ONFS：一時的なネットワーク環境下ですぐに利用できる共有ファイルシステム

福田伸彦[†] 楯岡孝道[††]
中村嘉志[†,☆] 多田好克[†]

会議などの人々の集まる場にネットワークを構築し，参加者の計算機を接続することで，様々な情報交換や共有が可能となる．本稿では，そのような場における情報の交換に有用な共有ファイルシステム ONFS（Offhand Network Filesystem）を提案する．会議などの場に設置されるネットワークは，その場だけで一時的に運用される．そのため，(1) 計算機の接続と切り離しが頻発する，(2) 構成メンバがその場によって異なる，という状況が生じる．既存のシステムで (1)，(2) の状況に対応するためには，その場での繁雑な設定作業をしなければならない．そこで，これらの問題を解決するために，我々は NFS（Network File System）をベースに，サーバの自動探索，一時ユーザの管理，切断時の対応処理の機能を ONFS に実装した．ONFS を利用することで，ユーザは計算機をネットワークに接続するだけで即座にファイル共有を実現できる．

ONFS: A Shared File System Usable Immediately under Circumstances of Temporary Network

Nobuhiko Fukuda,[†] Takamichi Tateoka,[††]
Yoshiyuki Nakamura[†,☆] and Yoshikatsu Tada[†]

In places where people come together, such as a meeting, participants can exchange and share information if they build a network with their computers. In this paper, we propose a shared file system, ONFS (Offhand Network Filesystem), which is useful to exchange information. The temporary network is employed only at that place and time. As a result, the followings occur; (1) computers are connected and disconnected frequently, and (2) members vary from place to place. Existing systems impose users to handle troublesome settings under such situations. To solve these problems, we implemented following functions on ONFS, based on NFS (Network File System); an automatic server searching, a temporary user management, an operation at disconnection. Users using ONFS can share files immediately by connecting a computer.

1. はじめに

携帯型計算機は，移動先での計算機利用を可能にした．会議に計算機を持参して，プレゼンテーションや，議事録の作成を行う人々が増えてきた．また，携帯電話を経由して，いつでもインターネット上の情報にアクセスできるようになった．このような有用性から，携帯型計算機によるモーバイルコンピューティングは，今後ますますさかんになると考えられる．

会議などの人々の集まる場にローカルネットワークを構築し，参加者の計算機を相互に接続することで，参加者間で様々な情報交換や共有が可能となる[1)]．たとえば，会議や講義での資料の配布，計算機を用いた演習での課題の提出，多人数によるプログラム開発でのソースコードの共有などがあげられる．そこで本研究では，そういった人々の集まる場において，情報の交換に有用な共有ファイルシステム ONFS（Offhand Network Filesystem）を構築した．

ONFS は，会議の場に一時利用のためのファイルサーバを設置し，参加者がそれぞれ各自の計算機からアクセスできるようにする．共有ファイルシステムであるため，サーバのファイルシステムに透過的にアク

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 電気通信大学情報工学科
Department of Computer Science, The University of Electro-Communications
☆ 現在，産業技術総合研究所
Presently with National Institute of Advanced Industrial Science and Technology (AIST)

セスでき，ファイル転送のための専用コマンドを必要としないという利点がある．

ONFS の設計にあたっては，その場で即座に利用できるよう，ネットワークに物理的に接続するだけで利用でき，切り離すだけで終了できるシステムとすることに留意した．また，不特定の人々が集まる場で利用する場合に，限られた人だけにファイルの書き込みを許可できるよう，ユーザ単位でのアクセス制御を提供することにした．

会議の場に設置されるネットワークは，その場だけで一時的に運用される．そのため,(1) 計算機の接続と切り離しが頻発する,(2) 構成メンバがその場によって異なる，という状況が生じる．既存のシステムで (1), (2) の状況に対応するためには，その場での繁雑な設定作業を必要とする．これは，既存の共有ファイルシステムが，安定したネットワークのもとで定常的に運用されることを前提としているからである．繁雑な作業は具体的には，サーバおよび共有対象の指定，クライアントのアクセス制御情報の調整，ユーザの設定と削除などである．これらの繁雑な作業によって，実際に共有ファイルシステムを利用するまでに多大な時間と労力を要する．また，設定ミスにより，システムの安定性やセキュリティに影響を与えてしまいかねない．

これらの問題を解決するために，ONFS は NFS (Network File System)[2] をベースに，サーバの探索，一時ユーザの管理とアクセス制御，切断時の対応処理の機能を付加した．本稿は，これらの機能を中心に，一時的なネットワーク環境で即座に利用できる共有ファイルシステムの設計と実装について論じる．

本稿の構成を示す．2 章では ONFS の想定する利用環境を述べる．3 章では既存システムの問題を示す．4 章では ONFS に対する要求と設計方針について，5 章では設計と実装について説明する．6 章では ONFS の評価を行い，7 章では考察を行う．最後に 8 章で本稿をまとめる．

2. 利用環境

本章では，ONFS の想定する利用環境について説明する．ONFS は，会議などの人々の集まる場でのファイル交換を目的とした共有ファイルシステムである．ONFS を利用する機会には，一般の会議をはじめ，研究会やシンポジウム，学校での講義や演習，合宿形式のプログラム開発やソースコードの精読を行うソース読み会などがあげられる．

2.1 形　態

ONFS は，ネットワークを利用した共有ファイルシステムによって，参加者間でのファイル交換を実現する．共有ファイルシステムの利点は，透過的なファイルアクセスを行えることである．ユーザは，専用のコマンドを用いてファイルを転送することなく，通常のファイル操作コマンドを用いてファイルのコピーや編集を行うことができる．

