広域分散FTPサービスへのデマンド型配送方式の適用と評価

菅野浩徳† 曽根秀昭††

インターネットにおける主要なサービスの1つに，ミラーリングによりFTPサーバを複数化し，広域に分散配置した，広域分散FTPサービスがある．しかし，配送されたファイルには，削除されるまでにまったく利用されないものも多く存在し，無駄な配送と蓄積を生じている．その大きな要因の1つは，ファイルの配送が網羅的であり，需要を反映する仕組みを有しないことである．本稿では，大規模なミラーサーバを積極的に活用したシステムとして代表的なRingサーバを対象に，デマンド型配送方式の適用について考察する．本方式は，キャッシング技術をベースに，利用者の要求に応じた配送を行うことで，ファイルやネットワークリソースの効率性を向上させる．実験に基づく評価により，サーバ間の不要な配送トラヒックや不要なファイルの蓄積が抑制され，ファイルの利用効率が向上することが確認でき，提案方式の有効性が示された．

Application of an On-demand Delivery Method to Wide-area Distributed FTP Service and Its Evaluation

Hironori Kanno† and Hideaki Sone††

There is a wide-area distributed FTP service as one of the main services in the Internet. In this system, a FTP server is copied by mirroring to plural servers, and these servers are distributed and installed in a wide area on a network. However, in delivered files, there are a lot of files which are not used at all till it is deleted. It has produced useless delivery traffic and useless accumulation. We consider that one of the big reason is as follows, it performs the file delivery between servers comprehensively, and it does not have the structure of the file delivery which can reflect a server user's demand. In this paper, we consider an application of an on-demand delivery method to the Ring server which is the representative system which utilized large-scale mirror server positively. The proposal method is based on a caching technology, which can improve the efficiency of files and network resources by doing delivery depending on a demand of a user. We evaluated it by experiment, and verified its validity. From the result of this experiment, the unnecessary delivery traffic between servers was able to be stopped, and the use efficiency of file has improved. Therefore, its validity was verified.

1. はじめに

本稿では，ミラーリング技術をベースとした広域分散FTPサービスに対し，ファイルやネットワークリソースの効率化を可能とする，キャッシング技術をベースとしたデマンド型配送方式[1)~3)]の適用について考察する．具体的検討のため，大規模なミラーサーバを積極的に活用したシステムとして代表的なRingサーバ[4)]のFTPサービスを対象とする．

インターネットにおける主要なサービスの1つに，ミラーリングによりFTPサーバを複数化し，広域に分散配置した，広域分散FTPサービスがある．クライアントはネットワーク的に近いサーバを選択することで，より良好なスループットが得られ，また，サーバの負荷分散や可用性向上の効果が期待できる[5)]．

しかし，近年のインターネット利用者数の増加や利用形態の多様化にともなって扱うファイルの数やサイズがともに増加しており，ミラーサーバへ分散蓄積を行うための通信トラヒックの増加や各サーバにおける蓄積領域の大容量化など，その運用と管理にかかるコストが肥大化し，システムの維持・運用が困難となってきている．筆者らはRingのミラーサーバの1つであるring.tains.tohoku.ac.jpを運用管理している．このサーバはミラー元であるベースサーバから全ファイルをミラーしており，ミラーリングのための転送（受信）量は1日2GBを超える場合もあり，ファイル容量は800 GBを超えている（2005年2月現在）．しかしミラーされたファイルの約9割は，削除されるまで

† 仙台電波工業高等専門学校
Sendai National College of Technology
†† 東北大学情報シナジーセンター
Information Synergy Center, TOHOKU University

にまったく利用されていない[6]．

ミラーリングでは，利用者が必要と思われるファイルをあらかじめ選択して配置することになるため，その見極めと管理が複雑となる．ミラーリングの設定は，一般的に管理者による人的な作業によるため，ファイル数が多くなると細かな制御が困難になり，あるいは細かな制御のための労力が増大する．一方，たとえばOS などのディストリビューションのように，内容の一貫性や統一性，利便性のために，個々のファイルの利用の有無にかかわらず，ディレクトリツリー構造を含めた同一性の保持が求められるものもある．この場合，各サーバに配置されるファイルの選択とその配送は，網羅的にならざるをえない．しかしながら，利用のないファイルを配送し蓄積することは，ネットワークシステムの効率性から見て好ましくなく，また，先述の運用コスト問題の一因にもなっており，改善を要する．

ミラーサーバ間の配送は，クライアントのアクセスとは独立に行われ，クライアントの要求を反映する仕組みを有さない．これが利用のない無駄なファイルの配送と蓄積を生じる大きな要因の 1 つと考えられる．これら現行のシステムにおける問題を解決するためには，利用者からのリクエストに応じて目的とするファイルを転送して可用化するデマンド型が有効であると考えられる．これにより，利用者の需要をサーバ間のファイル配送に反映させて，効率的な配送が可能になる．一方，デマンド型配送では，ファイルの最初の利用者について，サーバ間の配送時間が可算されるため，素早い応答を得るためにはサーバ間の通信回線が十分に高速である必要がある．近年，ネットワークの広帯域化が急速に進行しており，今後，さらにデータ伝送時間は短縮され，応答時間は改善されてゆくものと考えられる．それでも，ミラーリング方式との比較において，ファイルの最初の利用者の応答時間の差はゼロにはならないが，その差を許容して効率性や経済性を求める場合に，本方式は有効と考える．実際に Ring サーバにおいても，効率性や経済性の観点から，キャッシング技術をベースとした方式への見直しが検討されており，本研究はその 1 つの指針を与えるものである．

以下では，2 章で従来方式の概要について述べる．3 章で提案する方式の概要と動作などについて述べる．4 章で実験による評価と考察を，そして 5 章で他方式との比較と検討を行い，6 章で本稿のまとめを述べる．

2. 従来方式の概要

ミラーサーバを用いた広域分散 FTP サービスは，

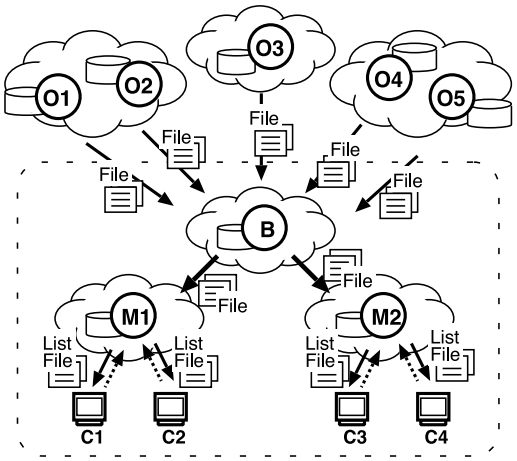

図 1 Ring サーバのミラーリングモデル
Fig. 1 Mirroring model of Ring servers.

