Internetにおける資源アクセス装置の提案

門林雄基[†] 山口英[††] 宮原秀夫[†]

Internetが商用化され，ネットワークに関する専門知識を持たない一般ユーザによる広域ネットワークの利用が進むにつれて，セキュリティをはじめとする様々な問題が表面化している．今日のInternetでは，このような一般ユーザからのアクセスを支援する機構が求められている．本論文では，広域ネットワークにおける資源アクセスの複雑な部分を隠蔽し，LAN環境における資源と同様に簡単に扱うことを可能とする「アクセス装置」を提案する．NFSに基づく分散ファイルシステム上にアクセス装置を実現し，実装に基づいてアクセス装置の有効性を確認した．

A Mechanism for Simplifying Access to Internet Resources

YOUKI KADOBAYASHI,[†] SUGURU YAMAGUCHI[††] and HIDEO MIYAHARA[†]

The commercialization of the Internet promoted the use of wide area networks by naive users without networking expertise, which exposed various problems, typically in the security aspects. In today's Internet, some mechanisms are needed to solve these problems by supporting naive users. This paper proposes "access engine", which hides complex access procedures in wide area networks, thereby enabling simple access to resources in wide area networks. We implemented access engine in our NFS-based distributed filesystem, through which we confirmed its effectiveness.

1. はじめに

計算機の相互接続による広域ネットワークInternetの商用化にともない，一般ユーザを中心とした広域ネットワークの利用がさかんになりつつある．ところが，Internetは元来学術ネットワークとして発達してきたため，ネットワークに関する専門知識を持たない一般ユーザに対する配慮が十分でない．このため，一般ユーザが広域ネットワークにおける複雑なアクセス手順に従わねばならず，このことが様々な問題の原因となっている．

従来のネットワークアプリケーションでは，広域ネットワーク上の資源アクセスにおいて，資源の位置情報をユーザに求める方式がとられている．広域ネットワークの構成情報などの専門知識を持たない一般ユーザがこのようなアプリケーションを用いた場合，様々な問題が起きる．具体的には，悪意を持った組織を判別することができないため，トロイの木馬プログラムを実行してしまうことによるセキュリティ問題が起こる可能性がある．また，ネットワーク構成情報を知らずに資源アクセスを行うため，遠方のサーバへのアクセスによる通信コスト増大の恐れがある．同様に，代替サーバが用意されている場合においても，サーバの障害に対処できない可能性がある．次に，広域ネットワーク上の資源に対する多様なアクセス方法に対応できず，様々なツールを使い分けることができない可能性がある．

このような問題を解決するために，本研究では広域ネットワークにおける資源アクセスの複雑な部分を隠蔽し，LAN環境における資源と同様に簡単に扱うことを可能とするような「アクセス装置」を考える．アクセス装置では以下のようなことを目標とする．代替サーバや地理情報に基づくサーバ選択といった広域ネットワーク特有のセマンティクスをユーザから隠蔽し，アクセスを簡単化する．次に，多様なアクセス方法を隠蔽し，LAN環境における資源アクセスと同様に透過的なアクセスを行うことができるようにする．また，キャッシュなどの手法を用いてアクセス時間を短縮し，円滑なアクセスを行うことができるようにす

† 大阪大学基礎工学部情報工学科
Faculty of Engineering Science, Osaka University
†† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

る．さらに，ネットワーク構成や課金体系を考慮した資源アクセスを行うことで，アクセスにかかるコストを軽減できるようにする．

本研究ではアクセス装置を分散ファイルシステムNFS上に実現し，実装に基づいて評価を行うことでアクセス装置の有効性を確認した．

2. 広域網における資源アクセスの問題点

従来のシステムにおいて，広域ネットワーク上の資源に対して適切なアクセスを行うためには，LAN環境におけるアクセスと比べ，様々な知識が必要となる．専門的な知識を持たない一般ユーザが従来のシステムを用いてアクセスした場合，以下にあげるような問題が起きる．

セキュリティの問題：信頼できる組織とそうでない組織を見分けることができない一般ユーザにとって，悪意を持った組織が作成したトロイの木馬プログラム[1),2)]を見分けることは困難である．したがって，広域ネットワーク上のあらゆる資源に直接アクセスすることが可能な従来のシステムを一般ユーザが利用した場合，トロイの木馬プログラムを誤って実行してしまうことによるセキュリティ問題が起きる可能性がある．

