシステムは自ホストと相手ホストが一対一で同期を取るシステム、または、単体のサーバに対し、複数のクライアント群がサービスを享受するものである。前節で述べたように、サーバへの依存度が高いシステムでは、サーバの障害がクライアントに大きな影響を与える。移動計算機環境では、不安定な通信環境からサーバとの通信が不可能になる状況が頻繁に起こり得る。そのため、サーバに依存した現存のモデルを改善する必要がある。

第3章　移動体を考慮した情報共有モデル

本研究では、情報共有システムの設計と実装を行う。システムを考案する際、その利用目的によって様々なモデルが考えられる。本章では、本研究の主眼である移動体に適した情報共有システムを考慮し、そのための最適なモデルを考察する。

ここでは、情報共有のモデルを大きく二つに大別する。すなわち、サーバ集中型、分散型である。この二つのモデルの長所・短所に注目し、本研究で提案するシステム構築のために必要な機能を挙げる。その結果、どちらのモデルが移動体における情報共有に関し、より向いているかを考察する。

3.1　サーバ集中型の情報共有モデル

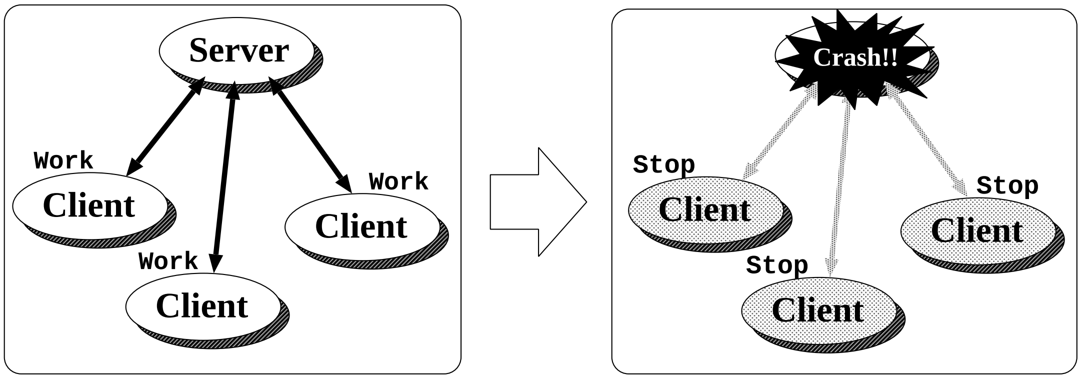

図 3.1: サーバ集中型モデルの危険性

サーバ集中型のシステムでは、ファイルをサーバが管理する。単数ないし複数のクライアントがサーバにアクセスし、必要に応じて、ファイルの更新や取得を行う。

全てのファイルはサーバ上に存在し、サーバ上のファイルに直接アクセスすることで、更新を行う。これにより、ファイルを即時的に更新することができる。情報の整合性についても、容易に確保することができる。このようなシステムは、回線の状況を考慮する必要がなく、複数の利用者が頻繁に更新を行うような状況において有効である。

しかし、図 3.1 に示すように、サーバ自体に致命的な障害が発生すると、クライアントと通信

できなくなる。ファイルは全てサーバが管理している為、こうした状況下では、結果としてクライアントがファイルの取得・更新を行えなくなる危険性がある。

こうしたサーバ上の障害から発生する問題を解決する手段の一つが、バックアップサーバの導入である。サーバのファイルを定期的に別の計算機にバックアップすることで、サーバ上での障害発生時には、バックアップサーバのファイルを利用し、障害の回避・復旧を行うことができる。障害回避の方法として、この手段は有効なものである。しかし、ファイルシステムとは別に、新たなシステムとそれを動作させるための計算機を用意する必要がある。そのため、バックアップサーバの導入は、導入の容易性やコスト面での問題を抱えている。

3.2　分散型の情報共有モデル

分散型のシステムでは、各ホストがファイルを個別に管理する。各ホストで作成・更新されファイルは、他のホストと定期的に同期を取るなどして共有する。この性質上、図 3.2 のように、仮に一つのホストで障害が発生したとしても、障害が発生したホストと同期を取る間隔が延びるに過ぎない。これにより、ホストの障害による、システム全体の致命的な障害を回避することができる。

しかし、あるホストで更新されたファイルの更新内容を反映するためには、通信可能時に同期を取る必要がある。このため、例え自ホストのファイルが更新されたとしても、その時点で他のホストに更新内容が反映されているという保証はない。ファイルの整合性確保という面においては、サーバ集中型のシステムに劣る。

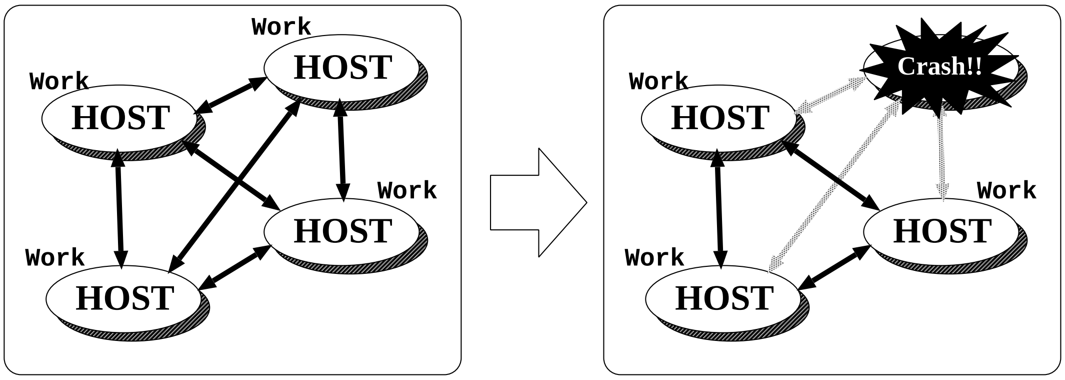

図 3.2: 分散並列型モデルにおける障害回避

3.3　移動体に適したファイル共有モデル

移動計算機環境でのファイル共有において、必要となる機能や特質を以下に挙げる。

回線切断時のファイルの操作環境

移動型計算機を使用する際は、その可搬性から、利用者が計算機を携帯し、移動中に操作を行うことがある。こうした状況での操作を滞りなく行う環境が必要である。

ローカルなネットワークでのファイル共有

利用者が計算機を携帯することがあるため、移動先で必ずしもグローバルなネットワークに接続できるとは限らない。移動先で複数人の利用者が計算機を持ち寄り、赤外線などを使用してローカルなネットワークを構成した際に、そのネットワーク内でのファイル共有を可能にする必要がある。

耐故障性

ファイル共有を行う際に、共有を行うホスト群の一部に障害が発生する場合がある。また、不慮の回線切断によって他の計算機と通信が行えなくなる可能性もある。不安定な移動体通信ではこうした状況が頻繁に起こり得る。このため、障害発生時にシステム全体への被害を最低限に抑える必要がある。

3.4　まとめ

移動計算機環境においては、計算機の移動による回線の切断に起因する、ネットワークの障害が起こる可能性が高い。また、通信状況が不安定になる、もしくは利用者が回線を切断することがある。利用者はこうした状況においても、回線接続時の作業を継続する必要性にられている可能性がある。サーバとの直接通信を行う利点として、即時的なファイル更新やファイルの整合性を確保し易いという点がある。しかし、移動計算機環境を想定した際、そういった利点を多少犠牲にしても、前述のような理由から回線切断時の円滑な操作環境や、耐故障性に優れたシステムを提供することが必要である。

こうした点を考慮すると、移動計算機環境という特殊な状況下では分散型のファイル共有モデ

ルを採用することにより、不安定な通信環境下でファイル共有を行うことができる。そこで本研究では分散型ファイル共有システムを採用し、これを実現するシステムの設計と実装を行う。

第4章　システムの設計

4.1　設計目標

本研究では不安定な通信環境下でのファイル共有を行うために以下に示す目標を定めて設計を行う。

- 複数ホスト間でそれぞれ同様の情報を保持し、共有できるようにする。
  単一のサーバが情報を管理するサーバ集中型のシステムでは、通信路やサーバそのものに障害が発生した際、システムが動作不能になる。こうした障害を回避するため、複数のホストが協調して情報の共有を行うシステムが必要となる。
- 計算機間の通信が不可能な場合も、利用者が回線の状態を意識せずに通常と同様の操作が行なえるようにする。
  計算機が移動する際、無線 LAN や携帯電話の通信状況が悪い状態や、回線を切断した状態では他のホストと通信不可能である。移動計算機環境において、こうした状況は頻繁に発生する。そこで、ファイル操作を円滑に行う為には、計算機間の通信状態に関わらず、同様の作業環境を提供する必要がある。
- あるホストで障害が発生しても、システム全体の動作を妨げないようにする。
  あるホストで障害が発生した際にこの影響がシステム全体に及んでは、ファイルの共有を行うことができない。移動計算機環境という状況では、通信回線の状況から障害発生を起こす可能性が高い。一部のホスト上に障害が発生した場合にもシステム全体は動作し、ファイルの共有を行えるシステムが必要である。
- ファイルの共有を行うホストを自動検知できるようにする。
  ファイル共有を行うホストの自動検知を行うことによって、利用者は設定コストを軽減することができる。

その他、利用者の混乱を避け、利用を容易にするために現存のシステムと同様のインタフェースも必要である。しかし、今回はファイルの共有に着目するため、上記の目標の実現を優先する。

4.2　　システム概要

本研究では、単一のサーバに依存せず、複数のホスト間でファイルの同期を行うシステムを構築する。同期を取ることによって各ホストは同様のファイルを所有する。利用者はローカルディスクのファイルを操作するため、ファイル同期時以外には外部のホストと通信する必要がない。この結果、単一のサーバへのアクセス集中を避けることができる。また、通信路の状況に関わらず、利用者は高速なファイルアクセスを行うことが可能となる。

反面、ファイルの同期は通信可能時に行われるが、利用者がファイルを更新・削除した場合に全てのホスト上のファイルを直接更新するわけではない。利用者の編集したファイルは更新・削除後に他ホストと同期を取られて内容が反映される。このため、即時的なファイル更新が難しく、整合性に関しては保証されない。

本研究では、移動計算機環境という特殊な状況下における情報共有システムの構築を目的とする。そこで本システムでは、定期的なファイル同期機能の実現を優先した設計を行う。

4.2.1　　ファイルの中継

本システムを使用した際のファイル中継動作を、図 4.1 に示す。

ファイル共有を行うホスト群、Host A, Hos B, Host C, Host D が存在する。Host A は Host B, Host C と通信可能である。Host D は、Host A と直接通信することはできないが、Host B との通信が可能である。Host A で作成・削除されたファイルは、Host A が通信可能である HostB, Host C との間で同期を取った時点で Host B, Host C に複製・削除される。HostB も同様に通信可能はホストと同期を取るため、この時点で Host D にファイルが複製・削除される。結果として、Host A のファイル更新は、間接的に Host D の内容に反映される。