あらかじめファイルをミラーサーバにコピーして可用化するミラーリング技術に基づく．プロトコルは FTP プロトコル[7),8)] を用いる．ミラーリングツールには，ftpmirror[9)] や rsync[10)]，wget[11)] などが用いられる．Ring サーバは，ベースサーバと，広域に分散配置された 30 台強のミラーサーバからなる．ベースサーバは，国内外の著名なライブラリを持つオリジンサーバに接続してミラーを行い，ミラーサーバはベースサーバに接続してミラーを行う 2 段ミラー構成となっている．

そのミラーリングモデルを**図 1** に示す．

図中の O1〜O5 はオリジンサーバ，B はベースサーバ，M1，M2 はミラーサーバ，C1〜C4 はクライアントとする．Ring サーバには，ベースサーバのバックアップサーバもあるが，この図では省略している．また，当モデルにおいてオリジンサーバはベースサーバからの要求に対して受動的にファイルを供給するだけの役割を持ち，直接的な構成要素ではない．M1，M2 のミラーサーバのような，クライアントの近端に配置されるサーバはエッジサーバと呼ばれることもある．

B は O1〜O5 の各サーバに定期的もしくは必要に応じてアクセスを行いミラーしておく．M1，M2 は同様に B にアクセスしてミラーしておく．サーバ間の同期については，それほどの即時性は要求されず，かつ，更新頻度は低いのでミラーリングは 1 日 1 回ないし数回程度である．また，サーバ側の情報に更新があっても，クライアントへ通知することはなく，情報の同期に対して比較的寛容である．クライアント C1〜C4 は近隣のミラーサーバを選択してアクセスし，目的とするファイルを取得する．クライアントからのアクセスは基本的に読み出しだけであり，書き込みは行われ

ない．

このように，ミラーサーバ間の配送処理はクライアントからのアクセスとは独立に行われ，配送の際にそのファイルに対する需要を知る術がない．これが，利用されることのない無駄なファイルの転送と蓄積を生じる大きな要因の 1 つと考えられる．また，ミラーサーバの中には，ベースサーバの一部のファイルしかミラーしていないものもある．それは，ベースサーバに蓄積されるファイル容量が非常に大容量でかつ増加し続けており，各ミラーサーバの運営母体が異なることもあって，十分な DISK 容量や回線容量を確保できない，などといった事情による．この状態では，どのエッジサーバからでも同じようにファイルを取得することが難しくなり，利用者へのサービスレベルが低下する．さらに，各ミラーサーバの管理者は運用状況に応じてミラーリングの設定を見直す必要があり，その運用負荷はけっして低くない．

3. 提案方式

提案するデマンド型配送方式は，クライアントからの要求によってファイルをキャッシュサーバにコピーして可用化するキャッシング技術に基づく．デマンド型配送方式によれば，利用者の需要をサーバ間のファイル配送に直接反映させ，利用のない無駄なファイルの転送を抑えて，効率的な配送が可能となる．

3.1 概　　要

提案方式は，ディレクトリサーバ，一次サーバ，二次サーバ，クライアントの 4 種類のノードから構成される．以下に各ノードの主な役割を説明する．

ディレクトリサーバ：一次サーバのディレクトリ情報の管理，二次サーバのアドレス情報や認証情報などの管理を行う．また，これらの情報を二次サーバへ提供する．

一次サーバ：オリジナルのファイル（一次ファイル）を保持し，ディレクトリサーバからの要求に応じてディレクトリ情報を提供する．また，二次サーバからの要求に応じてファイルを提供する．

二次サーバ：クライアントからの要求に応じてディレクトリ情報を提供する．また，クライアントからの要求に応じて，一次サーバや他の二次サーバにアクセスしてそのファイルを取得し，自分の蓄積領域に蓄積（キャッシュ）するとともにクライアントに提供する．

クライアント：ファイル転送ソフトウェアが動作する端末装置．二次サーバに対して接続し動作する．

デマンド型配送方式を適用したモデルを**図 2** に示す．O1〜O5 は一次サーバを，S1，S2 は二次サーバを，D はディレクトリサーバを，C1〜C4 はクライアントを示す．図 2 のネットワークやサーバ構成は論理的なものであり，物理的な構成を示すものではない．

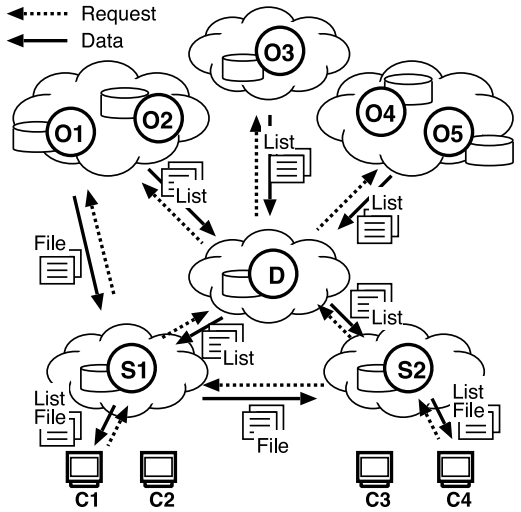

図 2 デマンド型配送方式を適用したモデル
Fig. 2 An on-demand delivery system model.