会議の場では，ファイルを提供したい参加者がファイルサーバを稼働させ，その場にいる参加者間で共有利用する．たとえば，参加者は会議資料ファイルのコピーを行ったり，プレゼンテーションのデータファイルの閲覧を行ったりする．また，参加者からのファイルアクセスは読み込みだけではなく，書き込みも行われる場合がある．たとえば，演習における課題の提出や，プログラムの共同開発におけるソースコードの編集などである．

2.2 機　器

ONFS の利用に必要な計算機やネットワーク機器は，会議の参加者がそれぞれ用意する．一般的には，参加者が普段携帯している計算機を利用する．ネットワークは Ethernet や無線 LAN などのブロードキャスト通信が可能な媒体を利用し，単一セグメントで構成する．ネットワークプロトコルは一般的なインターネットプロトコルを用いる．

2.3 規　模

会議の規模は数名の小規模な会合から数十名程度の研究会や講義，そして百名程度のシンポジウムまでを想定する．この程度の規模であれば，単一セグメントで構成可能である．

2.4 期　間

会議の期間は数時間程度から数日程度を想定する．ネットワークは期間中のみしか構築しない．また，期間中参加者はネットワークに対する計算機の接続や切り離しを随時行う．

2.5 組　織

会議の参加者は，大学や研究会といった，参加者の異なる複数の組織に所属していることを想定する．参加者は，1 台の計算機を持ち歩き，各組織で開催される会議に参加し計算機を利用する．

会議の場にネットワークを構築し，共有ファイルシステムを利用することで，情報交換を円滑に進めることが可能となる．また，参加者は普段から使い慣れている自分の計算機を使うことができる．

3. 既存システムの問題

本章では，2 章で述べた一時的なネットワーク環境において，既存の共有ファイルシステムを用いた場合

の問題について議論する．現在，広く利用されている共有ファイルシステムとして NFS や SMB/CIFS（Server Message Block/Common Internet File System）[3),4)]がある．しかし，これらのシステムは定常的なネットワーク環境での利用を前提としており，一時的なネットワーク環境での利用を想定していない．以下に，各システムを利用するうえでの問題点を示す．

3.1 NFS

NFS は Sun Microsystems 社が開発した分散共有ファイルシステムである．NFS プロトコルはこれまでにバージョン 2[5)]，バージョン 3[6)]およびバージョン 4[7)]が規定されており，Unix 系 OS 環境では事実上の標準となっている．NFS プロトコルの特長として，サーバがクライアントの状態を持たない（ステートレスサーバ）ようになっているため，サーバ障害からの復旧が容易であることがあげられる．

NFS は固定のネットワーク環境での定常運用を前提とした設計となっている．そのため，NFS を一時的なネットワーク環境で利用した場合に以下の問題が生じる．

サーバの明示指定

NFS クライアントは，利用したいサーバのアドレスとディレクトリ名を明示的に指定する必要がある．定常的なネットワークでは，DNS（Domain Name System）[8)]などの名前解決サービスが利用できるため，アドレスではなくホスト名でサーバを指定できる．また，一度設定を行えばネットワーク構成が変化しない限り変更する必要はない．

しかし，一時的なネットワーク環境では計算機に割り当てられるアドレスを事前に特定できない．また，名前解決サービスも利用できないことが多い．そのため，利用のたびにサーバの情報をその場で調べて設定する手間が生じる．

アクセス中の切断

一時的なネットワーク環境では，計算機の切り離しが頻繁に発生する．NFS サーバに対してアクセス中に通信が不能になると，NFS クライアントはサーバとの通信が回復するまで待ち続けるか，中断までに長いタイムアウト時間を必要とする．その結果，クライアントの動作に支障を与えてしまう．

ファイル所有者情報の整合

Unix 系 OS 環境では，ユーザ識別子/グループ識別子（UID/GID）を利用してユーザを識別している．NFS プロトコルでは，ファイル所有者情報として UID/GID を透過的に扱うため，NFS サーバ/クライアント間で UID/GID 情報を統一する必要がある．しかし，一時的なネットワーク環境では，ネットワークを構成する計算機は参加者がそれぞれ持ち込んだものであり，UID/GID 情報が統一されていない．UID/GID 情報の整合をとるためには，サーバとそれを利用しているすべてのクライアントでの設定を統一しなければならず，手間が大きい．

3.2 SMB/CIFS

SMB/CIFS は Microsoft 社の Windows 系 OS 環境における標準のファイル/プリンタ共有システムである．Unix 系 OS 環境で SMB/CIFS を利用するためのシステムとして，Samba[9),10)]や Sharity[11)]などがある．

SMB/CIFS はクライアント上でサーバ一覧を参照するためのブラウジングと呼ばれる機構を持つ．したがって，クライアントはアクセス時にサーバ計算機のアドレスを明示指定する必要がない．しかし，特定のサーバがネットワーク上のサーバ一覧の情報を持つため，通信量は減らせるが，そのサーバがネットワークから切り離されたときに他のサーバがその代わりとなるまでサーバ一覧に参照できなくなる問題がある．

SMB/CIFS では共有ファイルシステムの利用開始時にユーザの認証を行う．アクセス制御が必要な場合は，利用前にサーバでユーザの登録が必要である．ユーザの登録作業の内容は実装によって異なるが，作業の手間が生じる．