4.2.2　　動作モデル

上記の動作を実現するために用意する機能を、以下に示す。

- ファイル同期機能
- グループ管理機能
- ホスト自動検知機能

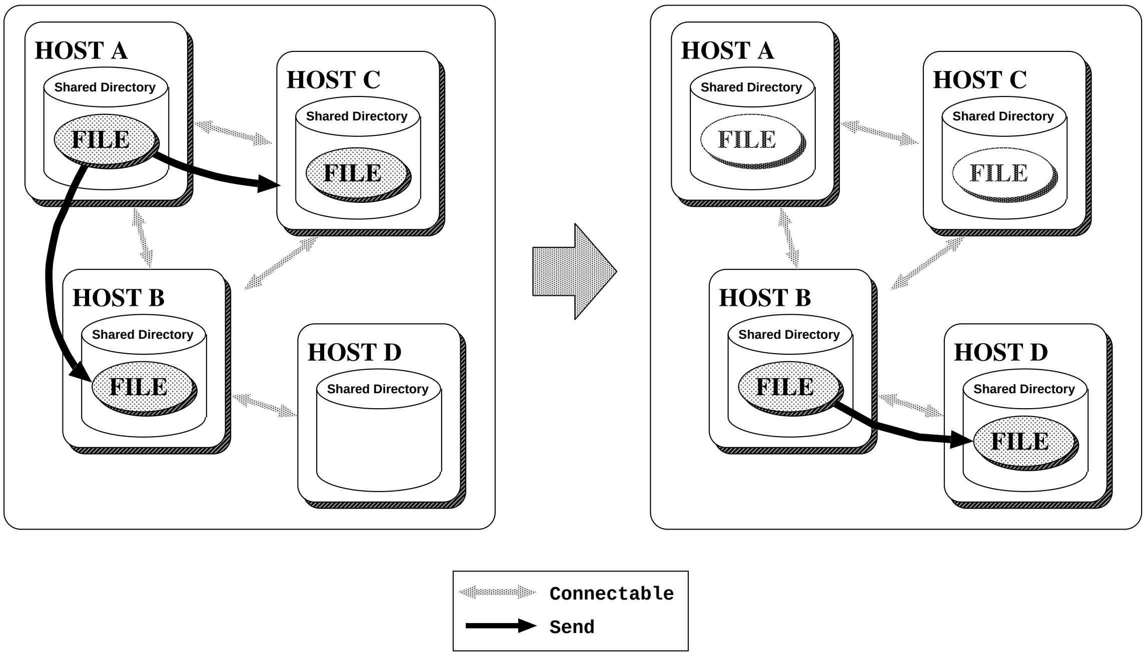

図 4.1: ファイルの中継

ファイル同期機能

他ホストのファイルを複製して共有するための機能として用意する。本機能では、ローカルホストと他ホストとのファイル同期を行う。この動作を図 4.2 に示す。ファイル同期を行い、同一のファイルを各ホストが所有することで、利用者は回線の状態に関わらず、ファイルアクセスを行うこが可能である。

ホスト間のファイル同期部分には NetNews [4, 5] の概念を応用する。NetNews では、各記事をファイルとして所有し、各サーバ間で定期的に通信を行い、同期を取る。クライアントはサーバから必要に応じて記事を取り寄せる。本システムでは、こうした NetNews におけるサーバ間での記事の同期部分を応用し、ファイルの同期を行う。NetNews における記事は、本システム上ではファイルとして扱う。例えばファイルの送信は NetNews における記事の送信、ファイルの受信は記事の講読、ファイルの削除は記事のキャンセルに、それぞれ相当する。

グループ管理機能

ファイル群をグループ化して管理するために用意する。例えばメーリングリストや NetNews では、情報を関連するものでまとめ、管理する。特に NetNews におけるニュースグループは、名

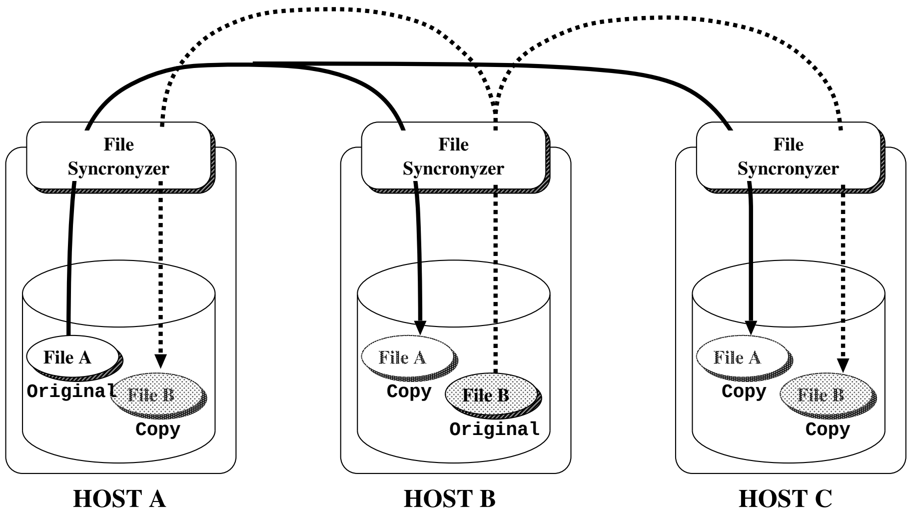

図 4.2: ファイル同期機能の動作モデル

前の通り、記事をそれぞれのジャンルに分けたグループ単位で管理している。利用者は情報取得時にこのグループを指定することで、関連する情報を取得することができる。本機能でも同様に、ファイル群をグループ化する。利用者は設定時に、共有可能なファイル群をグループとして設定する。ファイルの共有が行われるのは、このグループに設定されたファイルのみである。

グループ管理機能のモデル概略図を図 4.3 に示す。各ホストはファイル同期を行う前に、同期を行うファイルの属するグループ名を送信する。通信先では、受け取ったグループ名が自分の設定ファイルに記述されていれは、このグループに属するファイルの同期を行う。設定ファイル内にグループが存在しない場合は、通信を行わない。図 4.3 では、Host A が管理するグループであるGroup B を Host B では管理していないため、両ホスト間で通信を行った際、同期が取られるのは Group A, Group C, Group D に属するファイル群のみである。

ホスト自動検知機能

本研究で想定するシステムの利用形態は、離れた場所から持ち寄った計算機間で、ファイルの共有を行うものである。この際、移動を行う度に通信可能ホストを手動で再設定するのは、利用者にとって管理コストの面で大きな負担となる。この問題を解決するため、本システムでは、導入するホストの自動検知を行う。

本機能の動作モデルを図 4.4 に示す。このホスト自動検知機能はグループ管理システムと協調して動作する。

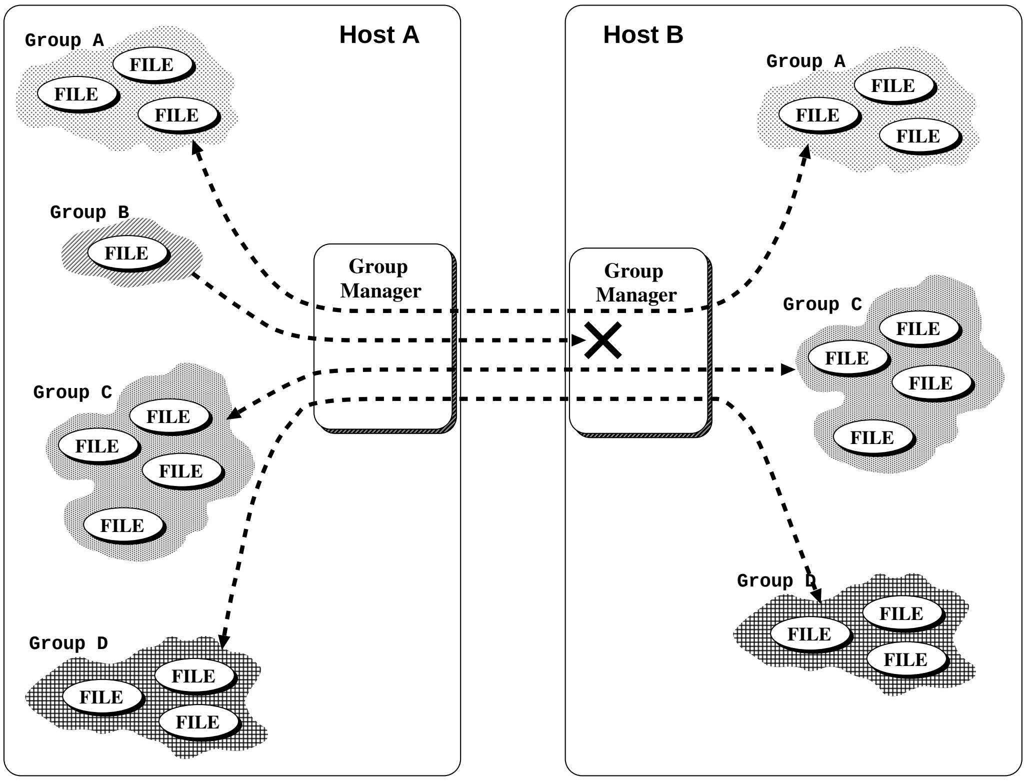

図 4.3: グループ管理機能の動作モデル

4.2.3　　システムの基本動作

あるホストで設定された共有ディレクトリにファイルが作成もしくは削除されると、ホスト上のデーモンがそれを検知し、ファイルリストに蓄積する。設定ファイルに記述されたホストと通信を行うことが可能であれば、各ホストは更新ファイルリストを他のホストと照合する。この際、自ホストが所持せず、削除ファイルとして登録されていないファイルを他のホストから取得する。結果、各ホストはそれぞれ、同じファイルをローカルディスク上に持つ。