D は O1〜O5 に定期的にもしくは必要に応じて接続し，各サーバからディレクトリ情報（ファイルリスト）を得て，自分の持つディレクトリ情報を更新するとともに，S1，S2 に通知する．クライアント C1〜C4 は近隣の二次サーバを選択してアクセスし，目的とするファイルを取得する．

動作例について述べる．たとえば C1 がファイルを取得する場合，C1 は S1 にディレクトリ情報を要求し取得する．C1 はそのディレクトリ情報を参照し，欲しいファイルを選択して S1 にファイルの取得要求を送る．S1 にそのファイルがキャッシュ済みであれば，S1 はそのファイルを C1 に配送する．なければ，ディレクトリ情報から当該ファイルを保有するサーバを検索し，仮に O1 が保有しているとすると，O1 からファイルを取得し蓄積するとともに C1 に配送する．そして，S1 はそのファイルを保持したことを他の二次サーバに通知する．その後，たとえば C4 でこれと同じファイルの取得を要求した場合，S2 は O1 に接続することなく，S1 からファイルを取得し蓄積するとともに C4 へ配送する．

2 章で示した Ring サーバのモデルと対比すると，一次サーバはオリジンサーバに，二次サーバはミラーサーバに相当する．一次サーバはディレクトリサーバや二次サーバからの要求に対して受動的にファイルを供給するだけの役割を持ち，直接的な構成要素ではな

い．二次サーバは，RFC959[7] で規定された標準的なFTP コマンドをサポートし，あたかも一次サーバであるかのように動作するため，クライアントのソフトウェアの変更は必要ない．

ディレクトリ情報の同期精度については，従来方式と同様に，一次サーバへのポーリング間隔が支配的であるため，一次サーバにおけるファイル情報の変更頻度や従来方式における設定など十分考慮して適切に設定する必要がある．一次サーバへのポーリング間隔が従来方式のミラーリング間隔と同じであれば，同等のサービスが提供される．

3.2 ディレクトリサーバによるディレクトリ情報の管理

本方式では，ディレクトリサーバが一次サーバからディレクトリ情報を取得し，すべての二次サーバに配布する仕組みとした．これは，二次サーバのファイルは初期状態においては空であり，運用状態においても部分的であることから，二次サーバのディレクトリ情報に頼るわけにはいかないこと．また，既存のキャッシュシステムのように，各サーバが一次サーバにアクセスしてディレクトリ情報を取得することは，その要求や返送内容が重複することも多いことから非効率であり，サーバ負荷やトラヒックの増大にもつながるため，これを効率的に抑制する仕組みを持つことが望ましいこと，などの理由による．

一次サーバのファイルシステムは，ディレクトリとファイルからなるツリー構造を持つものとし，ディレクトリサーバでもそれに準じた構造で管理を行うものとする．そして，たとえば **図 3** に示すように，一次サーバの任意のディレクトリやファイルを，ディレクトリサーバ側の任意のディレクトリにマウントすることで，1 つのツリー構造にまとめて管理するものとする．

これにより，異なる複数の一次サーバが提供する独立したファイルを，クライアントは二次サーバを通して仮想的な 1 つの巨大なファイルとして認識することが可能となる．

二次サーバでは，クライアントからのリスト要求などに対して，このディレクトリ情報を用いて返答する（**図 4**）．

3.3 ディレクトリ情報の配布

ディレクトリサーバで管理するディレクトリ情報を二次サーバに配布する方式について述べる．

ディレクトリサーバでは，ディレクトリ情報を二次サーバへ配布する必要があるので，二次サーバの IP アドレス情報を保持しておくものとする．また，ディレクトリサーバと二次サーバ間の情報交換のために**表 1** に示すようないくつかのコマンドを FTP の拡張コマンドとして実装するものとする．

各コマンドの構文や機能については，関連する個所において説明する．

ディレクトリサーバは，初めてシステムを起動するときなど，(起動時にパラメータなどで指定することで) 初期化状態で起動するとき（以下コールドスタートと

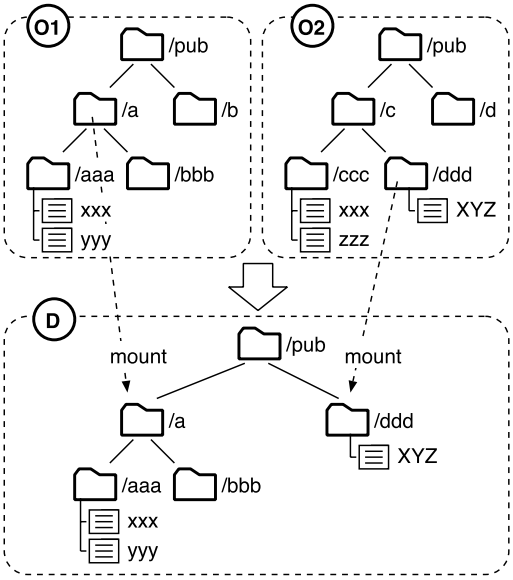

図 3 ディレクトリサーバでのディレクトリ構造例
Fig. 3 An example of directory structure on directory server.

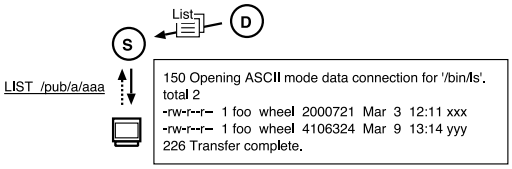

図 4 二次サーバへのリスト要求と応答
Fig. 4 Request to secondary server and response.

表 1 コマンドセット
Table 1 Command sets.