コストの問題：従来のシステムでは広域ネットワーク上の資源を指定するときに，ホスト名やドメイン名を明示的に指定する方式がとられている．このような資源の位置情報をユーザに求める方式では，広域ネットワーク上に複数配置されたサーバを効率的に利用することができず，コストが増大する．具体的には，最適配置戦略に基づいて配置された近傍のサーバにおいて遠方のサーバと同一の資源が提供されていた場合，ネットワーク構成を知らない一般ユーザが明示的に遠方のサーバを指定してしまい，アクセス時間が悪化する場合がある．また，通信路において距離と転送量に基づく課金体系が採用されている場合，遠方のサーバへのアクセスによる通信コスト増大の恐れがある．

障害の問題：耐故障性を考慮して等価な資源を提供するサーバが複数用意されている場合においても，一般ユーザはネットワーク構成に関する知識がないため代替サーバを利用できない可能性がある．

一様でないインタフェースの問題：広域ネットワーク上では様々なオペレーティングシステムやサーバソフトウェアが用いられているため，それらの上で提供される資源に対するアクセス方法も異なる．一般的なユーザがこれらの違いを理解し，様々なツールを使い分けることは困難であると考えられる．

ポリシーの問題：Internetでは負荷分散や耐故障性などを考慮し，サーバを相互にバックアップするといったことが行われているが，災害時やネットワーク障害発生時を除けば，他組織に用意された代替サーバへのアクセスは歓迎されず，自組織のサーバを利用することが奨励される．このようなポリシーを一般ユーザに説明し，遵守を求めるのは困難であると考えられる．

3. アクセス装置

本研究では，前章で述べたような広域ネットワーク上の資源アクセスにおける問題点を解決することをめざす．具体的には，広域ネットワークにおける資源アクセスの複雑な部分を隠蔽し，LAN環境における資源と同様に簡単に扱うことを可能とするような機構を考える．以下では，このような装置をアクセス装置と呼ぶ．

アクセス装置に期待される効果は以下のようであると考えられる．

E1. 広域セマンティクスの隠蔽： サーバの最適配置戦略の利用や，代替サーバへの切替え，サーバの利用ポリシといった広域ネットワーク特有のセマンティクスをユーザから隠蔽し，アクセスを簡単化する．

E2. インタフェースの連続性： 広域ネットワーク上の資源に対するアクセス手順をLAN環境におけるアクセス手順と同一なものとし，LAN環境における資源アクセスと同様に透過的なアクセスを行うことができるようにする．

E3. アクセス時間の短縮： 一般にLAN環境と比較して遅延の分散が大きい広域ネットワークにおいて，アクセス時間を短縮することでLAN環境と同様に円滑なアクセスを行うことができるようにする．

E4. アクセスコストの軽減： ネットワーク構成や課金体系を考慮した資源アクセスを行うことで，アクセスにかかるコストを軽減する．

アクセス装置を用いることで，広域ネットワーク上の資源はLAN環境における仮想的な資源として見ることができる（**図1**）．

E1～E4であげたような効果を得るためには，資源は位置独立性（location independence）と位置透過性（location transparency）を備えていなければならない．位置独立性とは，資源の物理的な位置が変わっても資源名を変更する必要がないことをいう．また，位置透過性とは，資源名に物理的な位置がいっさい含まれていないことをいう．

アクセス装置はサーバとクライアントの間に位置す

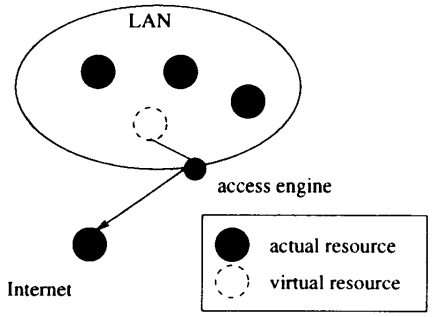

図1 アクセス装置
Fig. 1 Access engine.