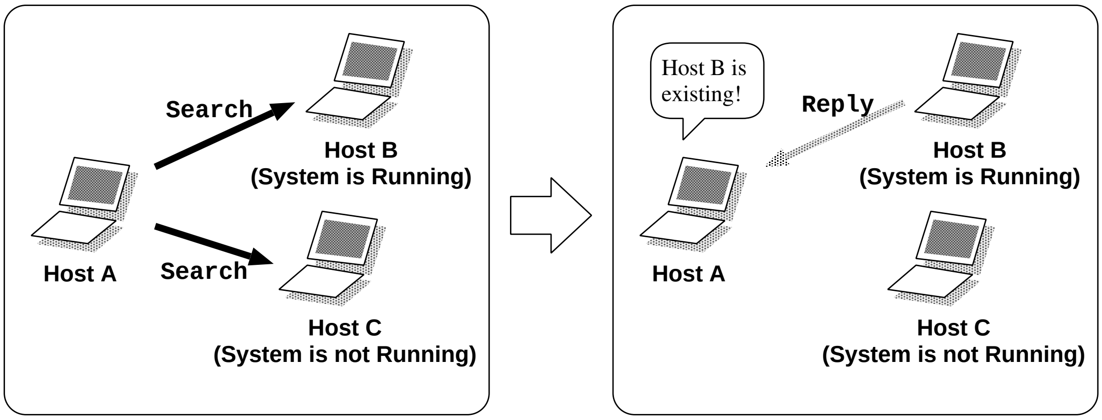

図 4.4: ホスト自動検知機能の動作モデル

4.3　システム構成

本システムはその機能ごとに、以下の構成要素に大別される。

ファイル監視デーモン

共有ディレクトリを定期的に監視し、現存ファイルおよび削除ファイルを抽出・管理する。

グループ管理システム

同期を取るファイルの属するグループが、通信先ホストの設定に登録されていることを確認する。

ファイル同期デーモン

通信先ホストとのファイル同期を行う。

ホスト検知デーモン

本システムを導入しているホスト群を検知する。

ファイル監視デーモンとファイル同期デーモンに注目した処理の主な流れを、図 4.5 に示す。

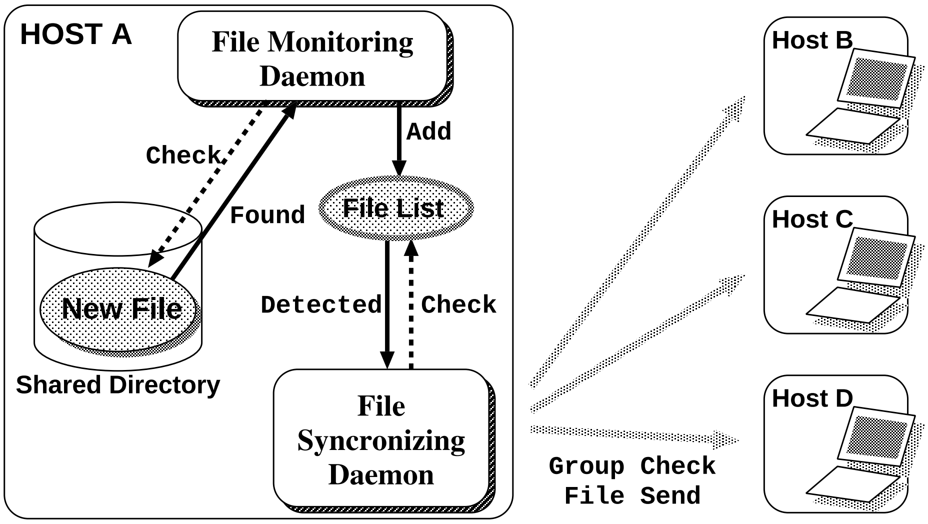

図 4.5: システム構成図

4.3.1 処理の流れ

ファイル監視デーモンの動作

図 4.6 中の (1) ~ (3) が、ファイル監視デーモンの動作である。以下にその処理を示す。

1. 定期的に共有ディレクトリ以下のファイル全てにおいて、ファイル名、最終更新日時を調べる。
2. 前回に共有ディレクトリを調べた結果と今回の結果を比較し、共有ディレクトリ内に存在するファイルを、現存ファイルとして扱う。前回に作成されたファイルリストから削除されているファイルは削除ファイルとして扱う。
3. 発見した現存ファイルおよび削除ファイルをファイルリストに追加する。同時に、前回のディレクトリ検索結果を、新しく得られた結果で上書きする。

ファイル監視デーモンは、この動作を一定時間毎に繰り返す。

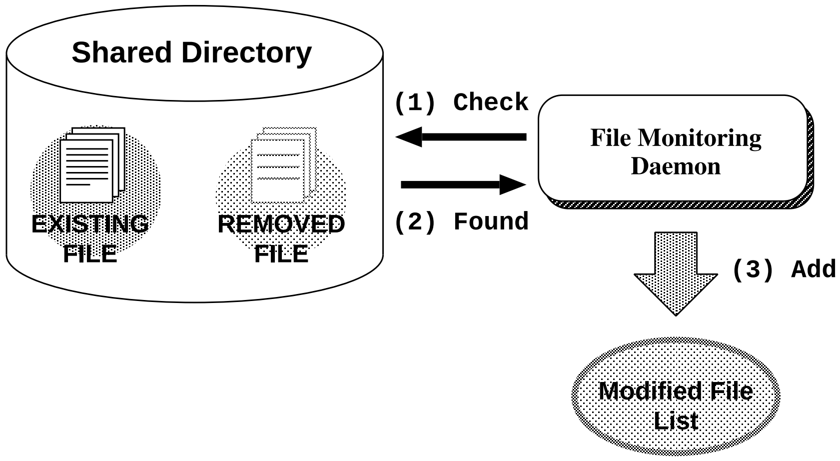

図 4.6: ファイル監視デーモンの動作

ファイル同期デーモンの動作

図 4.7 中の (1) ~ (4) が、ファイル同期デーモンの動作である。以下にその処理を示す。

1. 定期的にファイル更新リストをチェックする。
2. 現存するファイル、および削除ファイルを検知する。ファイルリストが更新されていない時は処理を中断し、一定時間後に (1) の処理を繰り返す。

3. 設定ファイルに記載されたホストと接続を行う。接続後、グループ管理システムは、接続先のホストがこれから送信するファイルのグループに所属しているかどうかを調べる。グループ管理システムの動作については、4.3.2 で解説する。
4. 接続後、ファイルの同期を行う。

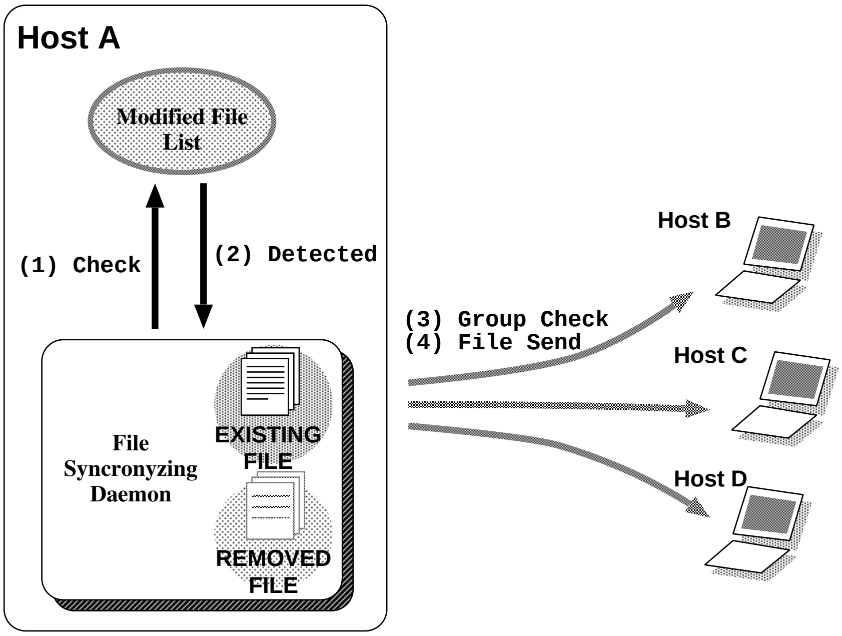

図 4.7: ファイル同期デーモンの動作

グループ管理システムの動作

グループ管理システムはファイル同期デーモンの中で、他ホストとの接続の際に機能する。各ホスト上で利用者の設定したグループを他ホストと比較し、互いのホストに存在するグループに属するファイルの同期を許可する。詳しい処理については、4.3.2 で述べる。

ホスト自動検知デーモンの動作

ホスト自動検知デーモンの動作を図 4.8 に示す。システムホスト自動検知システムは、ファイルの同期を行うホスト群を検知する際に動作する。あるホストが、通信可能なホスト群にブロードキャストメッセージの配信を行う。本システムを導入しているホストは、このメッセージを受信すると、配信元に制御メッセージを返信する。同報メッセージの配信元は返信されたメッセー

ジを受け取ると、そのホストをファイル共有ホストとして登録する。図 4.8 では、Host B, Host E, Host F が、ファイル共有の可能なホストとして Host A に認識される。

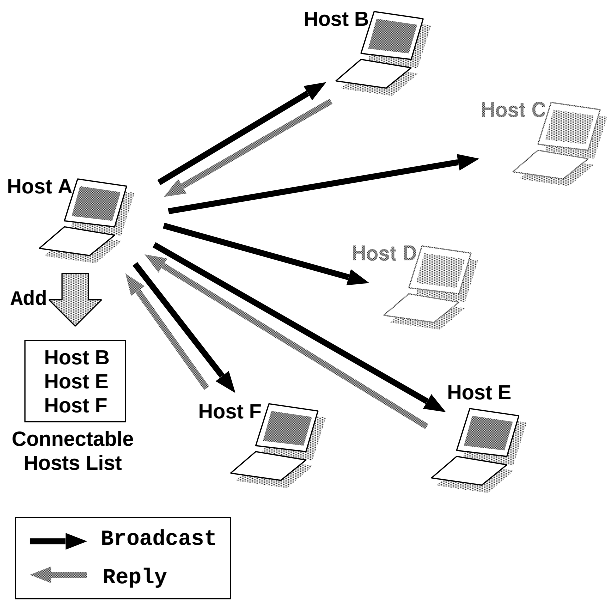

図 4.8: ホスト自動検知デーモンの動作

4.3.2 通信プロトコル

本システムではホスト間の通信を行う際、以下のプロトコルに従って通信を行う。

メッセージフォーマット

本システムで使用するメッセージフォーマットは、図 4.9 の通りである。本システムでは全てのプロトコルに共通して、このフォーマットを使用する。図からもわかるように、メッセージは、メッセージ情報を含む可変長パケットとして送受信される。通信は TCP [6] によって行われる。本メッセージフォーマットの詳細を表 4.1 に示す。

ファイル同期プロトコル

ファイルの同期を行う際に使用する。本プロトコルで使用する制御メッセージの一覧を表 4.2 に示す。

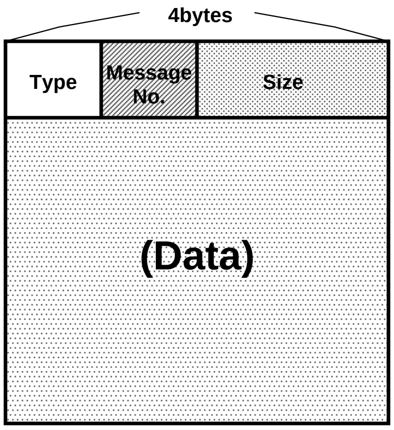

図 4.9: メッセージフォーマット

表 4.1: メッセージフォーマット一覧

種類	内容
Type	メッセージの種類 (1: File, 2: Message) を記述する。
Message No.	Type が 2 の場合、制御メッセージが記述される。プログラム内では、各メッセージごとに、個別の数値として識別される。
Size	パケットの総サイズを記述する。
Data	Type が 1 の時は送信するファイルが、2 の時はメッセージで使用する情報が送信される。メッセージの種類によっては、Data 部を含まないものもある。