3.3 Ufo

Ufo[12)]はユーザレベルで利用可能なグローバルファイルシステムである．Ufo は FTP（File Transfer Protocol）や HTTP（Hyper Text Transfer Protocol）を用いて，大規模ネットワーク上のリモートファイルへの透過的なアクセスを提供する．ファイルシステムの実現には，proc ファイルシステムのトレース機能を用いたシステムコールのフックによる方法を用いている．

Ufo を用いることで，FTP や HTTP によるファイルアクセスを，ファイルシステムのインタフェースで透過的に実現できる．しかし，アクセス時にファイル全体をクライアントにコピーするため，全体の転送が完了するまでファイルにアクセスできない問題がある．また，サーバへのファイルの書き込みは，クローズ時に FTP でファイル全体を転送する手法のため，巨大なファイルを扱うときに遅延が大きくなる．

Ufo ではサーバ側は FTP あるいは HTTP サーバである．そのため，サーバでユーザの認証を行う場合，利用前にユーザを登録する必要がある．

3.4 Coda

Coda[13)]は切断操作に対応した分散ファイルシステ

ムである．Coda のクライアントはキャッシュを持ち，切断中のファイルアクセスを可能としている．ファイルの更新は，クライアントがサーバに再接続されると自動的に行われる．また，Coda はすべてのサーバボリュームを単一の名前空間に自動構成する．

Coda はユーザを単一サーバで集中的に管理するため，固定のサーバを用意する必要がある．そのため，一時的なネットワーク環境での利用は，サーバの準備や設定の手間が大きい．

3.5 PFS

PFS [14] は通信環境に動的に適応する共有ファイルシステムである．PFS はクライアントキャッシュを用意し，通信路の状態に応じてアクセス方法を変化させることで，低速な回線の利用時や切断時にもユーザに高速なファイルアクセスを提供する．PFS は NFS プロトコルを利用してユーザプロセスとして実現することで導入を容易にし，移植性の高いシステムを実現した．

PFS は固定のファイルサーバと携帯型計算機間でのファイル共有を目的としたシステムである．そのため，NFS の問題と同様に，サーバの明示指定が必要となる問題と，サーバ/クライアント間での UID/GID 情報の整合がとれない問題がある．

4. ONFS の設計方針

既存の共有ファイルシステムを一時的なネットワーク環境で利用する場合，3 章で述べた問題が発生する．そこで，本研究ではそれらの問題を解決し，一時的なネットワーク環境下で即座に利用できる共有ファイルシステムの開発を行った．本章では，ONFS の設計方針について述べる．

4.1 システムへの要求

一時的なネットワーク環境で利用するための共有ファイルシステムに対する要求を以下に示す．

計算機の特定

一時的なネットワーク環境では，DHCP（Dynamic Host Configuration Protocol）による計算機アドレスの動的割当てがよく利用される．動的割当ては，事前に計算機にアドレスを割り当てる必要がなく，かつアドレス設定を自動化できる利点がある．しかし，計算機に割り当てられるアドレスを特定できないため，その場で通信したい計算機のアドレスを調べるための機構が必要である．

ユーザの識別

Unix 系 OS などのマルチユーザ OS では，ファイルごとに所有者属性があり，アクセス制御に利用している．ファイルの所有者はユーザ/グループの識別子によって表現される．定常的なネットワーク環境であれば，ネットワーク内で一意な識別子を設定できる．しかし，個人利用している計算機で構成される一時的なネットワーク環境では，計算機ごとに識別子が異なる．そこで，サーバ/クライアント間でユーザ/グループの識別子の整合を保ち，適切なアクセス制御を可能にするための機構が必要である．

切断時の対応

定常的なネットワーク環境では，障害が発生しない限り常時通信可能であり，一時的に通信が切断した場合でも遠からず復旧する．しかし，一時的なネットワーク環境ではネットワークが存在する時間が限られており，利用終了後は物理的に通信が不可能となる．

既存の共有ファイルシステムの実装には，切断前にクライアントが終了作業を適切に行わないとシステムの動作に不具合が発生するものがある．また，クライアントの利用中に不注意でサーバを切り離してしまう場合もある．そこで，不意の切断時にもシステムの動作に影響を与えないようにしなければならない．

4.2 システムの設計方針

前節の要求をふまえて，ONFS の設計を行った．以下にその方針を示す．

サーバ指定の自動化

クライアントはネットワーク上の利用可能なサーバを探索し，サーバとそのサーバの提供する共有ディレクトリの一覧を自動的に列挙する．それにより，クライアントのユーザがサーバのアドレスや共有ディレクトリを明示的に指定しなくてもアクセスできるようになる．

ユーザ管理の自動化

サーバはクライアントのユーザを自動的に登録し，ファイルのアクセス制御を可能にする．そして，ファイルアクセス時にサーバ/クライアント間のユーザ/ グループの識別子の変換を動的に行い，整合を保つようにする．それにより，サーバのユーザが識別子の管理や整合性の維持を気にしなくてもよくなる．

切断時処理の自動化

クライアントはサーバとの接続が切断したときに，自動的に処理の中断を行い，システムの動作に影響を与えずに利用を終了できるようにする．それにより，クライアントのユーザがネットワークから切り離すときに明示的な終了作業を行わなくてもよくなる．

5. ONFS の設計と実装

本章では，ONFS の設計と実装について説明する．

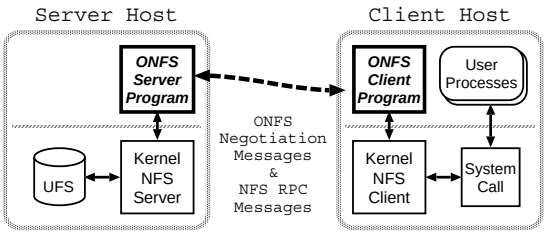

図 1 システム構成
Fig. 1 System components.