るが，極端な場合としてアプリケーションにアクセス装置の機能を埋め込むことも考えられる．しかしながら，この方式ではサーバ選択アルゴリズムやキャッシュアルゴリズムを変更するためにはすべてのアプリケーションを変更しなければならず，アルゴリズム改善の効果を得ることが難しい．なお，サーバにおいてアクセス装置を実装した場合，障害時における代替サーバへの切替えやアクセス時間の短縮といった効果が実現できないため非現実的である．

4. アクセス装置の実現

本研究ではアクセス装置を分散ファイルシステム[3]上に実現することを考える．多くのアプリケーションは分散ファイルシステム上に構築されているため，分散ファイルシステムに対してアクセス装置を付加し，従来LAN環境で用いられてきた分散ファイルシステムを広域ネットワークへ拡張することで，広範囲にわたってE1～E4で述べたような効果が得られると考えられる．

なお，広域ネットワークにおいて運用可能な分散ファイルシステムとしてAFS[4]があるが，AFSでは図2に示すようにパス名に位置情報が含まれており，位置透過性および位置独立性を備えていないため，E1，E4に示した効果を得ることができない．

このような理由から，本研究では分散ファイルシステムWWFS[5),6)]においてE1～E4で述べたような効果を得ることを目標としたアクセス装置csdを設計，実装した．csdをNFSサーバとして実現することで，インタフェースの連続性が得られると考えられる．また，csdにおいてボリュームを基本としたファイルの名前付けを行うことで，広域ネットワークにおける様々なセマンティクスを隠蔽することができる．さらに，ファイルやディレクトリ情報をキャッシュすることでアクセス時間が短縮でき，位置透過性および位置独立性を備えた名前付け機構と組み合わせることでアクセスコストを軽減することができると考えられる．

```
/afs/ibm.uk/public/doc/afs.faq
/afs/transarc.com/public/afs-contrib
```

図2 AFSにおけるパス名の例
Fig. 2 An example of AFS pathnames.

以下ではcsdの実現方式について述べる．また，広域ネットワークにおいて分散ファイルシステムを実現する際の問題点についても触れ，csdにおいてそれらをどのように解決したかについても述べる．

4.1 クライアントからのアクセス方式

現在LAN環境で用いられている多くのファイルシステムは，vnodeインタフェース[7]と呼ばれるファイルシステムインタフェースを採用している．vnodeインタフェースはopen，read，write，closeなどの操作からなる．既存の多くのアプリケーションはvnodeインタフェースを基本としたファイルシステム上に実現されているため，広範囲にわたってアクセス装置の効果を得るためにはvnodeインタフェースを持つアクセス装置が必要である．csdは，vnodeインタフェースに基づいて設計された分散ファイルシステムNFS[8]のサーバとして振る舞う．NFSクライアント機能は多くのオペレーティングシステムにおいて実装されているため，アクセス装置を広い範囲において適用することができる．また，csdをNFSサーバとして実現することで，ネットワーク透過なファイルシステムを前提として実装された従来のアプリケーションをそのまま用いることができる．

4.2 ファイルサーバへのアクセス方式

広域ネットワーク上のファイルサーバに対してアクセスするためにNFSを用いることも可能であるが，NFSでは長距離の通信路における遅延特性を考慮していないためプロトコル効率が悪く，アクセス時間がきわめて悪化する．この問題を解決するために，本研究ではLAN環境においてNFSを用い，広域ネットワークへのファイルアクセスでは遅延特性を考慮したアクセスプロトコルを用いる．現在Internetに多く存在するファイルサーバにおいてアクセス時間の改善効果を得るために，アクセスプロトコルとしてFTP[9]を用いた．現在の実装ではNFSとFTPの相互変換のみを行っているが，csdでは網の特性にあわせた複数のプロトコル間の相互変換を考慮し，図3に示すようなプロトコル変換器を実現した．NFS-FTPプロトコル変換の詳細については5章において説明する．