表 4.2 に記述したメッセージのやり取りを中心に、ファイル同期プロトコルの動作を図 4.10 に示す。

1. IHAVE メッセージに続いて、Host A で作成された現存・削除ファイルのリストを Host B に送信する。
2. Host B は IHAVE メッセージに続くファイルリストを受信し、自ホストのファイルリストと比較する。この際、自ホストに存在しないファイルのリストを、SENDME メッセージに続いて送信する。
3. Host B は、自ホストの削除ファイルリストに記述されておらず、Host A からの削除ファイ

表 4.2: ファイル同期プロトコルのメッセージ

メッセージ	役割
IHAVE	現存するファイル群、および過去に削除されたファイル群を通知する。
SENDME	ファイルの送信を他ホストに要求する。
READY	ファイル受信か可能であることを通知する。
FILENAME	ファイル名とパスを通知する。
FILESIZE	ファイルのサイズを通知する。
SEND	ファイルの送信を開始する。
COMPLETE	ファイル送信の完了を通知する。

ルリストに登録されたファイルを削除する。削除されたファイルは、自ホストの削除ファイルリストに追加する。

4. Host A は、SENDME メッセージのファイルリストに記述されたファイルを、Host B に送信する。
5. 送信終了後、Host B に COMPLETE メッセージを送信する。送信ファイルが複数存在する時は、(4) に戻り、全てのファイルを送信するまで、動作を繰り返す。

グループ管理プロトコル

ホスト間での通信開始時に使用する。通信元のホストは、これから送信するファイルの属するグループをメッセージとして送信する。通信先のホストはこのメッセージを受信し、自ホストの設定にこのグループが存在するかどうかを確認する。これを確認した時点でファイル同期プロトコルの動作に移り、ファイルの同期を行う。同期を行う際に送受信するファイルは、設定されたグループに属するものに限られる。

本プロトコルで使用するメッセージの一覧を表 4.3 に示す。

本プロトコルは、以下の流れに沿って動作する。処理の流れを図 4.11 に示す。

1. Host A が Host B との通信を開設し、HELLO を送信する。
2. Host B が Host A に HELLO を送信する。
3. Host A は GROUP, HOST を送信する。
4. Host B は送信された GROUP, HOST が自分の持つ設定ファイル内に記載されているか確認する。
5. 設定ファイルに記載されている場合、Host B は Host A に PERMIT を送る。そうでない場

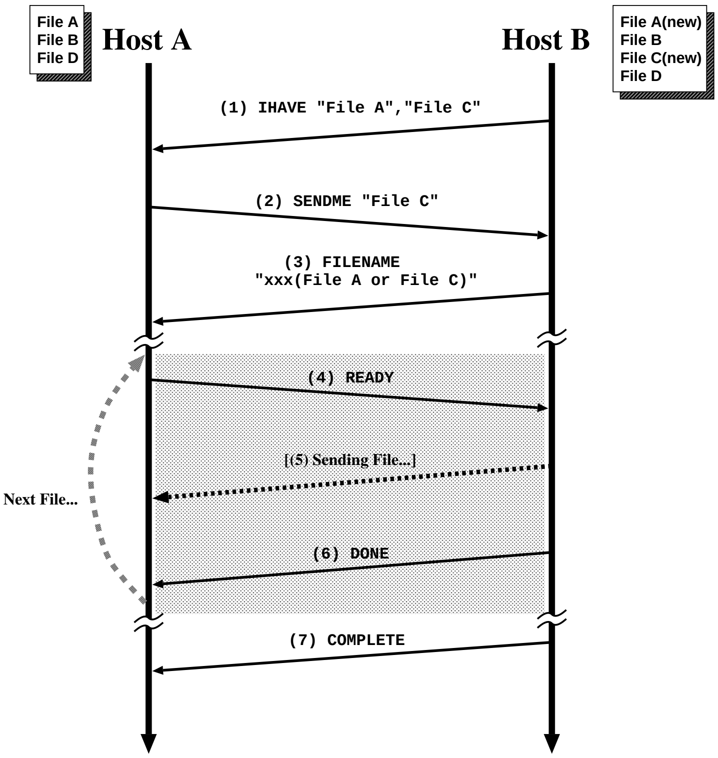

図 4.10: ファイル同期プロトコルの動作

合は FORBID を送る。

6. Host A は PERMIT メッセージを確認し、ファイル同期プロトコルの動作に移る。FORBID を送信された場合は、通信を終了する。
7. ファイル同期プロトコルの動作が終了し、Host A が Host B に COMPLETE を送信する。
8. Host B は COMPLETE を確認し、BYE を送信して通信を切断する。

1 から 8 の動作を、設定ファイルに記述された全ホストに対して行う。

ホスト自動検知プロトコル

本プロトコルが使用するメッセージの一覧を表 4.4 に示す。

本プロトコルがホスト検知を行う手順を、以下に示す。この処理の流れを、図 4.12 に示す。

1. 通信を行おうとするホストがネットワーク全体にブロードキャストを行う。
2. 本システムを導入しているホストはブロードキャストされたメッセージを受信し、送信元に自ホストの存在を通知する。

表 4.3: グループ管理プロトコルのメッセージ

メッセージ	役割
HELLO	通信を開始する。
GROUP	自ホストの所持するグループ群を通知する。
HOST	自分のホスト名を通知する。
PERMIT	通信を許可する。
FORBID	通信を禁止する。
BYE	通信を終了する。

表 4.4: ホスト自動検知プロトコルのメッセージ

メッセージ	役割
WHEREAREYOU	本システムを導入しているホスト群に、各自ホスト存在の返信を要求する。
IAMHERE	自ホスト名、および自ホストの所持するグループ群を通知する。

3. グループ管理システムと連動。
   (a) 送信元のホストは自分のホスト名と、ホストで管理するグループ名を各ホストに送信する。
   (b) 各ホストはグループ名とホスト名を送信元から受け取り、自分の存在するグループであるか、また、グループに所属するホストであるかを調べる。
   (c) 同一グループであることが確認された時点でファイルの同期を行う。
4. 3 の動作を、存在を通知してきた全てのホストとの通信が終了するまで繰り返す。

4.4　設計上の注意点

本項では、設計を行った時点での注意点および問題点を述べる。

4.4.1　ファイルの整合性

ファイルの同期を行う際、複数ホストで同一名のファイルが作成される可能性がある。こうした状況でホスト間のファイルを同期する場合、それらを別ファイルとして管理するべきか、それ

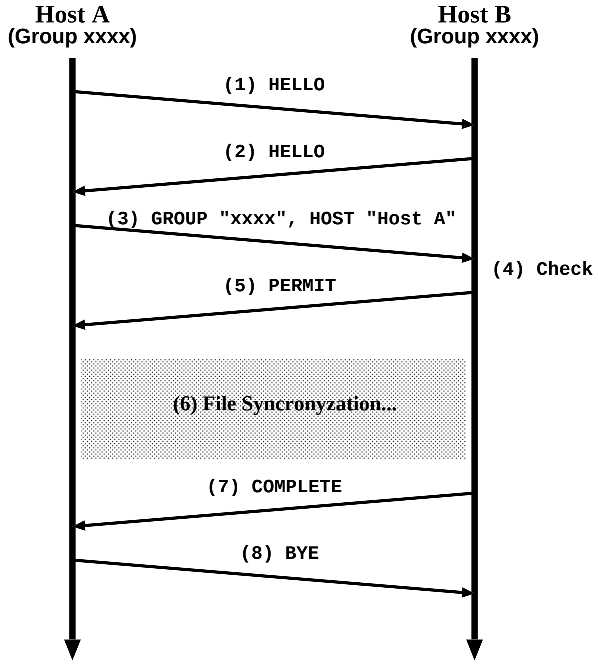

図 4.11: グループ管理プロトコルの動作

とも上書きするべきかという問題点が挙げられる。

本システムでは、重複したファイルはそれぞれ別のファイル名で管理する。具体的には、重複したファイルが存在した時、ファイル名の末尾にファイル操作の行われたホスト名とシリアル番号をつけ、ファイル名の重複を避ける。このシリアル番号は、元ファイルの通し番号と、UNIX の **gettimeofday**[1] システムコールから得られた時間を組み合わせたものとする。

4.4.2 ファイルのスレッド管理

同一ファイルが複数のホストで作成・更新された場合に、NetNews の概念を応用し、各ファイルを記事のような扱いにする。NetNews における最新スレッドを最新ファイルとして扱い、これを通常のファイルと同様に利用者に見せる。その為には、NetNews のメッセージ ID のように、各ファイルがどのスレッドに属するのかを記述する必要がある。

[1] UNIX のシステムコール。1970 年 1 月 1 日午前 0 時 0 分 0 秒 (世界標準時) から現在までに経過した時間を、マイクロ秒単位で知ることができる。

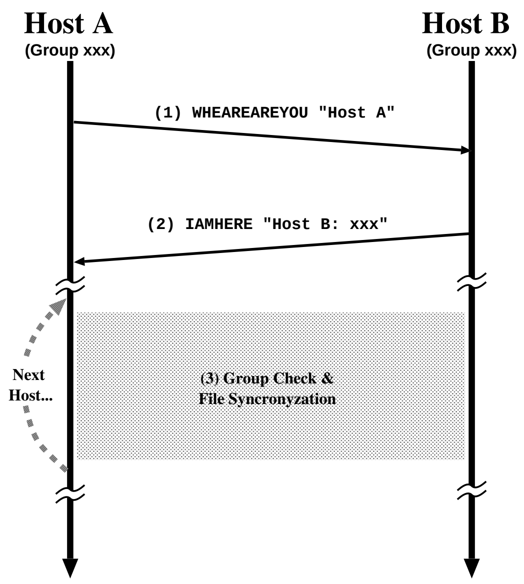

図 4.12: ホスト自動検知プロトコルの動作

4.4.3 ファイルシステムとしてのインタフェース

今回は、ファイル共有の部分に注目し、ファイルシステムとしてのインタフェースは実現しない。しかし、本システム最終的な形としては、xxfs [7] のようなシステムと併用することで、ファイルシステムとしてのインタフェースを提供する。その際、ファイルをスレッドとして管理するための情報として、メッセージ ID の代替となるものが必要となる。本システムでは、ファイル名の末尾に付加するシリアル番号をこのメッセージ ID の代替として使用する。

第 5 章　実装

5.1　実装目標

本システムを実装するにあたって、以下の目標を設定した。

- 一般的なオペレーティングシステムで動くようにする。
  本システムはファイルの共有を行うものである。共有を行うためには、多くの計算機上でシステムを稼働させせなければならない。一般的なオペレーティングシステムでの動作を可能にすることで、より多くの計算機上で、本システムが使用可能になる。
- システムの導入を容易にする。
  システムの導入が困難なものであると、導入コストがかかる。結果として、利用者は導入の困難なシステムを避けてしまう。本システムをより多くの計算機で稼働させるには、利用者がシステムの導入を容易に行えるようにする必要がある。

