モーバイルコンピューティング 7－3
（1998. 12. 4）

移動体を考慮した情報共有システムの設計と実装

石井 徹[†1] 植原啓介[†1]
渡辺恭人[†2] 村井 純[†1]

本研究では、回線の切断・高遅延状態を利用者が意識せず、ファイルを複数ホスト間で共有することのできるシステムを提案し、その評価を行った。特に不安定な通信環境によって生じる障害を軽減することを考慮し、システムを設計・実装した。本論文では、最初に現在の移動体通信環境に適したファイル共有システムの必要性と、現存のファイル共有システムの問題点を述べる。そして、問題点解決法を提案し、それに基いた実装を行う。現在、この実装は Linux 2.0.35 で稼働している。

A Design and Implementation of The Information Sharing System for Mobile Computing Environment

TOHRU ISHII ,[†3] KEISUKE UEHARA ,[†3]
YASUHITO WATANABE [†4] and JUN MURAI [†3]

This paper proposes and evaluates a file sharing system between multiple hosts without being anxious about network connectivity. Primary goal of this system is to decrease the troubles which are caused under the unreliablenetwork environment. First, we describe a necessity of a file sharing system under the current network computing environment and point out its problems. Then we propose a solution for these problems with describing implementation of our system. This implementation is now running on Linux 2.0.35.

1. はじめに

現在、計算機の小型化・軽量化は進み、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。こうした中、移動先で他のホストとのファイルのやり取りを行う場合に、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。

一方、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、現存のファイル共有システムをそのまま使用した場合、高遅延や回線の切断断などの障害が生じた際に、長時間にわたる動作の停止や、システムそのものの終了を招くことがある。

また、多くの分散システムは、主に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して情報の取得・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ことがある。

現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。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たこれらのシステムでは、不安定な通信状況に対応できない。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。本研究では、この問題点を解決するための情報共有システムを実現することを目的とする。

†1 慶應義塾大学 環境情報学部
†2 慶應義塾大学大学院 政策・メディア研究科
†3 Keio University, Faculty of Environment Information
†4 Keio University, Graduate School of Media and Governance

2. 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、以下に示す目標を定め、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。
- 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。

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が、今回はファイルシステムに着目し、上記の目標の実現を優先する。

3. 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が同様のファイルを所有し、単一のサーバへの集中を避ける。また、ファイルをローカルディスク上に所有するので、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。これによって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できる。この反面、定期的にファイルの同期を行うために、即時的なファイル更新は難しく、ファイルの整合性に問題を持つ。

本研究では、移動体通信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。そのため、定期的なファイル同期機能の実現を優先する。

3.1 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1に示す。

ファイル共有を行うホスト群である、Host A, B, C, D が存在する。Host A は Host B, C と通信可能である。Host D は、Host A と直接することはできないが、Host B との通信が可能である。Host A で作成されたファイルは、Host A が通信可能である HostB, C との間で同期を取った時点で Host B, C に複製される。Host B もまた通信可能はホストと同期を取るため、この時点で Host D にファイルが複製される。結果として、Host A で作成されたファイルは、間接的に Host D へと送信される。

3.2 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　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、

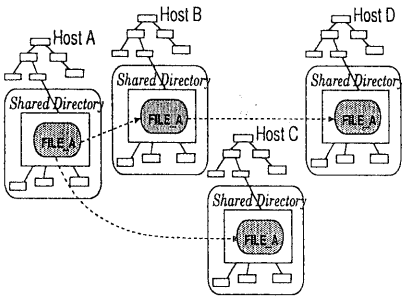

図1　ファイルの中継

ローカルホストと他ホストとのファイル同期を行う。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に、ファイルアクセスを行うことができる。

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[1),2)] 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、情報取得時にまず、こうした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。利用者は設定時にこのグループを指定することで、ファイル共有時に、グループに属するファイルを取得することができる。

3.2.1 ファイル同期機能

ホスト間のファイル同期部分には NetNews 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。

3.2.2 グループ管理機能

各ホスト上に設定ファイルを容易し、本システムがそれを参照する。システムは設定ファイルに記述されたホストとの通信を試み、成功すると自分のホスト名とグループ名を送信する。通信先では、送信されてきたホスト名とグループ名を確認し、自分の設定ファイルに記述されていることを確認すると通信を許可する。

設定ファイル内にグループが存在しない場合は、通信を終了する。

3.3 システムの動作

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されると、ホスト上のデーモンがそれを検知し、更新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し合い、自分の持っ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。