4.3 ファイルの名前付け

本研究ではファイルシステムの一部分を複製し，広

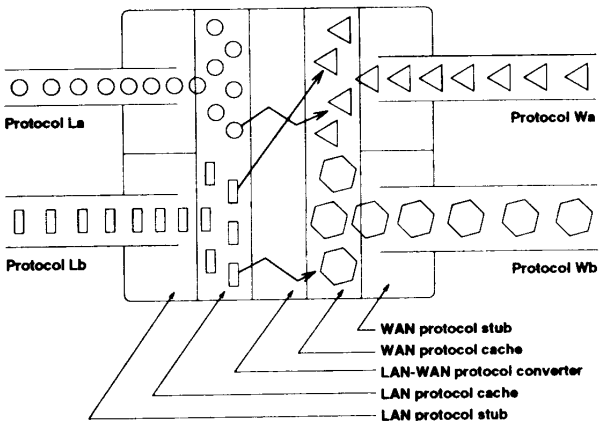

図3 プロトコル変換器
Fig. 3 Protocol converter.

```
ftp-server:      ftp.jaist.ac.jp
ftp-directory:   pub/RFC
ftp-server:      ftp.iij.ad.jp
ftp-directory:   pub/RFC

ftp-server:      ds.internic.net
ftp-directory:   rfc
ftp-server:      ftp.germany.eu.net
ftp-directory:   pub/documents/rfc

description:     Request For Comments
```

図4 ボリューム記述ファイルの例
Fig. 4 An example of volume description file.

域ネットワーク上に分散配置したものをボリュームと呼ぶ．ボリュームは，(ホスト名，該当するファイルシステムまでのパス名) のリストで構成される．csd におけるボリュームの記述ファイルを**図4**に示す．csd ではボリュームに属するすべてのファイルシステムを等価な内容を持つものとして扱い，サーバ選択の過程をユーザから隠蔽する．このような隠蔽を行うことで次のような効果が得られる．いずれかのサーバにおいて障害が発生したときに，代替サーバに切り替えることができる．また，サーバの最適配置戦略を利用し，ネットワーク構成を考慮したサーバ選択を行うことができる．

ボリュームは csd の管理者によって設定され，ボリュームを設定していないファイルシステムに対してはアクセスできない．このように，ネットワーク上のすべてのファイルに対して無制限にアクセスを許可するのではなく，必要とされるファイルに対してのみアクセスを許可することで，トロイの木馬プログラムを一般ユーザが実行してしまう可能性を減らすことができると考えられる．

名前空間の共有方式としては，ユーザごとに名前空間を構築し，ユーザ間で互いの名前空間にアクセスする方式[10]と，ユーザ間で共通の名前空間を持つ方式が考えられる．本研究では広域ネットワーク上にある共有ファイルを対象としているため，ユーザ間で共通の名前空間を持つ方式を採用した．csd が提供する名前空間は各ボリュームの名前空間と，それらを結合する名前空間からなる．各ボリュームの名前空間はファイルサーバ上の該当するファイルシステムの名前空間と同一である．csd が提供する名前空間は各計算機上のマウントポイントと呼ばれるディレクトリに結合される．一例を**図5**に示す．

```
$ ls /ww/fs
reeBSD/    NetBSD/      tf/
IMR/        RFC/         internet-drafts/
INDE
X       fj.sources/  isoc/

$ ls /ww/fs/ietf
00README                        ipngwg/
0mtg-agenda.txt                 ipsec/
0mtg-at-a-glance-95dec.txt      ipv6mib/
0mtg-at-a-glance-96mar.txt      isdnmib/
...
```

図5 csd が提供する名前空間の例
Fig. 5 An example of namespace served by csd.

4.4 共有のセマンティクス

分散ファイルシステムでは，共有ファイルを複数のクライアントが同時にアクセスしたときの振舞いを規定しなければならない．特に，複数のクライアントによってファイルが変更されたときに，そのような書き込み操作の衝突によるファイルの破壊を防ぐ必要がある．また，各々のユーザがファイルに対して加えた変更が失われないよう，複数のバージョンを作ることで書き込み操作の衝突を回避する，あるいはユーザに対して書き込み操作が無効になった旨を通知し，ユーザ間で書き込み操作の衝突を解消する，といった手法が適用されることが多い．また，ファイル読み出し時におけるファイル内容の一貫性を保証するためには，ファイル読み出し時においてもファイルをロックする必要がある．このようなファイル内容の一貫性を保証するための操作を，以下では一貫性制御と呼ぶ．