5.2　実装概要

本システムの実装では、一般的なオペレーティングシステムとして、UNIX オペレーティングシステムの一つである Linux 2.0.36(Slackware 3.1) を選択した。言語には、C++を使用した。

また、システムの導入を容易にするため、システム全体をオペレーティングシステムのユーザ空間におけるアプリケーションプログラムとして実装した。

5.3　ファイル監視機能の実装

ファイル監視機能を構成するモジュールと、その役割を以下に示す。モジュール構成図を図 5.1 に、状態遷移図を図 5.2 にそれぞれ示す。

設定ファイル解析モジュール

　設定ファイルを読み取り、その内容を各モジュールに引き渡す。

ディレクトリ検索モジュール

　設定されたディレクトリ以下を全て検索する。ファイル名の取得と、ディレクトリかどうか

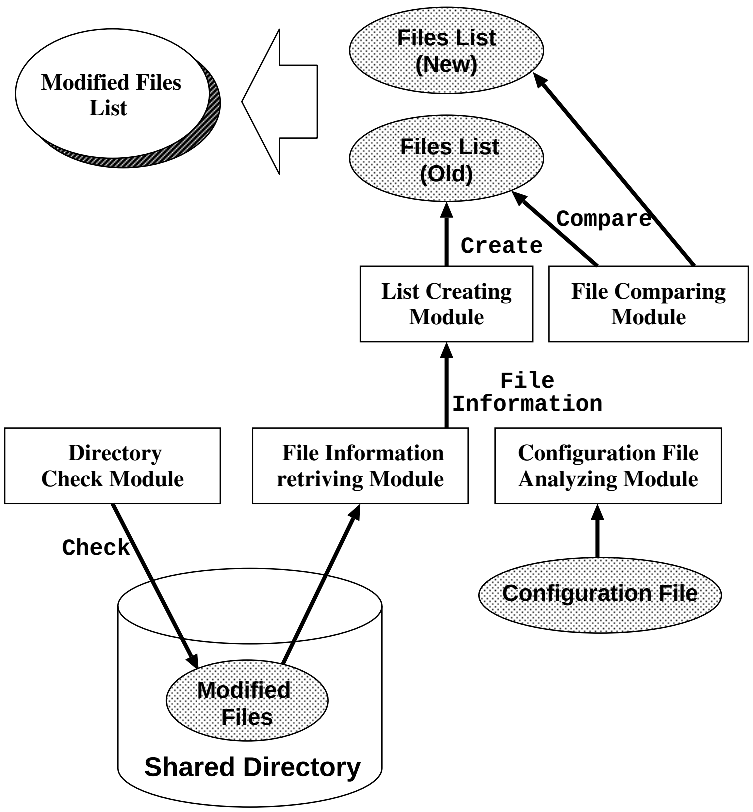

図 5.1: ファイル監視機能のモジュール構成

の判別を行う。ディレクトリを発見した場合は、その中身を検索し、最下層のディレクトリに到達するまで検索を続ける。

ファイル情報取得モジュール

最終更新日時やファイルサイズなど、指定したファイルの情報を取得する。ファイル情報の取得には、**stat**[1] システムコールを使用する。

リスト作成モジュール

ディレクトリ検索モジュール、ファイル情報取得モジュールから得た情報を統合する。統合した情報はファイルリストとして保存する。

ファイル比較モジュール

リスト作成モジュールの作成したファイルリストを前回作成時のものと比較する。前回作成時のファイルリストには存在しないファイルを新規作成ファイルして扱う。タイムスタンプの更新されたファイルは更新ファイルとして扱う。前回のファイルリストに存在し、比較時