5.1 概　　要

ONFS では共有ファイルシステムのプロトコル設計を最初から行うことは避け，既存の NFS プロトコルを利用することにした．NFS はプロトコル仕様が公開されており，多くの Unix 系 OS で利用可能である．そこで，既存の NFS サービスを利用し，3 章で述べた諸問題を解決する機能の追加を行った．これは，システムの開発を容易にすると同時に，高い移植性を得るためである．

ONFS の構成を**図 1** に示す．基本的な共有ファイルシステムの実現には OS カーネルの持つ NFS サービスを利用する．ONFS の提供する機能はユーザプログラムで実現した．ONFS の基本プログラムとして，サーバ側で動作する ONFS サーバプログラムと，クライアント側で動作する ONFS クライアントプログラムがある．これらのプログラムは，サーバ/クライアント間のネゴシエーションに加え，それぞれ OS との間で NFS メッセージを中継する．以降，それぞれのプログラムの機能について説明する．

5.2 ONFS サーバプログラム

サーバは，ネットワークで共有するファイルシステム空間を提供する．ONFS サーバプログラムの主な機能は，クライアントに対するサーバ情報の通知と，クライアントユーザの管理である．

サーバ情報通知

ONFS サーバプログラムはクライアントからのサーバ探索要求に応じて，サーバの情報をクライアントに通知する．サーバの情報は，サーバのホスト名，共有ディレクトリ名とその NFS ファイルハンドル☆である．

ユーザ管理

ONFS サーバプログラムはユーザ管理の作業として，クライアントユーザの登録と，サーバ/クライアント間の UID/GID 変換を行う．クライアントユーザは，サーバの一時的なユーザとして認識され，専用のユーザアカウントが割り当てられる．

クライアントユーザのアクセスパターンとして，(1) サーバが不特定のユーザ向けに公開しているファイルを取得する，(2) サーバにファイルを置いてそこで作業する，という場合を考えた．(1) の場合はすべてのユーザに読み込みのみのアクセスを，(2) の場合は特定のユーザのみ読み書きアクセスを許可することが望ましい．そこで，クライアントユーザのユーザ権限を，匿名ユーザおよび認証ユーザの 2 段階で扱うことにした．

匿名ユーザは未認証のクライアントユーザである．ONFS サーバプログラムは，すべての匿名ユーザに同一の匿名アカウント☆☆を割り当てる．

一方，認証ユーザはパスワード認証済みのクライアントユーザである．認証ユーザは，パスワードを共有するグループ（パスワードグループ）ごとに管理され，サーバ上ではアカウントを必要数用意する．ONFS サーバプログラムは，認証ユーザに対して個別にアカウントを割り当てる．サーバプログラムは割り当てたアカウントの情報（クライアントのホスト名とユーザ名）を保存し，次回同じクライアントユーザを認証したときに前回と同じアカウントを割り当てる．未割当てアカウントが不足した場合は，現在利用されていない割当て済みアカウントを再利用する．

認証ユーザのパスワードは，サーバユーザが事前にクライアントユーザに通知する．パスワードグループは，複数の認証ユーザに同一のパスワードを割り当てるために導入した．それにより，個別にパスワードの設定と配布を行う手間を軽減した．

ONFS サーバプログラムは，NFS メッセージを書き換えることによって，サーバ/クライアント間で対応する UID/GID を変換する．クライアントでは，匿名ユーザはクライアント側の匿名アカウントとして，認証ユーザはクライアントユーザ自身のアカウントとして識別される．ユーザの詐称を防ぐため，UID/GID 変換はすべてサーバプログラムが行う．

認証ユーザに対するアカウント割当ての例を**図 2** に示す．各ユーザアカウントは通常の Unix ユーザアカウントで，ONFS の管理するパスワードグループによってグループ化される．認証ユーザのアカウントは，クライアントユーザが入力したパスワードと一致するパスワードを持つグループから割り当てられる．すなわち，割り当てるアカウントをパスワードによって

☆ NFS ファイルハンドルは NFS プロトコルにおけるファイルの識別子である．

☆☆ 標準では nobody アカウントを用いる．nobody は，NFS において不特定の一般ユーザを表すアカウントとして利用される．

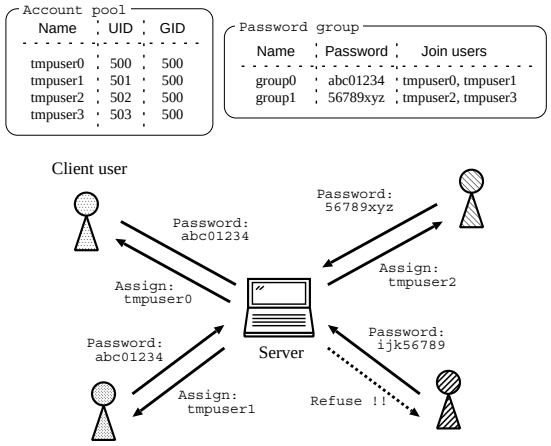

図 2 ユーザアカウントの割当て
Fig. 2 Assignment of user accounts.