一貫性制御は LAN 環境では有効であるが，これを広域ネットワークに拡張した場合，アクセス時間が著しく増大すると考えられる．なぜならば，クライアント間の一貫性制御のためには少なくとも三相からなるハンドシェイクが必要であり，広域ネットワークでは遅延が大きいため LAN 環境と比較してそのようなハンドシェイクにかかる時間が長いからである．

本研究ではファイルの読み出し時における広域ネットワークの複雑なアクセス機構をユーザから隠蔽し，アクセスを最適化することを目的としているため，csd ではファイルの書き込みを支援しない．したがって，csd では書き込み操作の衝突回避および衝突解消によってアクセス時間が悪化することはない．

本研究ではファイルに対する変更はファイルサーバに対して直接行うものとした．ファイルサーバでは，衝突回避のための仕組みとしてロックを利用することが可能である．ファイルが変更された場合，キャッシュを破棄する必要が生じる．これについては後述する．

4.5 キャッシュ

csd ではアクセス時間を短縮するため，ファイルおよびディレクトリ情報をディスク上のキャッシュに蓄える．キャッシュ方式としてはこのほか，クライアントにおいてキャッシュを行う方式[4] も考えられるが，ここではディスクレスワークステーションやディスク容量の限られた携帯端末からの利用を考慮し，csd においてキャッシュを行う方式を採用した．

キャッシュをメモリ上に置くことも可能であるが，本研究でキャッシュの対象となるファイルサーバは広域ネットワーク上で提供される大容量なものであり，総容量が 1 Tbytes を超えることから，キャッシュヒット率の改善を目的としてディスク上にキャッシュを置く方式をとった．LAN 環境では，ディスク上の小さなキャッシュ（64～512 Kbytes）によってサーバに対するトラヒックが 60%から 90%程度減らせることが Blaze らのシミュレーションに基づく評価によって明らかにされている[11]．

csd ではキャッシュ管理を簡単化するため，ファイル全体をキャッシュする方式をとった．この方式では，ファイルの一部分しか必要としないアクセス要求に対してはアクセス遅延の増加ならびにトラヒックの増加をまねくが，その影響は小さいと考えられる．Baker ら[12]はファイルシステムのアクセスパターンを観測し，書き込みをともなわないファイルアクセスの 78%はファイル全体に対するアクセスであると報告している．

4.6 一貫性の制御

csd ではクライアントに対してサーバと同一内容のファイルを提供する必要があるため，キャッシュの一貫性制御を行う必要がある．csd では書き込みを支援しないため，一貫性を保つために必要となる作業はキャッシュの破棄だけである．ファイルサーバにおいて更新通知サーバ[13]が動作している場合，ファイルあるいはディレクトリが変更されたときに，すべての csd に対して更新されたパス名が通知される．csd では，これを用いてキャッシュを破棄することができる．このほか，csd では従来の FTP サーバとの互換性を保つため FTP プロトコルを用いたヒューリスティクスに基づいた一貫性制御を行う．具体的には，前回のアクセス時とのディレクトリ情報を比較し，ファイルサイズが変化した場合はファイルキャッシュを破棄する．また，サブディレクトリがなくなった場合はディレクトリキャッシュを破棄する．

4.7 負荷分散

csd においてキャッシュを行うことで，サーバへの負荷を軽減することができると考えられる．しかしながら，クライアント数が多い場合には csd が過負荷となることが考えられる．このような場合には複数の csd を用意し，クライアントを分割する手法が有効であると考えられる．このとき，キャッシュヒット率の低下を防ぐため，csd を木構造に階層化し，複数の csd からのアクセスを集約することでキャッシュのヒット率を改善する手法が考えられる．しかしながら，この方式ではクライアント・サーバ間に介在する csd の数が増えるため信頼性が低下する，キャッシュの一貫性制御が複雑になるといった問題がある．また，Muntz ら[14]はシミュレーションを用いてサーバを階層化したときの効果を評価し，AFS クライアントにおいて 20 Mbytes のキャッシュを行った場合で 10～20%，80 Mbytes のキャッシュを行った場合には 7%の改善しかみられないと報告している．このようなことから，本研究ではサーバを階層化せず，適正な数のクライアントに対して 1 つの csd を用いる．

csd への負荷集中は大きな問題にならないと考えられる．LAN の帯域幅を B_L，WAN の帯域幅を B_W とすると，csd に対する負荷は最大 B_L+B_W，NFS サーバに対する負荷は最大 B_L である．csd は NFS サーバと同様の構成をとっているため，広域ネットワークの現状（$B_L > B_W$）を考慮すると csd に対してかかる負荷は NFS サーバと同程度であると言える．