コマンド	概要
UPDATE	ディレクトリ情報の変更通知
ALIST	ディレクトリ情報の全リストの要求
MLIST	ディレクトリ情報の更新リストの要求
IHAVE	一次ファイルの取得通知
XRETR	二次サーバ間での一次ファイルの取得要求
HELLO	二次サーバ間での状態確認
SELIST	二次サーバリストの要求
XSELIST	二次サーバリストの送信

呼ぶ）には，一次サーバからディレクトリ情報を新たに取得するものとする．このとき，コールドスタート処理が完了したときの時刻をコールドスタートタイムとして記録し，ディレクトリ情報の更新回数を 0 にしておく．同様に二次サーバも，初期化状態で起動するとき（同様に以下コールドスタートと呼ぶ）には，ディレクトリサーバからディレクトリ情報とコールドスタートタイム，更新回数を新たに取得して，同期させておくものとする．

運用状態において，ディレクトリサーバは定期的もしくは必要に応じて一次サーバへアクセスし，ディレクトリ情報に変更があれば UPDATE コマンドを二次サーバに送信する．これには，マルチキャストもしくはユニキャストを選択可能とし，ネットワークやサーバ環境を考慮して管理者が決定するものとする．UPDATE の構文を以下に示す．

```
UPDATE  seq
{cstm}
{list 1}
  :
{list n}
.
```

‘cstm’ はディレクトリサーバのコールドスタートタイムである．‘list 1’〜‘list n’ は変更のあったファイルのリストで，更新リストと呼ぶ．‘seq’ はこの更新リストの番号であり，ディレクトリサーバのコールドスタート時からのディレクトリ情報の更新回数をあてる．更新リストの先頭には，追加，内容変更，削除を示すフラグ {A,C,D} が付与される．UPDATE コマンドの終端は，行頭がピリオド（‘.’）の単独行とする．二次サーバは，UPDATE コマンドに示されるコールドスタートタイムが，自身が保持しているディレクトリ情報のコールドスタートタイムと一致するかどうかをまず確認する．一致であれば，両サーバにおいて，更新情報のベースとなるディレクトリ情報が同じものであることを示している．すなわち，ディレクトリサーバのコールドスタート時のディレクトリ情報と，二次サーバがコールドスタート時にディレクトリサーバから得たディレクトリ情報とが同じものであることを示している．したがって，二次サーバは，UPDATE コマンドに示される更新情報に基づき，自身の情報を更新して，その seq 値を記憶する．不一致であれば，ディレクトリサーバが再度コールドスタートを行ったなどの理由により，両サーバにおいて，更新情報のベースとなるディレクトリ情報が異なってしまっていることを示している．よって，この UPDATE 情報を二次サーバに適用すると，誤ったディレクトリ情報を保持してしまう可能性があるため，不適用とする．この場合には，ALIST コマンドにより新たにディレクトリサーバからディレクトリ情報を取得して，両サーバのディレクトリ情報を同期させることが必要となる．なお，ディレクトリサーバでは，最新の更新リストから過去数世代分の更新リストを保存可能とする．

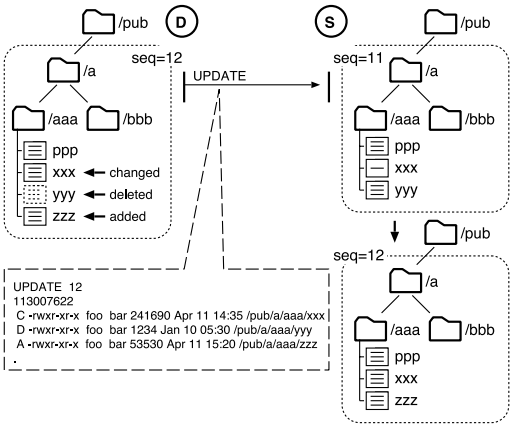

図 5 ディレクトリ情報の更新通知例
Fig. 5 An example of an update notice of directory information.

図 5 はディレクトリ情報の更新通知の例である．

図左下の枠中が UPDATE コマンドの内容で，seq 値は 12，cstm 値は 113007622，`/pub/a/aaa/xxx` は内容変更，`/pub/a/aaa/yyy` は削除，`/pub/a/aaa/zzz` は追加，であることを通知している．二次サーバではこの情報を基にディレクトリ情報を更新して，その seq 値 12 を記憶する．またキャッシュファイルを調査して，変更や削除対象のファイルがあれば蓄積領域から削除する．もし，cstm 値が一致しているにもかかわらず，ディレクトリサーバからの seq 値が二次サーバの記憶する seq 値 +1 を超える値だった場合，UPDATE コマンドの取りこぼしが考えられる．この場合には，二次サーバが MLIST コマンドにより取りこぼし分の更新リストの再送依頼をする．そして得られた更新リストと UPDATE で通知済みの更新リストからディレクトリ情報を更新する（**図 6**）．ディレクトリサーバにおいて，コールドスタートがミリ秒単位などの非常に短い間隔で繰り返し行われることは考えにくいため，cstm 値は秒単位で十分と思われる．図 5 では一例として UNIX Time を用いている．

MLIST の構文は以下のとおりである．

MLIST seq

‘seq’ は，要求する更新リストの番号である．MLIST コマンドの応答メッセージは，応答番号，MLIST 文，

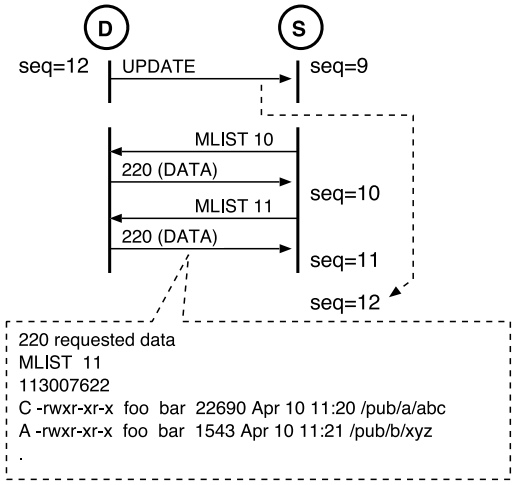

図 6 更新リストの再送例
Fig. 6 An example of re-transmission of update list.

seq 値，cstm 値と更新リストから構成されるものとする．応答メッセージの終端は，行頭がピリオド（‘.’）の単独行とする．再送処理はユニキャストによるものとする．

当通知方式はディレクトリサーバ主導による方式であり，変更があったタイミングですみやかに二次サーバに通知されるのでディレクトリ情報の変化に対する追従性に優れる．一方，一定時間間隔もしくはディレクトリサーバの一次サーバへの接続時間などに同期して二次サーバ側から更新の有無を問い合わせる方式も考えられるが，ディレクトリ情報に変更がない場合にも問合せを行うので，無駄なトラヒックとサーバ負荷を生じる．これに対して，当通知方式の場合にはそのようなことがない．