[1] UNIX のシステムコール。ファイルのサイズや属性、タイムスタンプなどの情報を得ることができる。

に実際のディレクトリに存在しないファイルは、削除ファイルとして扱う。

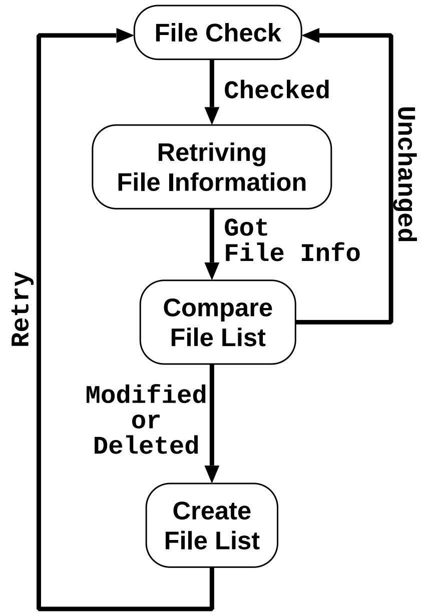

図 5.2: ファイル監視機能の状態遷移図

5.4　ファイル同期機能の実装

ファイル同期機能を構成するモジュールと、その役割を以下に示す。本機能のモジュール構成図を図 5.3 に、状態遷移図を図 5.4 にそれぞれ示す。

設定ファイル解析モジュール

　設定ファイルを読み取り、その内容を各モジュールに引き渡す。

ファイル送信モジュール

　接続先のホストに、ファイルを送信する。

ファイル受信モジュール

　接続先のホストから、ファイルを受信する。

メッセージ解析モジュール

　接続先のホストが送信して来たメッセージを解析し、各メッセージに対応する関数を呼び出す。

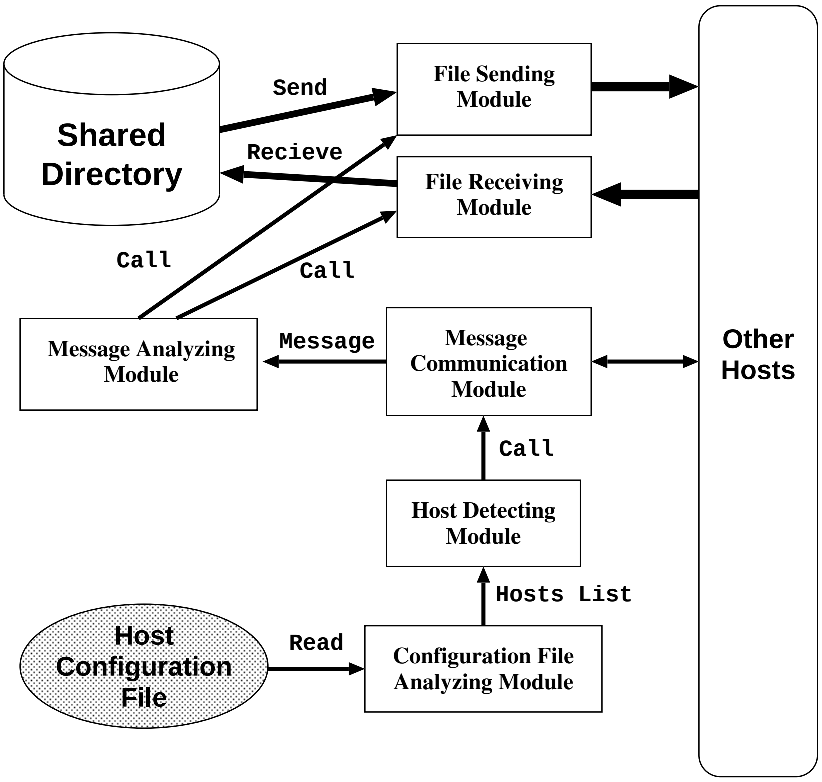

図 5.3: ファイル同期機能のモジュール構成

メッセージ通信モジュール

接続先のホストと、実際に制御メッセージの送受信を行う。

通信ホスト検知モジュール

設定ファイルから、定期的に通信を行うホスト群のリストを読み込む。

リスト比較モジュール

接続先のホストから送信された IHAVE メッセージ以下に続く更新ファイルのリストを、自分のホストのファイルリストと比較する。

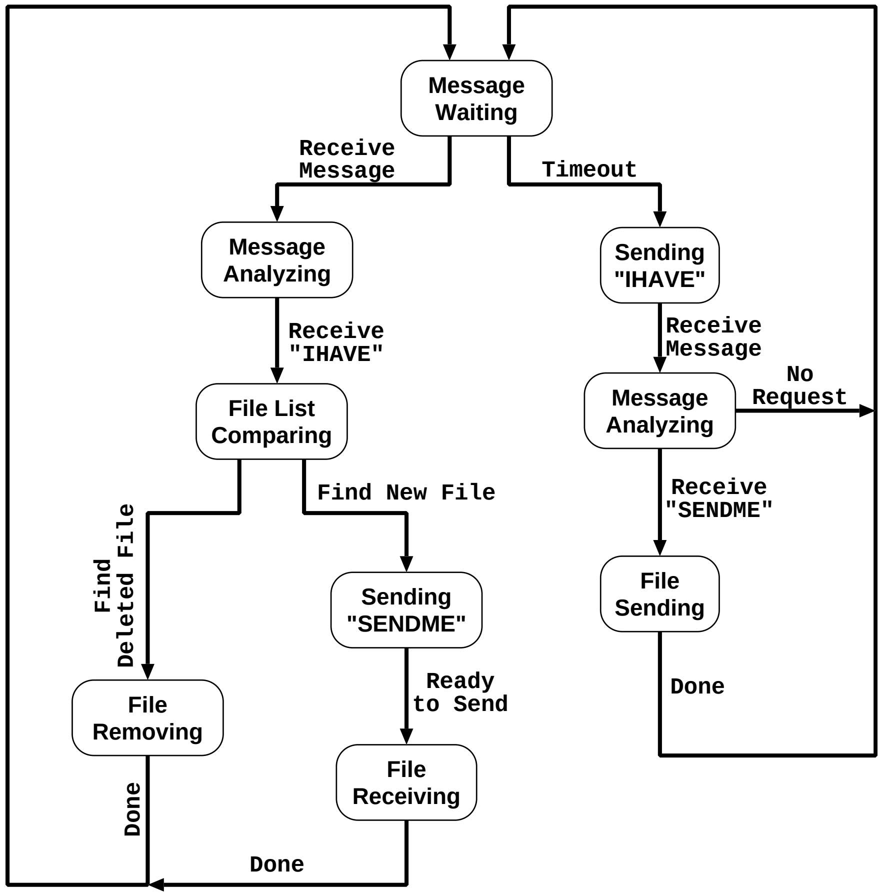

図 5.4: ファイル同期機能の状態遷移図

5.5 グループ管理機能の実装

グループ管理機能を構成するモジュールと、その役割を以下に示す。本機能のモジュール構成図を図 5.5 に、状態遷移図を図 5.6 にそれぞれ示す。

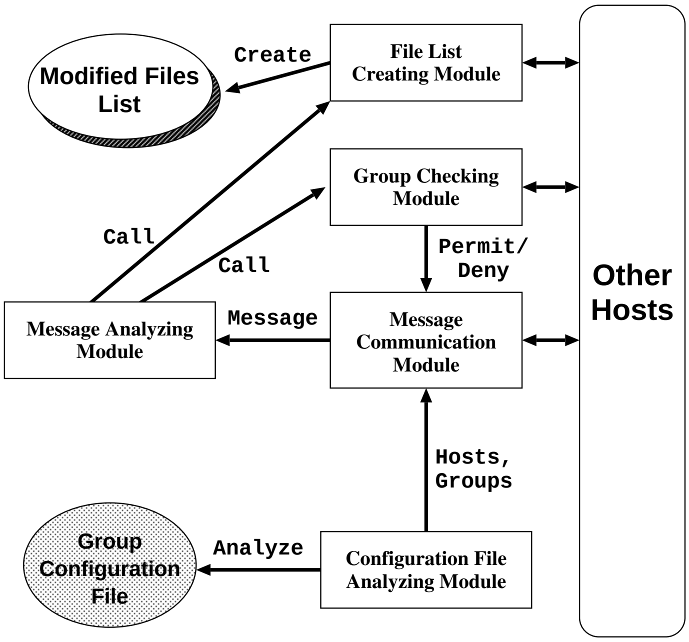

図 5.5: グループ管理機能のモジュール構成

設定ファイル解析モジュール

　設定ファイルを読み取り、その内容を各モジュールに引き渡す。

メッセージ解析モジュール

　制御メッセージの解析を行い、各メッセージに対応する関数を呼び出す。

メッセージ通信モジュール

　制御メッセージの送受信を、接続先のホストと行う。

グループ判別モジュール

　接続先のホストが、要求するファイルグループを所有しているかどうかを判別する。

ファイル抽出モジュール

　更新ファイルのリストを参照し、その中から、指定したグループに属するファイルを抽出する。抽出したファイルは、新たなリストとして保存する。

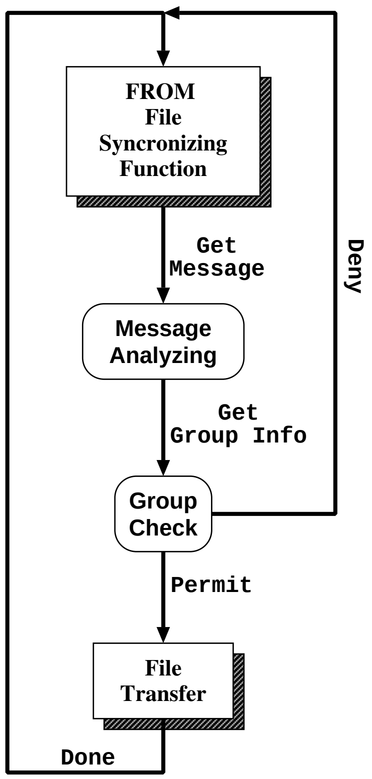

図 5.6: グループ管理機能の状態遷移図

5.6 ホスト自動検知機能の実装

ホスト自動検知を構成するモジュールと、その役割を以下に示す。本機能のモジュール構成を図 5.7 に、状態遷移図を図 5.8 にそれぞれ示す。

ブロードキャストメッセージ配信モジュール

通信可能な各ホストに、制御メッセージを送信する。

メッセージ通信モジュール

同報メッセージ受信側ホストで動作する。同報メッセージを受信した際、自ホストが本システムを導入していることを、同報メッセージの配信元に通知する。

ホスト管理モジュール

同報メッセージ配信側ホストで動作する。メッセージの帰って来た返ホストを本システムの稼働するホストと認識し、管理を行う。

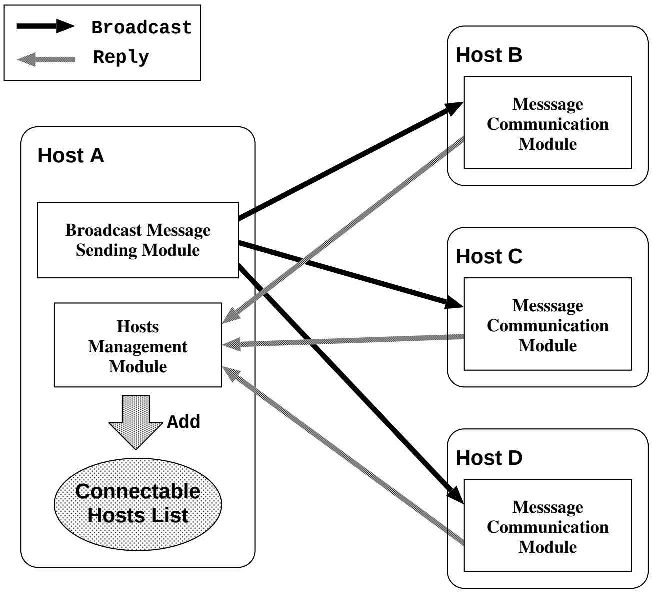

図 5.7: ホスト自動検知機能のモジュール構成

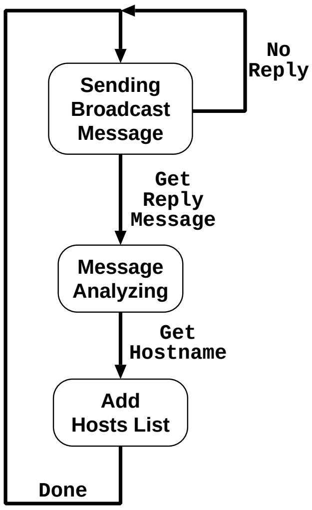

図 5.8: ホスト自動検知機能の状態遷移図

第6章　評価

本章では、本論文で設計・実装したシステムの評価を行う。評価の結果を他のファイル共有システムと比較し、移動計算機環境において、本システムの優れている点と、今後改善すべき点を考察する。

6.1　評価項目

6.1.1　移動時の作業継続性に関する評価

移動計算機環境では、移動する際に回線を切断して移動する状況や、通信中に回線の状況が不安定になる状況が頻繁に起こる。そこで、回線切断時にファイル操作を行い、移動時に円滑な作業継続を行なうことができることを確認する。

実験環境

本実験はプライベートアドレスで構築した、外部との接続性を持たないネットワーク環境で行った。

回線の切断や再接続を意図的に行うため、ネットワークへの接続にはイーサネットケーブル (10Base-T) を使用し、最大 10 メガビット/秒相当の通信速度で実験を行った。

実験で使用した計算機を、表 6.1 に示す。そのホスト名と、割り当てたアドレスは以下の通りである。

- 実験ホスト **A**
  - ホスト名: host01
  - IP アドレス: 192.168.0.1
- 実験ホスト **B**
  - ホスト名: host02
  - IP アドレス: 192.168.0.2
- 実験ホスト **C**
  - ホスト名: host03

  - IP アドレス: 192.168.0.3
- 実験ホスト **D**
  - ホスト名: host04
  - IP アドレス: 192.168.0.4

表 6.1: 実験に使用した計算機 (1)

使用した計算機	使用 OS	CPU クロック	メモリ	台数
DynaBook PORTEGE 660CDT	Linux 2.0.36(Slackware 3.1)	150MHz	32MB	1
Let's Note AL-N2	FreeBSD-2.2.7	166MHz	96MB	2
Let's Note AL-N2	FreeBSD-2.2.8	166MHz	96MB	1

評価内容

1. 回線切断

移動時の動作を確認するため、回線を意図的に切断する。

2. 作業

移動中の作業を想定するため、回線の切断された状態で作業を行う。

3. 回線接続

回線を再接続し、他ホストとファイルの同期を取る。

結果

回線切断時に、ファイルの新規作成、編集、削除を行い、再接続を行った。その結果として、いずれに関してもファイル同期後に、他のホストで更新内容が反映されていることを確認できた。

考察

出力されたログを A.1[1] に示す。そのうち一部を、図 6.1 に示す。

本実験では、ホスト検知や通信状態のチェックを 15 秒おきに行った。ログを参照すると、回線チェック時、次回のチェック時までにメッセージの返って来ないホストを、通信可能ホストのリストから削除していることがわかる。ここで削除されたホストは回線復旧時に、ブロードキャス

[1] 巻末 付録

```
Jan 24 20:14:01 host01 p2sync: REMOVED 192.168.0.4
Jan 24 20:14:01 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:16 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:31 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:46 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:46 host01 p2sync: ADDHOST 192.168.0.2
Jan 24 20:14:46 host01 p2sync: ADDHOST 192.168.0.4
Jan 24 20:14:46 host01 p2sync: ADDHOST 192.168.0.3

                        — 中略 —

Jan 24 20:15:03 host01 p2sync: SENDTO 192.168.0.4 2 FILE(s)
Jan 24 20:15:05 host01 p2sync: Sending Done.
Jan 24 20:15:05 host01 p2sync: SENDTO 192.168.0.3 2 FILE(s)
Jan 24 20:15:06 host01 p2sync: Sending Done.
Jan 24 20:15:06 host01 p2sync: RECVFROM 192.168.0.4 5 FILE(s)
Jan 24 20:15:08 host01 p2sync: Receiving Done.
```

図 6.1: 移動時の作業継続性に関する評価のログ

トされたメッセージを受信して送信元にメッセージを返すことで、再び通信可能ホストのリストに追加される。

回線のチェック時にはファイルリストの比較も行われる。ログ (図 6.1) を参照すると、回線を再接続した際にファイルリストの比較を行い、この時点でファイルの送受信を行っていることがわかる。ログの時間から、回線再接続後、15 秒以内にファイルの同期が取られることが確認できる。

6.1.2　耐故障性に関する評価

サーバに障害が発生した際、又は不測の事態によって回線が切断された状況が想定できる。特に後者のような状況は、携帯電話や無線 LAN を使用して通信を行う機会の多い移動計算機環境において、頻繁に起こり得る。そこで、利用者が作業を行っている最中に回線を切断し、その際に作業継続が行えることを確認する。

実験環境

本実験は、前実験と同様の環境 (表 6.1) で行った。

評価内容

1. 複数ホストでの動作
4 台のホストでシステムを構成する。

2. 通信先ホスト障害の発生
システム動作中のホスト 1 台で、意図的に障害を発生させる。ここでは、強制的にホストをシャットダウンする。

3. 動作の確認
シャットダウンしたホスト以外のホスト群において、ファイル操作・共有が行えることを確認する。

4. 障害発生ホストの再起動
シャットダウンしたホストを再起動し、ファイルの同期を行う。障害発生中に他ホストで行われたファイルの更新内容が、ファイル同期後に反映されていることを確認する。

結果

ファイル共有ホスト上での障害発生時にも通常の動作時と同様に、ファイルの更新・同期が行われることを確認した。また、他のホストで行われたファイルの更新が、障害復旧後、障害発生ホスト上に反映されていることを確認した。