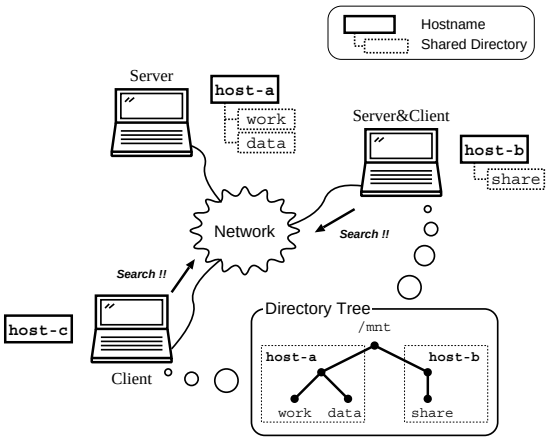

図 3 サーバ探索
Fig. 3 Search file servers.

区別できる．パスワードがどのグループにも一致しなかった場合は，アカウントの割当てを拒否する．

5.3 ONFS クライアントプログラム

クライアントは，サーバの提供するファイルシステム空間を共有利用する．ONFS クライアントプログラムの主な機能は，サーバの探索と自動マウント，サーバに対するクライアントユーザの登録，および切断時の処理である．

サーバ探索と自動マウント

ONFS クライアントプログラムはブロードキャストメッセージを利用して，ネットワーク上のサーバを探索する．サーバは探索要求を受信すると，クライアントに対してサーバ情報を通知する．クライアントプログラムはサーバ情報をもとにして，サーバとそのサーバの提供する共有ディレクトリの一覧を仮想的なファイルシステム空間上に対応づける．この仮想的なファイルシステムに対するアクセスは，ONFS マウントポイント（任意のディレクトリを指定可能）を通じて行われる．クライアントユーザが共有ファイルにアクセスしようとすると，ONFS マウントポイント以下に共有ディレクトリをマウントし，実際にアクセスできるようになる．

本機構によって，ユーザはディレクトリをたどるだけでサーバのファイルにアクセスできる．サーバの探索は，ONFS クライアントプログラムの起動時と，ユーザが ONFS マウントポイントのディレクトリにアクセス（ls コマンドを実行するなど）したときに行われる．サーバ探索の様子を**図 3** に示す．

ユーザ登録

ONFS クライアントプログラムは，サーバに対して匿名アカウントに対応する UID/GID と，クライアントユーザのアカウントに対応する UID/GID を通知する．サーバではこの UID/GID と，NFS メッセージに含まれるアクセス元のクライアントユーザの UID/GID を照合し，サーバ側の UID/GID と変換する．クライアントユーザの UID/GID が認証ユーザのものでなければ，そのユーザは匿名ユーザとして扱う．サーバは UID/GID の照合をクライアントごとに行うため，異なるクライアントが同じ UID/GID を利用していたとしても，サーバは異なるユーザとして識別する．クライアントでは，そのクライアントの認証ユーザが所有するファイルの UID/GID は，クライアントユーザの UID/GID として見える．それ以外のユーザの所有するファイルの UID/GID は，匿名アカウントの UID/GID として見える．

クライアントユーザがサーバに対して認証を行いたい場合は，専用のパスワード入力プログラムを用いてサーバにパスワードを送信する．パスワードは現在の実装では平文で送信しているが，チャレンジ/レスポンス方式によるパスワードの暗号化を行う予定である．

切断時の処理

ONFS クライアントプログラムは，サーバとの接続状態をつねに監視して，切断時に処理を中断し，自動的にアンマウントを行う．切断を判断するタイムアウトは 20 秒とした．

サーバとの接続が切断されると，NFS サーバからのメッセージが返らなくなる．そのため，OS カーネルの NFS クライアントサービスのタイムアウトまで，サーバのファイルにアクセス中のプロセスが長時間ブロックし，アンマウント不能になる問題がある．そこで，ONFS クライアントプログラムは処理の失敗を示すメッセージを NFS サーバの代わりに返信すること

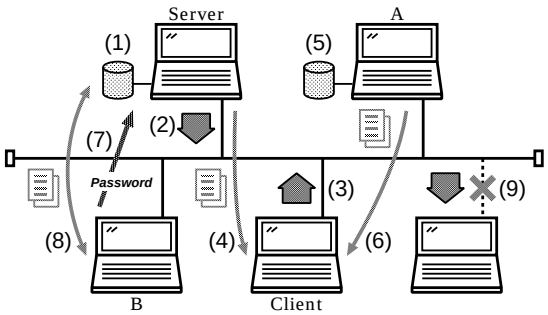

図 4 システム利用の流れ
Fig. 4 Flow of system using.

で，長時間のブロックを回避する．

本機構によって，ユーザはサーバ利用後に計算機をネットワークから切り離すだけでよい．また，不意の切断時もシステムの動作に支障を来さない．

5.4 利用方法

ソース読み会を実際的な例としてあげながら ONFS の利用方法を示す．ソース読み会とは，多人数で議論を交わしながらプログラムのソースコードの精読を行う会合のことである．ソース読み会は主にシステムソフトウェアの研究者の間で行われており，ソフトウェアの動作の理解やプログラミングの学習に役立つ．

ソース読み会ではしばしば携帯型計算機が持ち込まれる．その理由として，ソースコードが計算機上の情報であり，計算機上での閲覧が容易であるとともに，情報の検索性に優れることがあげられる．また，対象のプログラムをその場で実行して動作を確認できるという利点もある．そこで，参加者の計算機でネットワークを構築し ONFS を利用することで円滑な情報交換が可能となる．

ソース読み会で ONFS を用いた場合の流れを**図 4** に示す．この例では，ソース読み会の主催者の計算機をファイルサーバとして利用する．参加者には，共通のパスワードでそれぞれにユーザアカウントを割り当てる．本システムの利用に必要な設定は，システムの導入時に行っているものとする．