ディレクトリサーバや二次サーバでは，第三者からの不正なアクセスを防止するため，最低限の方策として，アクセス元 IP アドレスによる許可・不許可の制限や，ログインアカウントとパスワードによる認証を行うものとする．そのため，ディレクトリサーバでは，二次サーバの IP アドレス情報および，ログインアカウントとパスワードを保持し，二次サーバでは，ディレクトリサーバの IP アドレス情報および，ログインアカウントとパスワードを保持するものとする．

3.4 一次ファイルの取得

二次サーバおよびクライアントが一次ファイルを取得する場合の動作方式について述べる．**図 7** にその動作例を示す．

クライアント C は，二次サーバ S1 から LIST コマンドによって `/pub/a/aaa` のディレクトリ情報を取得し，このリストから目的とする一次ファイル `xxx` を

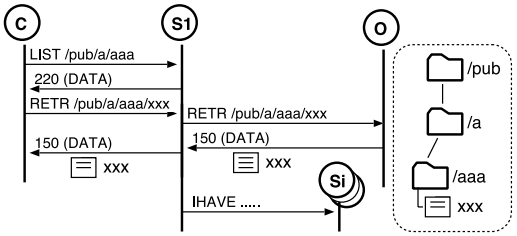

図 7 一次ファイル取得時の動作例
Fig. 7 Example of getting a primary file.

見つけ，RETR コマンドによって要求する．C からの要求を受けた S1 は，そのファイル `xxx` をすでに保有していれば C に配送する．保有していなければ，自分のディレクトリ情報からそのファイルを保有する二次サーバ名を調べ，どの二次サーバもまだ保有していない場合，S1 は一次サーバ O からファイルを取得し蓄積するとともに C に配送する．そして S1 はそのファイルを保持したことを IHAVE コマンドにより他の二次サーバすべて（Si：i=2〜n）に送信する．これには，マルチキャストもしくはユニキャストを選択可能とし，ネットワークやサーバ環境を考慮して管理者が決定するものとする．なお，二次サーバから一次ファイルを取得しようとしたときに，ダウンなどの理由で応答を返さないときは，他の二次サーバや一次サーバから目的とする一次ファイルを取得するものとする．

IHAVE の構文を以下に示す．

IHAVE mdtm md5checksum filepath

‘mdtm’ は一次サーバ上でのファイル更新時刻，‘md5checksum’ はファイルの中身から MD5 [12] によって生成した文字列，‘filepath’ は取得した一次ファイルのフルパス名である．IHAVE コマンドを受信した二次サーバでは，ディレクトリ情報に送信元の二次サーバ名，MD5 チェックサム，フルパス名を登録する．登録済みであった場合には，MD5 チェックサムには衝突の可能性があるので，ファイル更新時刻と MD5 チェックサムの双方が同じであることを確認し，送信元の二次サーバ名を追加する．その後，たとえば Si でこのファイル `xxx` の取得要求が発生した場合，Si はすでに S1 にあることを知っており，O1 に接続することなく S1 からファイルを取得し蓄積するとともにクライアントへ配送する．また，同様に IHAVE コマンドを S1 を除く全二次サーバに対して送信する．これらの処理により，一度どれか二次サーバに取得されたファイルは，二次サーバ間で流通されることになり，一次サーバへのアクセスを抑制できる．

クライアントへのサービス中に二次サーバのディレクトリ情報が更新されると不整合が生じるが，これは

従来方式の配信処理でも同様であり，本提案方式の問題ではないため本稿における検討の範囲外とする．

3.5 二次サーバの選択

複数の二次サーバが同一の一次ファイルを持っている場合に，どの二次サーバを選択するのかという問題について，本方式では能動的観測に基づいて行うことを考える．具体的には，RTT（Round Trip Time）とサーバ負荷の状態により最適なサーバを選択する．能動的観測のために，二次サーバでは，定期的に他の二次サーバと HELLO コマンドの交換を行う．HELLO コマンドを受信した側では，応答メッセージにログインユーザ数と CPU 平均負荷値を含めて返す．これらの値は，二次サーバがたとえば UNIX システムであれば kernel の保持する情報から uptime(1) コマンドや sysinfo 構造体などで取得できる．以下に HELLO コマンドの応答例を示す．

```
220 requested data
users: 7
load_average: 0.05, 0.06, 0.16
reply_delay: 0.005
```

CPU 平均負荷値は，過去 1，5，15 分の平均負荷である．HELLO コマンドは，これらいずれかの時間間隔（たとえば 5 分に 1 回）で発行して，その間隔に該当する平均負荷の値を採用することとする．これにより平均負荷を定常的に把握できる．reply_delay 値はサーバが HELLO を受信してからその応答メッセージを送信するまでにかかった時間である．HELLO を送信してから，その応答メッセージを受信するまでの時間を RTT とする．この RTT 値は，ネットワーク遅延と reply_delay 値の合計である．よって RTT 値から reply_delay 値を引くことでネットワーク遅延が得られる．ログインユーザ数や CPU 平均負荷値にはネットワーク性能についての情報がないので，これを補助的に用いるものとする．そして，二次サーバの選択には，ネットワーク遅延と reply_delay 値がそれぞれ最小のサーバを選択し，両方の値が等しい場合には，ログインユーザ数が少なく CPU 平均負荷値の小さいサーバを選択するものとする．

HELLO を交換するための二次サーバのリストは，SELIST コマンドによってディレクトリサーバから取得可能とする．これは，二次サーバのコールドスタートの際や，問合せ先の二次サーバが応答しなかったときなどに発行される．また，二次サーバのリストは，XSELIST コマンドによりディレクトリサーバからも送信可能とする．これは，二次サーバの増減にともないディレクトリサーバの二次サーバリストが変更された直後などに発行される．新たな二次サーバのリストを受信したサーバでは，減ったサーバについては，そのサーバに関係する不要となったデータを削除し，増加になったサーバについては，そのサーバからのデータの受け入れ準備などを行うものとする．