4.8 サーバ選択

csd ではネットワーク障害を考慮したサーバ選択を行うことで広域ネットワークが切断された場合におけるサービスの可用性を改善することができる．また，バックボーンプロバイダおよび地理情報をネームサーバに登録することで，それらを考慮したサーバ選択を行うことができる．

バックボーンプロバイダおよび地理情報はサブネットワークに対して定義される．それらの情報は，**図 6** に示すようなレコード（GTR レコード）によってネームサーバのデータベース内に表現される．

```
0.0.221.163.in-addr.arpa.   TXT   "GTR=Nara-WIDE-JP"
0.0.1.133.in-addr.arpa.     PTR   osakau-net.osaka-u.ac.jp.
                            TXT   "GTR=Osaka-SINET-JP"
```

図6 バックボーンおよび地理情報の表現
Fig. 6 Representation of backbone and geographical information.

```
region  Tokyo-WIDE-JP
region  Tokyo-IIJ-JP
region  Osaka-WIDE-JP
region  -WIDE-JP
region  -JP
```

図7 優先順位の設定ファイル
Fig. 7 Priority configuration file.

このようなレコードを元にして，サーバの最適配置戦略を利用することができると考えられる．具体的には，同一バックボーンに所属するサブネットワークにおいて複製ファイルサーバが用意されている場合，それを用いることでアクセスの最適化を行うことができる．また，同一地域において複数のバックボーンが相互接続されている場合，異なる地域にあるファイルサーバを用いた場合と比べて，同一地域にあるファイルサーバを用いたほうがアクセス時間が短くなるといった状況も考えられる．csdの管理者はネットワークの構成情報を元にして，このような優先順位を設定する．設定ファイルの例を図7に示す．

csdではGTRレコードと優先順位の設定ファイルに基づいてサーバ選択を行う．サーバ選択の手順は以下のようである．あるボリュームについてサーバを選択するとき，ボリュームを提供するすべてのサーバについてDNSからGTRレコードを得る．それぞれのGTRレコードについて優先順位のパターンとの部分文字列一致を行い，優先順位の高い順にサーバを並べ，先頭のサーバを用いる．

5. 実　　装

csdはC言語で実装されており，約14,000行からなる．csdはNFSプロトコルのサーバとして機能するため，オペレーティングシステムのカーネルに組み込む方式が適切であると考えられるが，本研究では多くのオペレーティングシステム上での動作を目標とし，ユーザ空間で動作するプログラムとして実現した．csdはカーネル内のNFSサーバとは異なるUDPポートにおいてサービスを受付ける．このため，NFSサーバとの共存が可能である．

クライアントとcsdの結合はmountシステムコールを用いて行うが，既存のmountプログラムではUDPポート番号を指定できないため，wwmount，wwu-mountプログラムを別途用意した．mountシステムコールが完了した後，csdが提供するファイルシステムはLAN環境におけるNFSファイルシステムと同様に扱うことができる．

広域ネットワークでは，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いため，広域ネットワーク上のファイルサーバへのアクセスはLAN環境と比べて遅い．このような状況では，NFSプロトコルはread1要求の再送を一定間隔で繰り返すため，csdに対する負荷がかかり，無駄なトラヒックが増加する．このような問題を避けるため，csdではread要求を1秒以内に満たせない場合にはエラーを返す．同時に，csdではFTPを用いてファイル転送を開始するため，数秒後にread要求が来た場合にはアクセスが成功する．