考察

出力されたログを A.2[2] に示す。ログの一部を図 6.2 に示す。

回線切断実験の時と同様に、回線のチェックは 15 秒おきに設定した。ログを参照すると、A.1 のログと同様に、回線が再接続されてから 15 秒以内にホストの検知を行っている。この際にファイルの同期を行っていることがわかる。

また、ホスト 1 台と通信不可能な状況でも、自ホストを含む 3 台はファイルの同期を正常に行えたことが、ログから確認できる。

[2] 巻末 付録

```
Jan 24 20:46:10 host01 p2sync: Starting Host Check...
Jan 24 20:46:10 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:46:10 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:46:10 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:46:11 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:46:11 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:46:11 host01 p2sync: Host Check Completed.
Jan 24 20:46:11 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:25 host01 p2sync: REMOVED 192.168.0.2

                         — 中略 —

Jan 24 20:46:26 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:26 host01 p2sync: SENDTO 192.168.0.4 2 FILE(s)
Jan 24 20:46:27 host01 p2sync: Sending Done.
Jan 24 20:46:27 host01 p2sync: SENDTO 192.168.0.3 2 FILE(s)
Jan 24 20:46:28 host01 p2sync: Sending Done.
```

図 6.2: 対故障性に関する評価のログ

6.1.3　通信量に関する評価

移動計算機環境では、回線の状況が不安定なため、少ない通信量でファイル同期を行う必要がある。そこで、本システムがファイルの同期を行う際の通信量を測定する。測定に関しては、更新ファイルの数やホストの台数を変更し、それぞれの結果を比較する。

実験環境

前実験の環境に新たな 6 台のホストを加えて、最大 10 台のホストで実験を行った。

実験に使用したホスト名とアドレスを以下に示す。これらの計算機の詳細を表 6.2 に示す。

- 実験ホスト **A**
  - ホスト名: host01
  - IP アドレス: 192.168.0.1
- 実験ホスト **B**
  - ホスト名: host02
  - IP アドレス: 192.168.0.2

- 実験ホスト **C**
  - ホスト名: host03
  - IP アドレス: 192.168.0.3
- 実験ホスト **D**
  - ホスト名: host04
  - IP アドレス: 192.168.0.4
- 実験ホスト **E**
  - ホスト名: host05
  - IP アドレス: 192.168.0.5
- 実験ホスト **F**
  - ホスト名: host06
  - IP アドレス: 192.168.0.6
- 実験ホスト **G**
  - ホスト名: host07
  - IP アドレス: 192.168.0.7
- 実験ホスト **H**
  - ホスト名: host08
  - IP アドレス: 192.168.0.8
- 実験ホスト **I**
  - ホスト名: host09
  - IP アドレス: 192.168.0.9
- 実験ホスト **J**
  - ホスト名: host10
  - IP アドレス: 192.168.0.10

評価内容

更新ファイルの数

更新を行うファイルの数を変更し、通信量を測定する。

ホストの台数

ファイルの同期を行うホストの台数を変更し、全体の通信量を測定する。

表 6.2: 実験に使用した計算機 (2)

使用した計算機	使用 OS	CPU クロック	メモリ	台数
DynaBook PORTEGE 660CDT	Linux 2.0.36(Slackware 3.1)	150MHz	32MB	1
Let's Note CF-C33J8	Linux 2.2.1(RedHat 5.2)	233Mhz	96MB	1
Let's Note AL-N2	FreeBSD-2.2.7	166MHz	96MB	2
Let's Note AL-N2	FreeBSD-2.2.8	166MHz	96MB	1
DynaBook SS3010	FreeBSD-2.2.8	266MHz	64MB	2
DynaBook GT475	Linux 2.0.35(Slackware 3.1)	75MHz	8MB	1
Digital Hinote Ultra II	FreeBSD-2.2.7	120MHz	40MB	1
Libretto 60	FreeBSD-2.2.7	100MHz	16MB	1

結果

測定結果を、図 6.3 に示す。ファイルおよびホストの台数に正比例して、通信量は増加した。ファイルの総サイズと実際通信量に微妙な誤差があるのは、制御メッセージの通信量を含むためである。制御メッセージの通信量は微量であり、ファイルの総サイズが増えるにつれて、この誤差は小さくなる。

表 6.3: 通信量の測定

ファイル数 (総サイズ)	1 host	5 hosts	10 hosts
10(177KB)	178KB	900KB	1.8MB
50(798KB)	799KB	4.0MB	8.0MB
100(2.5MB)	2.5MB	12.5MB	25.0MB
500(7.3MB)	7.3MB	36.5MB	73.0MB

ここでは片方向のみの測定を行ったが、同期を行うホスト全てが同量のファイルを更新したとすると、全ホストの総通信量は、

$$
\begin{aligned}
(\text{総通信量}) &= (\text{ホスト 1 台と同期を取るための通信量}) \\
&\quad \times (\text{全ホスト台数}) \\
&\quad \times (\text{自ホストを除くホストの台数})
\end{aligned}
$$

となり、ホスト 1 台と同期を取るための通信量を a、全ホスト台数を x とすると、総通信量 $f(x)$

は、

$$f(x) \quad = \quad ax(x-1)$$

という式で表せる。ファイルの共有は自ホストを含み、最低 2 台以上で行われる。このため、全体の通信量はファイル数に正比例し、ホストの総数の増加に伴って指数関数的に増加してすることになる。図 6.3 の 2 次関数グラフは、ホスト一台あたりの通信量が 5MB と仮定した際の、全ホストの総通信量を表している。仮に、各ホストで約 5MB のファイル更新を行ったとする。各ホストが同期を取る際、ホスト一台とファイル同期を取る際の通信量が 5MB だとしても、ホストが 10 台に増加すると、その総通信量は 100MB 近くなる。

各ホストで実際に約 5MB の更新を行い、同期を取った時の通信量を図 6.3 上に点で示す。結果は理論値に近く、プロトコルが設計通りに動作していることが確認できる。

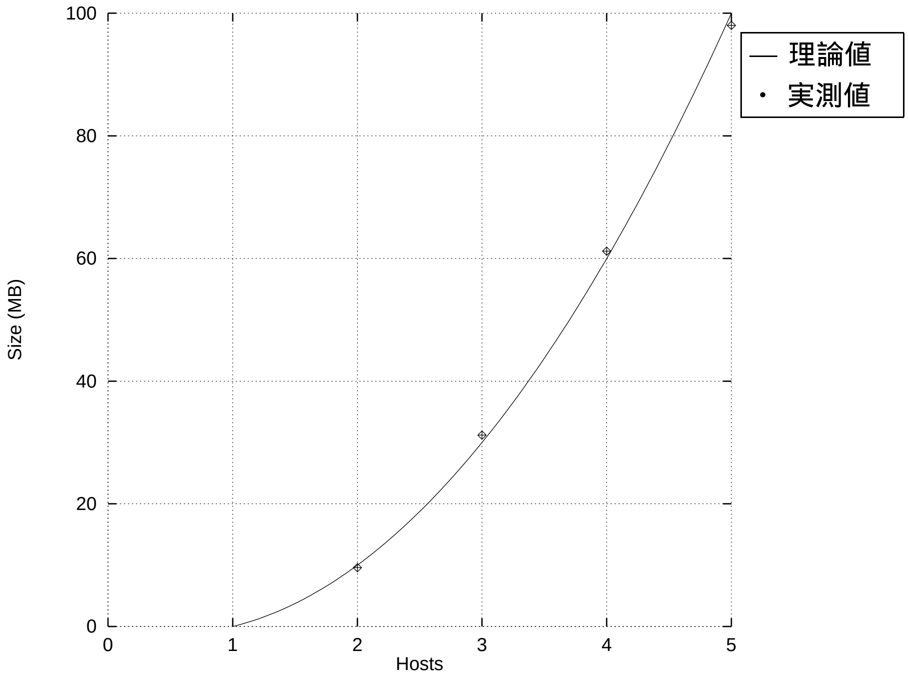

図 6.3: 一台あたりの通信量と全ホストの総通信量

6.2 総合評価

本システムの評価を、表 6.4 に示す。

前節の結果からわかるように、ファイルの通信量に問題は残るものの、作業継続性・耐故障性において、本システムが優れていることが確認できた。

表 6.4: システムの評価

システム	作業継続性	対故障性	トラフィック
本システム	○	○	△

6.2.1 設計目標の達成度合

ここでは、第 4 章で掲げた設計目標に関して、その達成度合を考察する。

- 複数ホスト間でそれぞれ同様の情報を保持し、共有できるようにする。
  ファイルの同期・共有に関しては、あるホスト上でのファイル更新内容が、ファイル同期後、正確に反映されていることを確認した。また、その際にファイルの同期が正常に行われていることを確認した。
- 計算機間の通信が不可能な場合も、利用者が回線の状態を意識せず、通常と同様の操作が行なえるようにする。
  作業継続性・対故障性に関する評価で述べたように、通信回線の不安定な状態でも通常と同様の動作が行えることが確認できた。

以上のことから、設計目標として定めた項目は達成できたと言える。

6.3 関連研究との比較

6.3.1 NFS

NFS は即時的なファイル更新を行うことができ、ユーザが書きこみを行ったファイルの整合性は保証される。本研究で提供するシステムでは、ファイルの即時的な更新・整合性の確保を保証していない。この面においては、NFS が優れている。

欠点として、低速な回線での応答性の悪さが挙げられる。また、回線の切断時には全く利用できない。クライアントサーバモデルを用いているため、サーバに障害が発生すると、サービスを受けるクライアントの動作は全て停止する。

本研究で提供するシステムでは、これらの問題を解決しており、低速回線利用時・回線切断時にも通信が可能である。

6.3.2 AFS

回線の切断が長時間に渡ると、クライアント上のキャッシュの整合性が確保できなくなる。このため、最終的にはファイルアクセスが不可能になってしまう。

本研究で提供するシステムでは、単一のサーバに依存しない設計をしている。よって、一つのホストからファイルを受信できなくても、他のホストが補い合い、ファイル共有を行うことができる。

安定した固定計算機環境で AFS を使用した場合は、ファイルの更新速度、整合性の確保の点で、本システムに比べ、優れている。

6.3.3 Coda

Coda では必要最低限のキャッシュしか行わないため、本システムと比べ、通信量を少なくすることができる。

欠点として、キャッシュしておくファイルを予め設定しておく必要があるため、キャッシュを行っていないファイルへのアクセスができない。本システムでは回線が使用可能な状態と時に、

更新したファイル全ての同期を取る。これによって、回線切断時にも、全てのファイルを利用することができる。

また、Coda の導入には、オペレーティングシステムのカーネルに変更を加える必要がある。このため、導入の容易さが犠牲になっている。

6.3.4　関連研究との比較に関するまとめ

評価項目を各システムに関して行う。この結果を、表 6.5 に示す。

作業継続性

回線切断時の動作に関しては、回線切断時のファイル操作環境を想定していない NFS や AFS に比べ、Coda および本システムは円滑にファイル操作を行うことができる。