3.6 ディレクトリサーバの冗長構成

当方式では，ディレクトリサーバに様々な情報が集約され，また発信されるなど，ディレクトリサーバの重要性が高い．システム起動時にディレクトリ情報は二次サーバに送信済みであり，また，サーバ間の情報交換についての再送機能を備えているので，ディレクトリサーバの短時間の停止については，システムとして大きな不都合を生じる心配はない．しかし，長時間にわたる停止があると，一次サーバでの情報更新があった場合に，いつまでも二次サーバ側へその情報が伝わらない状態が続くことになり，サービスの低下につながる可能性も高くなる．

そこで，実際の運用においては，ディレクトリサーバのハードウェア，ネットワークなどを二重化するといった手法や，ディレクトリサーバを複数台配置するホットスタンバイなどといった手法により，ディレクトリサーバの冗長性を高め，信頼性を確保することも必要である．複数のディレクトリサーバ間で情報の一貫性を保持するためには，当方式の特長を活かした最適な構成を考えることが必要となる．本稿ではその必要性への言及にとどめ，具体的な手法についての検討は今後の課題とする．

4. 実験と評価

運用システムでは，ネットワークやサーバの利用状態が刻々と変化するため，利用率や蓄積量などの評価指標について既存システムと提案システムとを同じ条件で比較するのは困難である．そこで本稿では計算機シミュレーション実験によって既存システムとの比較を行い評価する．本稿では，Ring サーバを比較対象とする．

4.1 準　　備

シミュレータは perl 言語で独自に構築したもので，UNIX プラットフォーム上で動作させた．実験システムは，最も基本的な構成を考え**図 8** のようなモデルを想定した．(a) は既存システムで，一次サーバ複数台（O1,O2,..,On）と，ミラーサーバ（M）1 台，クライアント複数台（C1,C2,..,Cn）とし，(b) は提案システムで，ミラーサーバを二次サーバ（S）1 台，ディレクトリサーバ（D）1 台に置き換えた構成とし，比較実験を行う．

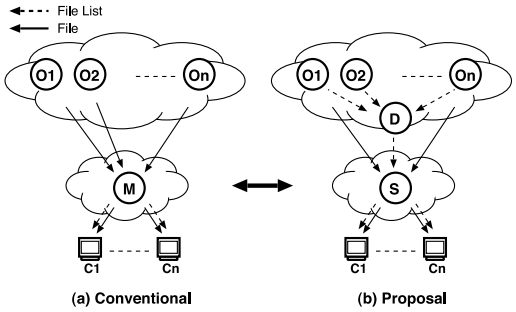

図 8 想定モデル
Fig. 8 Experimental environment.

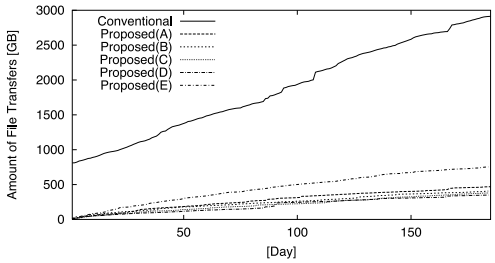

図 9 ミラーサーバ，二次サーバへのファイル転送量
Fig. 9 Amount of file transfers to mirror server and secondary server.

ミラーサーバやディレクトリサーバのシミュレーションデータには，ベースサーバの全ファイルをミラーしている ring.tains.tohoku.ac.jp のファイル情報を用いた．また，二次サーバのシミュレーションデータには，Ring サーバ群から任意に選んだ 5 つのミラーサーバ（サーバ A〜E とする）のアクセスログを用いた．測定は 1 日単位とし，使用したファイルリストおよびアクセスログは 2004 年 12 月上旬から 2005 年 6 月上旬までの約 180 日間（約 6 カ月間）のものを用いた．シミュレータは，これらの入力データから，各サーバのファイル転送量やディスク使用量，利用率といった情報を出力する．なお，使用したアクセスログについては，情報の秘匿性に注意を払い，利用者に関する情報が特定できる項目は使用していない．

4.2 結果と考察

ミラーサーバと二次サーバにおけるファイル転送量について**図 9** に示す．

図中で Conventional は従来方式（ミラーサーバ），Proposed（A〜E）は既存のサーバ A〜E に対して提案方式を適用し二次サーバとした場合である．ファイル転送量は 1 日目からの累積値である．従来方式では，ミラー開始時に一次サーバのファイルをあらかじめ転送するので，1 日目ですでに 800 GB 強の転送があることが分かる．さらにファイルの追加や変更による転送が約 11 GB/日あり，150 日目で約 2,500 GB を超えている．これに対して提案方式では，サーバ A〜E のいずれにおいても転送量が大きく抑えられており，この中で一番転送量の多いサーバ E でも約 4.1 GB/日（従来方式の約 37%）で，150 日目で約 670 GB（従来方式の約 27%）である．

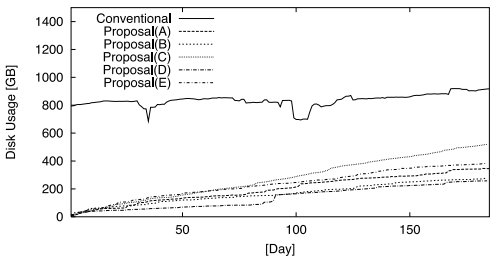

図 10 ミラーサーバ，二次サーバのディスク使用量
Fig. 10 Disk usage of mirror server and secondary server.

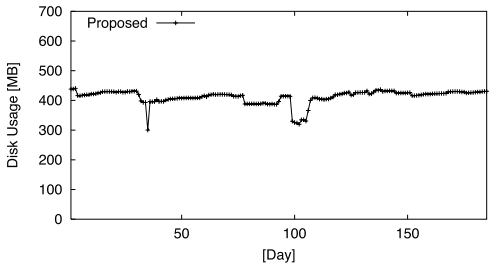

図 11 ディレクトリサーバにおけるディスク使用量
Fig. 11 Disk usage of directory server.