4. 設　　計

4.1 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

新規作成・更新されたファイルを管理する。

グループ管理システム

通信可能なホストを判別する。

ファイル同期デーモン

他ホストと、ファイルの同期を取る。

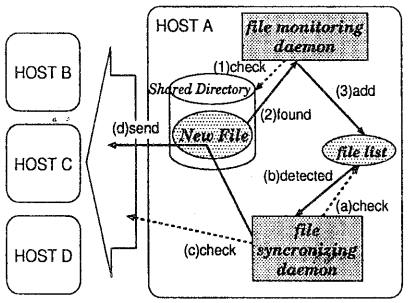

図 2　システム構成図

4.2 処理の流れ

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、その処理の流れを図 2 に示す。

4.2.1 ファイル監視デーモンの動作

図 2 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、前回の結果に存在しないファイルは新規ファイルとして扱う。前回の結果に存在するが、更新日時の新しくなっているファイルは更新されたファイルとして扱う。

(3)　発見した新規ファイルもしくは更新ファイルを、更新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、今回の結果で上書きする。

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。

4.2.2 ファイル同期デーモンの動作

図 2 中の (a) ～ (d) 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。

(a)　定期的にファイル更新リストをチェックする。

(b)　更新ファイルを発見する。更新ファイルが存在しない時は処理を中断し、一定時間後に (a) の処理を繰り返す。

(c)　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。この、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。

(d)　接続後、ファイルの同期を行う。

4.2.3 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し、利用者の設定したグループに属するファイルを選択する。処理の流れについては、4.3.2 で述べる。

4.3 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基いて通信を行う。

4.3.1 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 1 に示す。

表 1　ファイル同期プロトコルのメッセージ
Table 1　Messages of File Syncronyzation Protocol

メッセージ	役割
IHAVE	更新・新規作成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	ファイル受信が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。

表 1 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 3 に示す。

(1)　Host A に存在するファイルの内、新規作成・更新されたものを検知し、IHAVE メッセージに続いて更新ファイルのリストを送信する。

(2) Host B は IHAVE メッセージに続く更新ファイルのリストを受信し、ローカルディレクトリと比較する。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 A は SENDME メッセージの後に送信された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。

(4) 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、(3)に戻り、全てのファイルを送信するまで、動作を繰り返す。

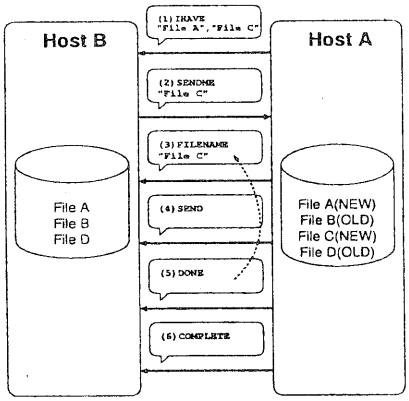

図 3 ファイル同期プロトコルの動作

4.3.2 グループ管理プロトコル

本プロトコルは、ホスト間での通信開始時に使用する。本プロトコルの動作により、通信先のホストが同一グループに属することを確認する。この時点で、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 2 に示す。

表 2 グループ管理プロトコルのメッセージ
Table 2 Messages of Group Management Protocol

メッセージ	役割
HELLO	通信を開始する。
GROUP	自分の所属するグループ名を通知する。
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。

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 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。

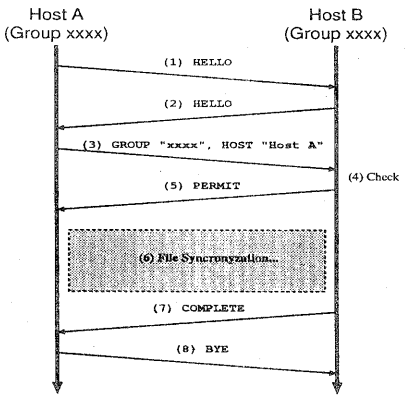

図 4 グループ管理プロトコルの動作

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。

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か、それとも上書きする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シリアル番号をつけ、ファイル名の重複を避ける。

ファイル共有グループの管理

5. 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。
- システムの導入を容易にする。

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 2.0.35 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、システム全て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 に示す。

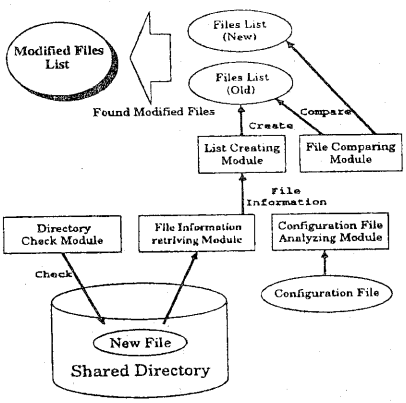

図 5　ファイル監視機能のモジュール構成

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。

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行う。

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 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。新しく作成されたファイルリストに存在し、前回作成時のファイルリストには存在しないファイル、または前回作成時のファイルリストに記述されているタイムスタンプよりも新しいファイルを、新規作成ファイル、もしくは更新ファイルとして抽出する。