対故障性

本システムでは複数の計算機間の通信を可能にしている。このため、サーバを起動する為の計算機を用意するシステムと比べ、耐故障性の面で優れている。

通信量

通信量については、ファイルの更新をサーバ上で直接行う、NFS が優れている。本システムではファイルの同期を各ホストに対して行うため、通信ホストの数が増えるにつれて、ネットワーク上の負荷が指数関数的に増加してしまう。

表 6.5: 関連研究との比較

システム	作業継続性	対故障性	トラフィック
NFS	×	×	○
AFS	△	△	○
Coda	○	○	○
本システム	○	○	△

第7章　結論

7.1　まとめ

本研究では、移動計算機環境という特殊な計算機環境において、ファイルの共有を行うことに着目してきた。今回の実装では、ファイルの更新や整合性の確保といった面で、固定計算機環境に最適化された情報共有システムに劣る。しかし、これは移動計算機環境に適したシステムを追及した結果であり、通信回線の不安定な環境下では、パフォーマンスを上げることができた。

ホスト検知を自動化することにより、円滑なコミュニケーション環境の提供を行うことができる。例えば会議室に各人が携帯型計算機を持ち寄り、ミーティング等を行う際に、ファイルの共有を行う必要があるとする。こうした状況でも、ホスト発見を自動化することによって、新たな設定を殆ど必要とせずにファイルの共有を行うことができる。

7.2　今後の課題

7.2.1　トラフィックの軽減

複数ホストとの通信を行う際、現在の実装では、ホスト一台づつと同様の通信を行い、ファイルの同期を取る。この方式では、一対一での通信に比べ、通信量が増加してしまう。移動計算機環境の不安定な通信環境を考慮し、この通信量をできるだけ軽減する必要がある。

7.2.2　ファイルシステムとしてのインタフェース

すでに第 4 章で述べたように、本システムは、その最終的な形として、ファイルシステムを提供するという目標を立てている。現在の実装では、ファイルの新旧を判別するため、ファイル名の後に時間を基にしたシリアル番号を付加している。このままでは、利用者が作成したファイルとの完全な対応が行えなくなってしまう。そこで、本システムを xxfs [7] のようなシステムと組

み合わせることで、ファイルシステムとしてのインタフェースを提供する必要がある。

7.2.3 通信可能ホストの自動検知

現在、本システムを導入しているホストの検知は設定ファイルの内容に従って行われている。このため、利用者は予め通信ホストの設定が必要となり、柔軟性のあるファイル共有を行うことができない。例えば、各利用者が離れた場所から計算機を持ち寄ってファイル共有を行うような状況を想定する。各利用者は集まった先で設定を行う手間が生じてしまう。これを解決するためには、本システムが通信可能なホストを自動的に検知する機能を提供する必要がある。

謝辞

本研究を進めるにあたって様々なアドバイスをして下さった、慶應義塾大学環境情報学部 徳田 英幸 教授、慶應義塾大学環境情報学部 村井 純 教授に深く感謝致します。

忙しい中、本論文の執筆とシステムの実装に関して常に惜しみない助力を与えて下さった、慶應義塾大学環境情報学部 植原 啓介 助手に深く感謝します。

同様に、論文の内容について忙しい中アドバイスを下さった、慶應義塾大学大学院政策・メディア研究科博士課程 渡辺 恭人 氏、慶應義塾大学大学院政策・メディア研究科修士課程 宅間 啓 氏に感謝致します。

また、研究テーマについての助言を与えて下さった、奈良先端科学技術大学院大学情報科学センター 砂原 秀樹 助教授をはじめとする Rover グループの諸氏に感謝致します。

そして、gnuplot の使用法とその他、非常に有効な助言を与えて下さった、慶應義塾大学大学院政策・メディア研究科後期博士課程 冨永 明宏 氏に感謝致します。

著者が論文執筆に集中できるようにと、執筆中に様々な面で気をつかってくれた、慶應義塾大学環境情報学部 川喜田 佑介 君と 佐藤 雅明 君に、心から感謝します。

本論文執筆中に著者が付き合いの悪くなる中でも、友達をやめずに暖かな目で見守ってくれた、慶應義塾大学総合政策学部の 井上 理 氏 と石幡 雄高 氏に感謝します。

最後に、本論文を執筆するにあたって論文の内容や実装について様々な意見を提供して下さった、慶應義塾大学村井研究室の諸氏に感謝します。

著者近影

参考文献

[1] SUN microsystems, Inc. *NFS: Network File System Protocol Specification.* RFC 1094, 1989.

[2] E. R. Zayas. *AFS-3 Programmer's Reference: Architectural Overview.* Transarc Cropora-tion, 1991.

[3] J. J. Kister and M. Satyanarayanan. *Disconnected Operation in the Coda File Sytem.* Proc. 13th Symposium on Operationg Systems Principles, 1991.

[4] S.L. Ererson. *Usenet, a bulletin board for Unix users.* USENIX, 1983.

[5] P. Lapsleyn and B. Kantor. *Network News Transfer Protocol: A proposed standard for the stream-based transmission of news.* RFC 959, 1986.

[6] J. Postel. *Transmission Control Protocol.* RFC 793, 1973.

[7] 関康治. 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. 情報処理学会, 1998.

付 録 A　ログ

A.1　ログ 1: 移動時の作業継続性に関する評価

```
[192.168.0.53]
Jan 24 20:13:16 host01 p2sync: Sending Broadcast Message...
Jan 24 20:13:16 host01 p2sync: ADDHOST 192.168.0.2
Jan 24 20:13:16 host01 p2sync: ADDHOST 192.168.0.3
Jan 24 20:13:17 host01 p2sync: ADDHOST 192.168.0.4
Jan 24 20:13:31 host01 p2sync: Starting Host Check...
Jan 24 20:13:31 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:13:31 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:13:31 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:13:31 host01 p2sync: LIVES 192.168.0.2
Jan 24 20:13:31 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:13:31 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:13:31 host01 p2sync: Host Check Completed.
Jan 24 20:13:32 host01 p2sync: Sending Broadcast Message...
Jan 24 20:13:46 host01 p2sync: Starting Host Check...
Jan 24 20:13:46 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:13:46 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:13:46 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:13:46 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:01 host01 p2sync: REMOVED 192.168.0.2
Jan 24 20:14:01 host01 p2sync: REMOVED 192.168.0.3
Jan 24 20:14:01 host01 p2sync: REMOVED 192.168.0.4
Jan 24 20:14:01 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:16 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:31 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:46 host01 p2sync: Sending Broadcast Message...
Jan 24 20:14:46 host01 p2sync: ADDHOST 192.168.0.2
Jan 24 20:14:46 host01 p2sync: ADDHOST 192.168.0.4
Jan 24 20:14:46 host01 p2sync: ADDHOST 192.168.0.3
```

```
Jan 24 20:15:01 host01 p2sync: Starting Host Check...
Jan 24 20:15:01 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:15:01 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:15:01 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:15:01 host01 p2sync: LIVES 192.168.0.2
Jan 24 20:15:01 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:15:01 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:15:01 host01 p2sync: Host Check Completed.
Jan 24 20:15:02 host01 p2sync: Sending Broadcast Message...
Jan 24 20:15:02 host01 p2sync: SENDTO 192.168.0.2 2 FILE(s)
Jan 24 20:15:03 host01 p2sync: Sending Done.
Jan 24 20:15:03 host01 p2sync: SENDTO 192.168.0.4 2 FILE(s)
Jan 24 20:15:05 host01 p2sync: Sending Done.
Jan 24 20:15:05 host01 p2sync: SENDTO 192.168.0.3 2 FILE(s)
Jan 24 20:15:06 host01 p2sync: Sending Done.
Jan 24 20:15:06 host01 p2sync: RECVFROM 192.168.0.4 5 FILE(s)
Jan 24 20:15:08 host01 p2sync: Receiving Done.
```

A.2 ログ 2: 対故障性に関する評価

```
[192.168.0.53]
Jan 24 20:45:40 host01 p2sync: Sending Broadcast Message...
Jan 24 20:45:40 host01 p2sync: ADDHOST 192.168.0.2
Jan 24 20:45:41 host01 p2sync: ADDHOST 192.168.0.4
Jan 24 20:45:41 host01 p2sync: ADDHOST 192.168.0.3
Jan 24 20:45:55 host01 p2sync: Starting Host Check...
Jan 24 20:45:55 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:45:55 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:45:55 host01 p2sync: LIVES 192.168.0.2
Jan 24 20:45:56 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:45:56 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:45:56 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:45:56 host01 p2sync: Host Check Completed.
Jan 24 20:45:56 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:10 host01 p2sync: Starting Host Check...
Jan 24 20:46:10 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:46:10 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:46:10 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:46:11 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:46:11 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:46:11 host01 p2sync: Host Check Completed.
Jan 24 20:46:11 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:25 host01 p2sync: REMOVED 192.168.0.2
Jan 24 20:46:25 host01 p2sync: Starting Host Check...
Jan 24 20:46:25 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:46:25 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:46:25 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:46:26 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:46:26 host01 p2sync: Host Check Completed.
Jan 24 20:46:26 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:26 host01 p2sync: SENDTO 192.168.0.4 2 FILE(s)
Jan 24 20:46:27 host01 p2sync: Sending Done.
Jan 24 20:46:27 host01 p2sync: SENDTO 192.168.0.3 2 FILE(s)
Jan 24 20:46:28 host01 p2sync: Sending Done.
Jan 24 20:46:40 host01 p2sync: Starting Host Check...
```

```
Jan 24 20:46:40 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:46:40 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:46:40 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:46:40 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:46:40 host01 p2sync: Host Check Completed.
Jan 24 20:46:40 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:41 host01 p2sync: ADDHOST 192.168.0.2
Jan 24 20:46:48 host01 p2sync: RECVFROM 192.168.0.4 1 FILE(s)
Jan 24 20:46:51 host01 p2sync: Sending Done.
Jan 24 20:46:55 host01 p2sync: Starting Host Check...
Jan 24 20:46:55 host01 p2sync: CHECK 192.168.0.4
Jan 24 20:46:55 host01 p2sync: CHECK 192.168.0.3
Jan 24 20:46:55 host01 p2sync: CHECK 192.168.0.2
Jan 24 20:46:55 host01 p2sync: LIVES 192.168.0.4
Jan 24 20:46:55 host01 p2sync: LIVES 192.168.0.3
Jan 24 20:46:56 host01 p2sync: LIVES 192.168.0.2
Jan 24 20:46:56 host01 p2sync: Host Check Completed.
Jan 24 20:46:56 host01 p2sync: Sending Broadcast Message...
Jan 24 20:46:56 host01 p2sync: SENDTO 192.168.0.2 2 FILE(s)
Jan 24 20:46:57 host01 p2sync: Sending Done.
```