従来方式でのファイル転送量は Ring ベースサーバからの全ファイルのミラーリングに要する転送量であり，サーバ A〜E についても同じベースサーバをミラー元としているため，これをほぼ上限値と考えて差し支えない．

ミラーサーバと二次サーバにおけるディスク使用量について**図 10** に示す．図中で Conventional は従来方式（ミラーサーバ），Proposed は提案方式（二次サーバ）である．

従来方式のディスク使用量は，約 800 GB から約 900 GB であるのに対して，提案方式ではサーバ A〜E のいずれにおいても使用量が大きく抑えられており，この中で一番転送量の多いサーバ C でも 150 日目で約 400 GB（従来方式の約 50%）である．従来方式のディスク使用量は先述のファイル転送量と同様の理由により，サーバ A〜E の上限値と考えて差し支えない．

次に，ディレクトリサーバにおけるディスク使用量の推移について**図 11** に示す．

この結果によれば，ほぼ 400 MB 程度で推移しており，とりわけ巨大な容量の DISK 装置は必要ないこと

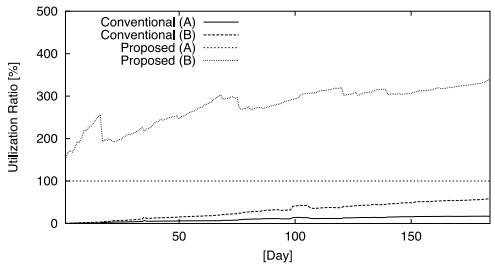

図 12 ファイルの利用率
Fig. 12 Utilization ratio of files.

が分かる．ディレクトリ情報に変更があった際に二次サーバに提供する更新リストは，すべてのディレクトリやファイルの情報ではなく，変更があったディレクトリやファイルについてのリストである．これは，全ディレクトリ情報に対して数%のオーダであり，数世代のリストを保存したとしてもそれほどディスク領域を圧迫しない．

続いて，ファイルの利用率について，サーバ E のグラフを**図 12** に例示する．

利用率は，ファイルのダウンロード回数 Ac と配送されたファイル数 Tc との比率 Ac/Tc とする．配送されたファイルが以後何回利用されたかということは日をまたがって計数しないとならないため，ダウンロード回数と配送されたファイル数は 1 日目からの累積値を用いる．このとき，同一ファイルに対して複数回のダウンロードがあった場合に，その回数を 1 回のみと計上した比率を利用率 A，そのまま回数分だけ計上した比率を利用率 B とする．図 12 において，Conventional（A）と Conventional（B）は，それぞれ従来方式における利用率 A と利用率 B で，Proposed（A），Proposed（B）は，提案方式における利用率 A と利用率 B である．提案方式では配送ファイル数はダウンロード回数に等しいため，利用率 A では 100%，そして利用率 B では 150%から 350%の高い利用率が得られている．これに対して，従来方式では利用率 A，利用率 B とも提案方式を下回っており，提案方式によってファイルの利用効率を大きく向上できることが分かる．他の 4 サーバについても，同様な結果が得られている．

最後に，二次サーバ間での交換ファイル数について示す．サーバ間の交換ファイル数は日をまたがって計数しないとならないため，1 日目からの累積値とした．A〜E の 5 サーバ間についてみると，2 サーバ間：336,838（21.4%），3 サーバ間：319,710（20.3%），4 サーバ間：249,156（15.8%），5 サーバ間：113,825（7.2%），交換なし：113,825（35.2%）であった．括弧内は全ファイルに対する割合である．このことから，6 割以上のファイルがサーバ間で交換され，再利用できることが分かる．二次サーバどうしでの交換ファイル数が多いほど，一次サーバへのアクセス抑制の効果が期待できる．

以上の実験結果から，既存のサーバ A〜E について，提案方式によりサーバ間の不要なトラヒックやディスク領域を抑制できることが示された．これは回線やディスク装置の効率的な利用と運用コストの削減につながる．

5. 他の方式との比較と検討

以上のとおり本方式は，ディレクトリ情報の一元管理とその分散管理の両立とキャッシング技術を主体とした効率性を特長とする．ディレクトリサーバに集約されたディレクトリ情報はすみやかに各エッジサーバ（二次サーバ）に伝達され，エッジサーバから一次サーバへのディレクトリ情報の問合せは発生しない．エッジサーバの保持するファイルは，さらに他のエッジサーバでも再利用される．これらの機能により一次サーバへの負荷が大きく抑制でき，また配送系全体についての配送効率の改善に有効である．さらに，本稿で示したとおり一次サーバには何ら変更を加えない構成が可能なため，現在ファイル提供サービスを行っている多くのサーバを一次サーバの対象として本システムに組み入れることができる．また，それらのサーバは本システム以外のユーザに対しては継続してサービスを提供できる．クライアントの変更も必要ないため，現システムに対する整合性も良い．

一般にキャッシュシステムには，squid [13] や apache [14] が用いられることが多いが，これらは Web サービスにおけるプロキシキャッシュとしての利用を前提に設計されているため，今回想定するような環境でのエッジサーバとして適用することは無理がある．たとえばプロキシに対応していない通常の FTP クライアントソフトは利用できず，Web ブラウザなどのようにプロキシに対応しているクライアントソフトを用いたとしても利用するエッジサーバを特定してブラウザに設定する必要がある．もしクライアントがファイアウォールの中にあって，外部のプロキシの利用に制限があったり，すでに組織内にあるプロキシの利用を義務付けられていたりした場合には当方式の利用は不可能である．このようなケースは企業などにおいて多々見受けられる．

キャッシュシステムを備えるリバースプロキシ機能によればエッジサーバ自身が一次サーバのように振る

舞えるためこのような問題は解消される．これに類似する方式として WWFS [15] や文献 16) のような方式などもある．しかしながら，これらの方式ではエッジサーバ間で取得したファイルを相互に融通するような機能を持たないため，キャッシュ効果は個々のエッジサーバに閉じられ局所的な効果にとどまる．すなわち，ファイルの変更状況の問合せやファイルの取得のために個々のエッジサーバが直接一次サーバへアクセスすることになり，一次サーバへの負荷を比較すると本提案方式のほうが優位である．