(1) 主催者は対象となるプログラムのソースコードやドキュメントのファイルを自分の計算機上に用意し，共有ディレクトリ上に配置する．
(2) 主催者は ONFS サーバプログラムを起動し，ネットワークに接続する．
(3) 参加者はそれぞれ ONFS クライアントプログラムを起動し，ネットワークに接続する．
(4) 参加者は ONFS マウントポイントを通じて，ソースコードに通常のファイル操作でアクセスできる．
(5) ある参加者 A が自分の持つファイルを他の参加者に公開したくなったら，ファイルを共有ディレクトリに置いて，ONFS サーバプログラムを起動する．
(6) 他の参加者のマシンでは，ONFS マウントポイントを通じて参加者 A の提供するファイルにアクセスできる．
(7) ある参加者 B が主催者の共有ディレクトリ上にファイルをコピーしたくなったら，認証プログラムを用いて主催者の計算機にパスワードを送信する．
(8) 主催者の計算機において参加者 B の認証に成功すると，参加者 B はファイルを書き込めるようになる．
(9) 読み会の終了時は，各計算機をネットワークから切り離すだけで共有が自動的に終了される．

以上の操作で簡単にファイルを共有できる．ONFS はネットワーク構築時の設定作業を必要とせずに利用できる．

6. 評価

本章では，NFS および自動マウンタとの比較，ONFS のアクセス性能および切断時の処理に関する実験と，移植性に対する評価を行う．

6.1 NFS との比較

一時的なネットワーク環境で NFS と ONFS を利用するときに必要な設定について，サーバ/クライアント側の双方で比較した．クライアント側では，自動マウンタを用いた場合についての比較も行った．

サーバ側

NFS の利用時に必要なサーバ側の設定を以下に示す．

(1) 共有ディレクトリの指定
(2) アクセスを許可するクライアントの指定
(3) ユーザアカウントの登録

具体的な設定方法は OS によって異なるが，設定ファイルに記述する場合が多い．一時的なネットワーク環境で NFS を利用する場合，(1) は事前に設定することができるが，(2) および (3) はその場でクライアントのアドレスやユーザ情報を調べて設定する必要がある．

一方，ONFS では，その場での設定を不要とした．(2) については，アドレスによる指定ではなく，パスワードによるユーザごとのアクセス制御を導入した．また，(3) については，事前に一時利用のためのアカウントをまとめて登録しておき，その中から動的に割

り当てるようにした．

クライアント側

NFS の利用時，クライアント側では利用時にサーバのアドレスと共有ディレクトリを指定してマウントする作業が必要である．また，利用終了時にはアンマウントする作業が必要である．

NFS には，マウントおよびアンマウント作業を自動化する自動マウンタという機構がある．自動マウンタを用いると，ユーザのアクセス時に自動的にマウントを行ったり，一定時間アクセスがない場合に自動的にアンマウントしたりすることが可能である．

しかし，自動マウンタを用いてマウントする場合でも，利用するサーバの名前かアドレスをユーザが指定する必要がある．ONFS では，サーバ一覧が利用時に取得できるため，アドレスを調べてマウントする必要がない．

また，NFS ではマウント中にサーバとの通信切断が発生するとアンマウントできなくなり，クライアントの動作に支障を与えてしまう問題がある．これは，自動マウンタを利用した場合も同様である．ソフトマウントを用いた場合でも，タイムアウトまでに長い時間を必要とする．ONFS は 20 秒で通信切断を判断しエラーを返すため，短時間でアンマウントすることが可能である．

6.2 実験環境

本章における実験環境について説明する．計算機の仕様を**表 1** に示す．計算機の形態はサーバ/クライアントともに携帯型である．計算機の接続にはスイッチングハブを使用した．NFS のプロトコルはバージョン 2，トランスポートプロトコルは UDP を用いた☆

6.3 アクセス性能

ONFS ではユーザプロセスによる NFS パケットの中継と，その書換えを行っているため，共有ファイルに対するアクセスにオーバヘッドが生じる．そこで，iozone ベンチマークプログラムを用いて I/O 性能の測定を行った．iozone はシーケンシャルデータの書き込み/読み込みを行い，その所要時間からスループットを計測するプログラムである．スループットの計測は，通常の NFS を利用した場合と ONFS を利用した場合について，ファイルサイズを 1 M バイトから 256 M バイトまで変化させながら行った．

結果を**図 5** に示す．通常の NFS に対する ONFS のスループットは，読み込みが約 90%，書き込みが約 70%であった．この結果から，性能低下は小さいことが分かった．ファイルサイズが 64 M バイト以上になると読み込みに若干の性能低下が見られた．これは，サーバにおいてバッファに利用可能な物理メモリのサイズよりもファイルサイズが大きくなり，書き込み時に確保されたバッファの内容が読み込み時にはすでに無効になっていたためと考えられる．

☆ ONFS クライアントプログラムと nfsiod プログラムを併用すると，再送の多発による性能低下が見られたため，本実験は nfsiod を用いずに行った．nfsiod は，複数の NFS リクエストを並行処理するためのシステムプログラムである．

表 1 実験に用いた計算機の仕様
Table 1 Specifications of computers for the experiment.