しかしながら，このようなアクセスセマンティクスでは必ずread要求が成功することを前提として作られた既存のアプリケーションを動作させることができない．このような場合にはUIPと呼ばれるアクセスプロトコルを用いる．UIPによって明示的にファイル転送を要求し，ファイル転送の完了を待つことで従来のアクセスセマンティクスを実現することができる．我々はこのような操作を行う関数群をlibwwwライブラリとして用意した．libwwwは，libcシェアードライブラリに埋め込む方法と，コンパイル時に指定する方法のいずれかを用いることによって利用でき，アプリケーションのソースコードを変更する必要がない．

csdでNSとFTPのプロトコル変換機能を実装している．NFSはコネクションレス型で，状態を持たない（stateless）プロトコルであり，一方FTPはコネクション指向型で，状態を持つ（stateful）プロトコルであるから，これらを変換することは困難であると考えられる．このため，本研究ではcsdをイベント駆動型のプログラムとし，それぞれのコネクション指向型プロトコルについて，コネクションごとにスレッドを持つような構成とした．NFSサーバ部は従来のファイルサーバの構造に基づいて実装し，キャッシュミスが起きた場合にFTPクライアントスレッドを生成するコードを付加した．

FTPサーバへのログイン操作は広域ネットワーク上のコネクション確立および認証をともなうため，それぞれのファイルアクセスについてログイン操作を行っ

た場合，アクセス時間が悪化する．これを改善するため，csd ではFTP サーバへのコネクションをキャッシュし，同一ファイルサーバへのアクセス時間を短縮している．

6. 評　価

E1～E4 にあげたアクセス装置の効果が csd によってどの程度満足されたかを明らかにするために，奈良先端科学技術大学院大学において 2 週間にわたって csd を運用し，効果を計測した．広域セマンティクスの隠蔽，インタフェースの連続性については前章で実現方法を述べた．これらの効果は数値化することが困難であるためここでは評価の対象としないが，我々の研究グループでは csd を含む WWFS のソースコードを Internet において公開し，広く評価を求めている☆．また，現在の Internet では距離と転送量に基づく課金が行われていないため，csd を用いることによるアクセスコストの軽減効果については明らかにすることができなかった．以下ではアクセス時間の短縮効果について述べる．

csd において NFS–FTP プロトコル変換を行うことで，平均アクセス時間を 1/2 に短縮することができた．コネクションキャッシュを用いることで，FTP サーバへのログイン操作を省略することができ，アクセス時間を 1/2 から 1/4 に短縮できることが確認できた．コネクションキャッシュのヒット率は 80%程度であった．

ファイルキャッシュのヒット率は 80%程度，ディレクトリキャッシュのヒット率は 85%程度であった．このうちのほとんどは同一ユーザによる複数回のアクセスであり，複数のユーザによってキャッシュを共有することによる効果は 10%以下であった．

GTR レコードを用いて適切なサーバを選ぶことで，アクセス時間を 1/2 から 1/10 程度に軽減することができた．

7. 考　察

分散ファイルシステムにおいてアクセス装置を実現することで，以下のようなことが明らかになったと考えられる．従来のクライアント・サーバ型ソフトウェアにおいて，クライアント・サーバ間にアクセス装置を設けることで E1～E4 であげたような効果が得られる．また，広域ネットワーク上のサーバに対するアクセス手順が複雑である場合，アクセス装置においてプロトコル変換を行うことで，資源アクセスの複雑な部分を隠蔽することができる．アクセス装置をアプリケーションに埋め込まず，共通のシステムサービスとして実装することで，様々なアプリケーションにおいてアクセス装置の効果を得ることができる．同時に，アプリケーションを簡単化することができる．

8. おわりに

一般ユーザによる広域ネットワークの利用が進むにつれて，セキュリティをはじめとする様々な問題が表面化している．これらの問題を解決するため，本論文では広域ネットワークにおける資源アクセスの複雑な部分を隠蔽し，LAN 環境における資源と同様に簡単に扱うことを可能とする装置として，アクセス装置を提案した．アクセス装置では，ネットワーク透過なサービスを広域ネットワークに拡張する．アクセス装置を分散ファイルシステム上に実装し，インタフェースの連続性，様々なセマンティクスの隠蔽，アクセス時間の短縮，アクセスコストの軽減などの効果が得られることを確認した．今後，アクセス装置の問題点，および，アクセス装置の適用範囲を明らかにしていく予定である．