また，エッジサーバにおけるファイルの消去タイミングが一次サーバと連動できず不要なファイルがいつまでもキャッシュ上に残る，といったような DISK 利用効率面での問題もある．

ICP [17] を用いたキャッシュサーバ間で取得したファイルを相互に融通するような連係機能は，キャッシュサーバ間の問合せトラフィックの増加や被接続キャッシュサーバの負荷を招くため効率が悪く，定期的（周期=1 時間程度）に接続している他キャッシュサーバから所有ファイルのデータベース（digest）を取得し利用する Cache-Digest という方式もあるが，個々のサーバ間でそれぞれ個別に問合せを行うことには変わりがなく本質的な解決策ではない．さらに，これらではキャッシュサーバごとに連係先のキャッシュサーバを設定する必要があり，台数が多くなると設定のための手間が大きくなる．

6. おわりに

本稿では，ミラーリング技術をベースとした広域分散 FTP サービスに対し，ファイルやネットワークリソースの効率化を可能とする，キャッシング技術をベースとしたデマンド型配送方式の適用について考察した．具体的検討のため，大規模なミラーサーバを積極的に活用したシステムとして代表的な Ring サーバの FTP サービスを対象とした．既存システムにおける問題として，配送されたファイルには，削除されるまでにまったく利用されないものも多く存在し，無駄な転送と蓄積を生じており非効率であること，これはサーバ間におけるファイルの配送が網羅的であり，サーバ利用者の需要を反映する配送の仕組みを有していないことが大きな要因の 1 つと考えられること，などを示した．また，デマンド型配送方式の適用について，サーバの機能と構成，動作，配送プロトコルなどの設計を示した．提案方式によれば，ディレクトリサーバを配置してディレクトリ情報を一元管理し，利用者の要求に応じてファイルを二次サーバへ配送し保持することで，ファイルやネットワークリソースの効率性向上を可能とする．提案方式について実験による評価を行った結果，サーバ間の不要な配送トラヒックやディスク上の不要なファイルの蓄積が抑制され，ファイルの利用効率も向上することが確認でき，提案方式の有効性が示された．

我々は現在，当システムの実装を進めており，実装を用いた実環境での試用評価が今後の課題である．たとえば，各ノード間のデータ転送時間の評価，ディレクトリサーバの冗長構成についての検討，スケーラビリティに関する評価，などがあげられる．また，人気の高い OS やライブラリなど，需要が確定的なファイルについては，その事前配送を可能とすることで，応答性や利便性の向上が期待できるものと考えられる．さらに，デマンド型配送方式については，メールシステムや動画像配信などといった他システムへの適用も可能であると考えている．今後，これらへの応用についても検討を進めたい．

参考文献

1) 菅野浩徳，曽根秀昭，根元義章：デマンド型配送方式のネットニュースシステムへの適用と評価，情報処理学会論文誌，Vol.42, No.12, pp.2963–2972 (2001).
2) 菅野浩徳，曽根秀昭，根元義章：デマンド型ネットニュース配送方式におけるトラヒックのモデル化とレスポンスタイム評価，情報処理学会論文誌，Vol.44, No.3, pp.535–543 (2003).
3) 菅野浩徳，曽根秀昭，根元義章：デマンド型配送システムにおけるディレクトリサーバ分散手法，情報処理学会研究会報告 2000-DSM-19，Vol.2000, No.92, pp.13–18 (2000).
4) Ring Server Project. http://www.ring.gr.jp/
5) Hull, S., トップスタジオ（訳）：CDN プロトコル入門，日経 BP (2003).
6) 農人 聡，菅野浩徳，曽根秀昭：東北大学 RingServer の運用と利用統計，ITRC Technical Report, No.26, pp.18–23 (2004).
7) Postel, J. and Reynolds, J.: File Transfer Protocol, IETF RFC959 (1985).
8) Braden, R.: Requirements for Internet Hosts — Application and Support, IETF RFC1123 (1989).
9) FTPMIRROR. http://toyama.net/~ikuo/tools/ftpmirror.html
10) rsync. http://rsync.samba.org/
11) GNU Wget. http://www.gnu.org/software/wget/
12) Rivest, R.: The MD5 Message-Digest Algorithm, IETF RFC1321 (1992).

13) Squid Web Proxy Cache.
http://www.squid-cache.org/
14) The Apache Software Foundation.
http://www.apache.org/
15) 山口 英：UNIX Communication Notes: WWFS, UNIX MAGAZINE 1994–11 月号, pp.34–42, 株式会社アスキー (1994).
16) 伊東大輔, 菊池 豊：キャッシュを用いた Ring-Server の構築, ITRC Technical Report, No.26, pp.8–12 (2004).
17) Wessels, D. and Claffy, K.: Internet Cache Protocol (ICP), version 2, IETF RFC2186 (1997).

(平成 17 年 7 月 8 日受付)
(平成 18 年 2 月 1 日採録)

菅野 浩徳（正会員）

昭和 59 年茨城大学工学部卒業．同年（株）富士通東北システムエンジニアリング入社．平成 9 年より宮城教育大学教育学部非常勤講師．平成 13 年東北大学大学院情報科学研究科博士前期課程修了．平成 15 年より仙台電波工業高等専門学校助教授．平成 16 年より独立行政法人情報通信研究機構特別研究員併任．分散システム，情報ネットワーク等の研究に従事．情報処理教育，地域情報化等にも興味を持つ．

曽根 秀昭

昭和 53 年東北大学工学部電気工学科卒業．昭和 55 年東北大学大学院工学研究科修了．同年同大学工学部助手，平成 4 年同大電気通信研究所助教授．同大学情報科学研究科，大型計算機センターおよび総合情報システム運用センター勤務を経て，平成 13 年より同大学情報シナジーセンター教授および副センター長．工学博士．計算機システムとネットワークシステム等の運用に関する研究と，電子応用計測，耐ノイズ性通信方式等の研究に従事．IEEE，電気学会等の会員．