	Server	Client
CPU	Intel Celeron 266 MHz	Intel Pentium/P55C 200 MHz
Memory	64 MBytes	48 MBytes
Disk	6 GBytes	5 GBytes
Network	10 BASE–T	10 BASE–T
OS	FreeBSD 3.4-RELEASE	FreeBSD 2.2.8-STABLE

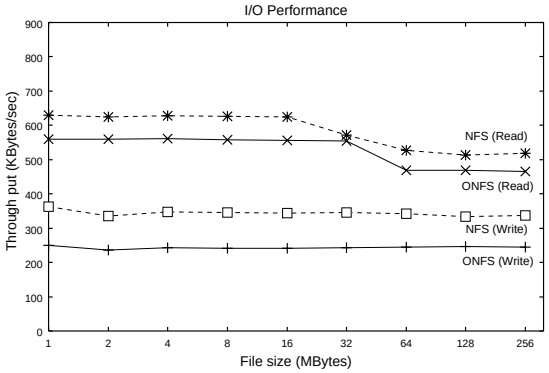

図 5 I/O 性能
Fig. 5 I/O performance.

6.4 切断時の処理

5 章で述べたように，クライアントがサーバとの通信切断を判断するタイムアウトは 20 秒である．この時間は，ONFS クライアントプログラムが OS カーネルの NFS クライアントサービスに対してエラーを返すようになるまでの時間である．実際にアクセス中のプロセスに対してエラーが返るまでには，さらに時間を要する可能性がある．それまでこのプロセスの実行がブロックされるため，計算機の利用性を損なうことになる．

そこで，サーバとの通信切断からプロセスに制御がもどるまでに要する時間の測定を行った．まず，サーバに 1 M バイトのファイルを用意し，クライアントの読み込みプロセスがそのファイルを 8 K バイトずつ読み込む．読み込みプロセスがファイルを 512 K バイト読み込んだ時点で，クライアントのネットワークイン

表 2 ソースコードの行数
Table 2 Number of source code lines.

OS independent part (Total)	11501
Common	3456
Server	3020
Client	4747
Other commands	278
OS dependent part	—
FreeBSD 2.2.x/3.x/4.x	154
Linux kernel 2.2.x	120
Solaris 2.x	123

タフェースを無効化する．それから，8 K バイトを読み込もうと試みて読み込みプロセスにエラーが返るのを待つ．ここで，ネットワークインタフェースを無効化してからエラーが返るまでの時間を測定した．OS によるバッファリングなどの影響を防ぐため，ONFS クライアントプログラムは測定ごとに再起動を行った．

この測定を 100 回行った結果，平均時間は 22.3 秒，標準偏差は 1.23 であった．この結果タイムアウトの 20 秒に加え，さらに 2 秒程度の時間で回復することが分かった．ONFS クライアントプログラムから NFS クライアントサービスに対してエラーが返されるのは，タイムアウト後に到着した再送パケットに対してであり，タイムアウト直後に返されるのではない．そのため，再送のタイミングによってタイムアウトから数秒の遅れが発生すると考えられる．

6.5 移 植 性

ONFS の開発は FreeBSD 2.2.8 上で行った．また，ONFS のコンパイルおよび動作の確認は FreeBSD 3.x/4.x，Linux kernel 2.2.x，Solaris 2.x で行っている．**表 2** に ONFS の各部分におけるソースコードの行数を示す．

OS に依存するのは，mount システムコールを発行してファイルシステムのマウントを行う部分であり，それぞれの OS において百数十行程度で実装できた．それ以外の部分は，OS 実装の違いに起因する小さな差異を吸収するための修正を数カ所で行った程度でコンパイルできた．このことより，Unix 系 OS であればマウント用のコードを記述するだけで ONFS を移植できることが分かった．

7. 考 察

本章では，ONFS の設計と実装について考察を行う．

7.1 サーバ探索

ONFS ではブロードキャストメッセージを利用してサーバ探索を行っている．そのため，クライアントユーザは計算機をネットワークに接続するだけで，すぐにサーバにアクセスできる．

ネットワーク上の探索可能なサーバは，クライアントと同一セグメント上のものに限られる．しかし，ONFS は単一セグメントで構成可能な規模のネットワークでの利用を想定しているため，単純なブロードキャストで対応した．もし複数のセグメントを利用する場合でも，メッセージ中継のための機構を用意するか，マルチキャストメッセージを利用することで対応できる．現状では，サーバのアドレスを直接指定してアクセスするための補助コマンドを用意することで，セグメントを越えたサーバの利用に対応している．

7.2 ユーザ管理

ONFS のユーザ管理機構は，サーバにおけるユーザ管理を自動化するとともに，サーバ/クライアント間におけるファイル所有者情報の整合をとる．そのため，ユーザは ONFS の利用時にユーザ管理を意識する必要がない．

ONFS ではパスワードを用いたユーザの認証を行っており，最小限のセキュリティを確保している．ネットワークは単一セグメントで構成されるため，通信の傍受が可能であるが，通信路の暗号化などの積極的なセキュリティの確保は行っていない．規模の小さい会議では参加者を把握できるため，実用上は問題ないと考えられる．しかし，規模が大きくなると参加者をすべて把握することは困難となる．通信の安全性が必要な場合には，ネットワーク層またはトランスポート層の暗号化で対応する必要がある．

7.3 切断時の処理

ONFS クライアントプログラムは，サーバとの接続の切断を検出し自動的に終了作業を行う．そのため，ユーザは明示的に利用終了の作業を行う必要がない．

サーバ/クライアント間におけるファイルの一貫性の維持は NFS プロトコルに依存する．NFS の実装にはクライアントキャッシュを持つものがあるが，ONFS はサーバとの同期制御は行わない．その理由は，接続が切断されたときにはすでにサーバとの通信が不能になっていることと，再接続時にサーバのファイルシステムの状態が変わっている可能性があるからである．したがって，書き込み中や，未書き込みのキャッシュデータがあるときに切断が発生すると，ファイルの一貫性が保たれない問題がある．