謝辞　本研究を行うにあたり熱心に議論していただいた WIDE プロジェクトの皆様に感謝いたします．

参考文献

1) CERT Coordination Center: wuarchive ftpd Trojan Horse, CERT Advisory CA-94:07 (1994).
2) CERT Coordination Center: ghostscript Vulnerability, CERT Advisory CA-95:10 (1995).
3) Levy, E. and Silberschatz, A.: Distributed File Systems: Concepts and Examples, *ACM Computing Surveys*, Vol.22, No.4, pp.321–374 (1990).
4) Satyanarayanan, M.: Scalable, Secure, and Highly Available Distributed File Access, *IEEE Computer*, pp.9–19 (1990).
5) Kadobayashi, Y., Yamaguchi, S. and Miyahara, H.: WWFS: A Framework for Distributing Information in the Internet Environment, *Proc. 7th IEEE Region 10 International Conference TENCON'92*, pp.227–231 (1992).
6) Kadobayashi, Y., Yamaguchi, S. and Miyahara, H.: WWFS: An Evolutionary Framework for File Sharing in the Internet, *Proc. INET'93*, pp.DGB-1–DGB-8 (1993).
7) Kleiman, S.R.: Vnodes: An Architecture for Multiple File System Types in Sun UNIX, *Proc. Summer USENIX Conference*, Atlanta,

☆ ftp://wwfs.aist-nara.ac.jp/WWFS/dist/

GA, pp.238–247 (1986).
8) Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D. and Lyon, B.: Design and Implementation of the Sun Network File System, *Proc. Summer USENIX Conference* (1985).
9) Postel, J. and Reynolds, J.: File Transfer Protocol, *RFC 959*, Information Sciences Institute (1985).
10) Neuman, B.C.: The Prospero File System User's Manual, Dept. of Computer Science, University of Washington, alpha 4.4 edition (1991).
11) Blaze, M. and Alonso, R.: Long-Term Caching Strategies for Very Large Distributed File Systems, *Proc. Summer USENIX Conference*, Nashville, TN, pp.3–15 (1991).
12) Baker, M.G., Hartman, J.H., Kupfer, M.D., Shirriff, K.W. and Ousterhout, J.K.: Measurements of a Distributed File System, *Proc. 13th ACM Symposium on Operating Systems Principles*, pp.198–212 (1991).
13) Lopez, H., Kadobayashi, Y., Yamaguchi, S. and Kikuno, T.: A Scalable Alternative for Replicated File Servers Using IP Multicast, *Proc. ICOIN-9*, pp.487–492 (1994).
14) Muntz, D. and Honeyman, P.: Multi-level Caching in Distributed File Systems, Technical Report CITI Technical Report 91-3, University of Michigan (1991).

(平成7年9月29日受付)
(平成8年3月12日採録)

門林 雄基（学生会員）

平成4年大阪大学基礎工学部情報工学科卒業．平成6年同大学大学院基礎工学研究科博士前期課程修了．現在同博士後期課程在学中．インターネットアーキテクチャ，分散ファイルシステムに関する研究に従事．電子情報通信学会会員．

山口 英（正会員）

昭和61年大阪大学基礎工学部情報工学科卒業．昭和63年同大学大学院博士前期課程修了．平成2年同大学大学院博士後期課程を退学し，同大学情報処理教育センター助手に着任．以後，平成4年奈良先端科学技術大学院大学情報科学センター助手，平成4年同大学情報科学センター助教授を経て，平成5年同大学情報科学研究科助教授．平成3年工学博士（大阪大学）．インターネットアーキテクチャ，ネットワークセキュリティに関する研究に従事．電子情報通信学会，IEEE, Internet Society各会員．

宮原 秀夫（正会員）

昭和42年大阪大学工学部通信工学科卒業．昭和47年同大学大学院博士課程修了．昭和48年京都大学工学部助手．昭和55年大阪大学基礎工学部助教授．昭和62年同大学大型計算機センター教授．平成1年同大学基礎工学部情報工学科教授．平成7年より同大学大型計算機センター長併任．昭和58～59年米国IBMトーマスワトソン研究所客員研究員．システム性能評価，マルチメディアシステム，広帯域通信網，ネットワーク管理に関する研究に従事．IEEE，電子情報通信学会各会員．