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、6 に示す。

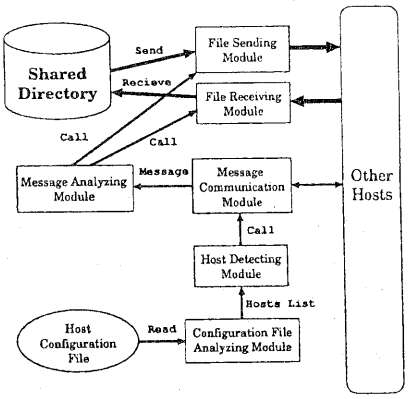

図 6　ファイル同期機能のモジュール構成

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。

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。

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。

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、7に示す。

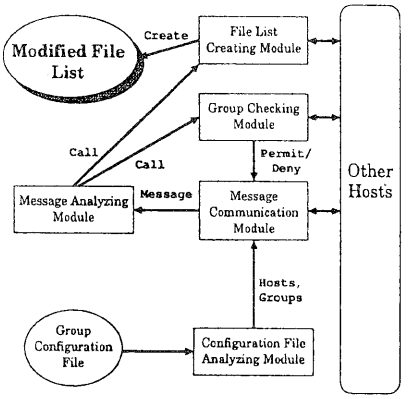

図7 グループ管理機能のモジュール構成

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 接続先のホストが通信可能なグループに属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。

6. 評　　価

前節で実装目標として掲げた項目はすべて達成できた。現在、Linux 上での動作を確認した。特に Linux に依存した実装は行っていないため、未評価ではあるが、BSD 系 UNIX や、SunOS への移植は容易であり、現在計画中である。

システムの動作確認は、意図的に回線を切断して行った。更新されたファイルは回線切断前に複製され、ローカルディスク上に保存されたファイルを扱えることを確認できた。

本システムで提供する機能の一つである、ファイルの中継については、まず2つのホスト間でファイルの共有を行い、ファイルが複製された時点で計算機を移動させ、別ホストに接続することで動作の確認を行った。ファイルは、計算機が移動先でネットワークに接続した後、移動先に用意したホストに複製された。

今回の実装では、接続可能であるホストを、利用者が手動で設定ファイルに記述する必要がある。このた、移動時の接続において、柔軟性に欠ける。

7. 関連研究との比較

7.1 NFS

NFS(Network File System)[3] は、Sun Microsystems Inc. が開発した、共有ファイルシステムである。NFS はクライアントサーバモデルに基いて設計されている。

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。

7.2 AFS

AFS[4] 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、コールバックと呼ばれる大容量キャッシュ管理機能が挙げられる。このキャッシュを利用して、サーバとの通信が不可能になっても、キャッシュを利用してファイル操作を行うことができる。この機能によって、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。

7.3 Coda

Coda[5]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おいて優れている。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。この機能によって、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。

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。

8. 今後の課題

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。

通信可能ホストの自動検知

現在、通信を行うホスト群の管理は、各ホストに予め設定記述されたファイルを基に行うため、ホスト発見の動作については自動化がなされていない。移動する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大きな負担となる。この問題を解決するため、今後の実装では本システムを導入するホストの自動検知を行う必要がある。

ホスト検知の自動化を行うための手法を、今後の指針として以下に定める。ホストの自動検知機能をグループ管理機能と組み合わせ、処理を円滑に行う。

(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。
(3) 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。
(4) 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。
(5) 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。
(6) (5)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。

9. おわりに

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、固定計算機環境に劣る面がある。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。

謝辞 本論文を執筆するにあたり、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。

参考文献

1) Ererson, S.: *Usenet, a bulletin board for Unix users*, USENIX (1983).
2) Lapsleyn, P. and Kantor, B.: *Network News Transfer Protocol: A proposed standard for the stream-based transmission of news*, RFC 959 (1986).
3) SUNmicrosystems, I.: *NFS: Network File System Protocol Specification*, RFC 1094 (1989).
4) Zayas, E. R.: *AFS-3 Programmer's Reference: Architectural Overview*, Transarc Croporation (1991).
5) Kister, J. J. and Satyanarayanan, M.: *Disconnected Operation in the Coda File Sytem*, Proc.

13th Symposium on Operationg Systems Principles (1991).