7.4 移 植 性

現在の ONFS の実装では共有ファイルシステムの実現に NFS を利用している．そのため，NFS サービスを持つ Unix 系 OS であれば，ファイルシステムのマウントを行うコードを実装するだけで対応できる．

その反面，NFS に対する依存が大きいため，NFS サービスを持たない OS での利用は困難である．より広範囲の OS に対応するためには，共有ファイルシステムのプロトコルと，ONFS の持つサーバ探索，ユーザ管理などのプロトコルを分離する必要がある．

8. おわりに

本稿では，一時的なネットワーク環境で即座に利用できる共有ファイルシステム ONFS の設計と実装について論じた．ONFS を用いることで，会議など人の集まる場において簡単に透過的なファイル共有を実現できる．

既存の共有ファイルシステムを一時的なネットワーク環境で利用するうえでの問題点として，計算機の特定，ユーザの識別，切断時の対応をあげた．ONFS では，これらの要求に基づきシステムの設計を行った．

ONFS は共有ファイルシステムのプロトコルとして移植性の高い NFS を利用し，サーバ探索，ユーザ管理，切断時処理の自動化のための機構を実装した．これらの機構により，ユーザは利用時の繁雑な設定作業から解放される．また，性能評価により ONFS の処理によるオーバヘッドは十分に小さく，NFS と遜色がないことを示した．また，切断時の復旧も短時間で行われることを示した．NFS は広く利用されており，ONFS が NFS と同様の感覚で利用できることは，共有ファイルシステムとして意義がある．

今後は，ONFS から NFS への依存部分を分離するとともに，一時的なネットワーク環境で利用できる汎用的なサーバ探索やユーザ管理機構の実現を目指す．

謝辞 システムの開発にあたっては，rover 研究グループの方々に多くの助言をいただきました．ここに感謝の意を表します．

参考文献

1) 倉島顕尚，前野和俊，市村重博，田頭 繁，武次將徳，永田善紀：集まったその場での共同作業を支援するモバイルグループウェア「なかよし」，情報処理学会論文誌，Vol.40, No.5, pp.2487–2496 (1999).
2) Callaghan, B.: *NFS Illustrated*, Addison-Wesley (2000).
3) Aggarwal, A., Agrawal, A., et al.: Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods, RFC1001 (1987).
4) Aggarwal, A., Agrawal, A., et al.: Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications, RFC1002 (1987).
5) Sun Microsystems: NFS: Network File System Protocol Specification, RFC1094 (1989).
6) Callaghan, B., Pawlowski, B. and Staubach, P.: NFS Version 3 Protocol Specification, RFC1813 (1995).
7) Shepler, S., Callaghan, B., et al.: NFS Version 4 Protocol, RFC3010 (2000).
8) Mockapetris, P.: Domain Names — Concepts and Facilities, RFC1034 (1987).
9) Samba Team: Samba Web Pages. http://www.samba.org/
10) Eckstein, R., Collier-Brown, D. and Kelly, P.: *Using Samba*, O'Reilly & Associates (1999).
11) Objective Development: Sharity — an SMB/CIFS Client for Unix. http://www.obdev.at/Products/Sharity.html
12) Alexandrov, A.D., Ibel, M., Schauser, K.E. and Scheiman, C.J.: Extending the Operating System at the User Level: the Ufo Global File System, *Proc. USENIX 1997 Annual Technical Conference*, pp.77–90 (1997).
13) Kistler, J.J. and Satyanarayanan, M.: Disconnected Operation in the Coda File System, *Proc. 13th ACM Symposium on Operating Systems Principles*, pp.213–225 (1991).
14) 楯岡孝道，植原啓介，砂原秀樹，寺岡文男：PFS：通信環境に動的に適応するファイルシステム，コンピュータソフトウェア，Vol.15, No.2, pp.62–81 (1998).

(平成 13 年 11 月 8 日受付)
(平成 14 年 12 月 3 日採録)

福田 伸彦（学生会員）

1999 年電気通信大学電気通信学部電子情報学科卒業．2001 年同大学大学院情報システム学研究科博士前期課程修了．現在，同大学院博士後期課程に在学．システムソフトウェアおよびネットワークに興味を持つ．

楯岡 孝道（正会員）

1996 年電気通信大学大学院情報工学専攻博士前期課程修了．1999 年同大学院情報システム設計学専攻博士後期課程単位取得退学．2000 年同大学電気通信学部助手．博士（工学）．モバイルコンピューティングおよびコンピュータネットワークに興味を持つ．電子情報通信学会，日本ソフトウェア科学会各会員．

中村 嘉志（正会員）

1971 年生．1994 年神奈川大学理学部情報科学科卒業．1996 年電気通信大学大学院情報システム学研究科博士前期課程修了．1997 年同専攻博士後期課程退学．同年同研究科助手を経て，現在産業技術総合研究所特別研究員．分散システムの研究に従事し，現在情報支援システムに興味を持つ．

多田 好克（正会員）

1985 年東京大学大学院工学系研究科情報工学専門課程博士課程修了．工学博士．同年電気通信大学電子情報学科着任．1992 年より電気通信大学大学院情報システム学研究科．並列・分散システムの記述法に興味を持ち，オペレーティングシステムをはじめとするシステムソフトウェアの実現法に関する研究に従事．ACM，電子情報通信学会会員．