平成 14 年度

修士学位論文

部分ミラーのサーバ群に対する DNS を用いたサーバ選択手法

A Server Selection Method Using DNS for Partial Mirror Servers

1055112　　廣瀬 崇夫

指導教員　　菊池 豊 助教授

2003 年 2 月 24 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

部分ミラーのサーバ群に対する DNS を用いたサーバ選択手法

廣瀬 崇夫

インターネットでは各種サーバの負荷集中への対策として、ミラーサーバを用いたサーバの複数化による負荷分散が広く行なわれている。

大規模なミラーサーバを積極的に活用したシステムに、Ring Server プロジェクトがある。

Ring Server は、ミラーサーバがマスターサーバと同一のディレクトリツリーを保持するといった管理ポリシーで運営している。しかし、実際にはミラーサーバの同期には、ずれが生じており、ソフトウェアの発展にともなう情報コンテンツの増加は、サーバ間の同期をますます悪化させている。このため、ユーザが選択しアクセスしたサーバに、欲する情報コンテンツが存在しないといった問題がある。

ミラーの同期のずれを少なくする方法として部分ミラーのサーバを複数用いる方法がある。ミラーサーバの保持する情報コンテンツが各サーバごとに分散することで、管理する情報コンテンツが減り、同期のずれの減少が期待できる。また、ネットワークトラフィックの軽減、サーバへのアクセスによる負荷の分散も期待できる。しかし、この場合、各ミラーサーバの内容を確認して選択する方法が必要である。

本研究は、部分ミラーのサーバを複数用意し、DNS を用いて任意の情報コンテンツを保持するサーバを選択する手法を提案する。本提案により、ユーザが 1 つのサーバ名と、ある程度のディレクトリ情報を組み合わせたドメインネームを用いることで、任意の情報コンテンツを保持しているサーバにアクセス可能である。この場合、サーバ選択の対象となるサーバ群は、ディレクトリ情報の同一性が保証されないサーバや部分ミラーサーバであっても構

わない。

キーワード　DNS、ミラーサーバ、サーバ選択、広域分散システム

Abstract

A Server Selection Method Using DNS for Partial Mirror Servers

HIROSE, Takao

This paper, the technique of selecting the partial mirror server using DNS prepares two or more partial mirror servers is proposed.

The mirror server distributes load is widely used on the Internet.

The Ring Server Project is managing the large-scale mirroring system, whose server provides the directory tree structure as a master server that the mirror server is the same. However, mirror servers can't be synchronized always because of delivering delay, mirror server the digital content, which a user wants to acquire, may not exist when a user accesses.

Therefore the ring server is built using two or more partial mirror servers because partial mirror servers decrease delay of the synchronization of mirroring. That digital content distribute using partial mirror servers can mitigate the mirror server load and network traffic. Though, when using two or more partial mirror servers, in case the server is selected, user must check whether it has the digital content that a user wants to acquire.

The proposal technique adds directory information to DNS and can select a partial mirror server with the digital content that a user wants to acquire.

Consequently, building a ring server using two or more partial mirror servers can reduce delay of the synchronization by mirror processing. Furthermore, a server with

the digital content that a user wants to acquire can be selected by using this technique.

key words DNS, Mirror Server, Server Selection

目次

図目次

表目次

第 1 章

はじめに

インターネットでは各種サーバの負荷集中への対策として、ミラーサーバを用いたサーバの複数化による負荷分散が広く行なわれている。

大規模なミラーサーバを積極的に活用したシステムに、Ring Server プロジェクトの Ring Server がある。

Ring Server は、ミラーサーバがマスターサーバと同一のディレクトリツリーを保持するといった管理ポリシーで運営している。そのため、ディレクトリツリーの相違は想定していない。しかし、実際にはミラーサーバの同期には、ずれが生じており、ソフトウェアの発展にともなう情報コンテンツの増加は、サーバ間の同期をますます悪化させている。このため、ユーザが選択しアクセスしたサーバに、欲する情報コンテンツが存在しないといった問題がある。

ミラーの同期のずれを少なくする方法として部分ミラーのサーバを複数用いる方法がある。ミラーサーバの保持する情報コンテンツが各サーバごとに分散することで、管理する情報コンテンツが減り、同期のずれの減少が期待できる。また、ネットワークトラフィックの軽減、サーバへのアクセスによる負荷の分散も期待できる。しかし、この場合、各ミラーサーバの内容を確認して選択する方法が必要である。

本研究は、部分ミラーのサーバを複数用意し、DNS を用いて任意の情報コンテンツを保持するサーバを選択する手法を提案する。本提案により、ユーザが 1 つのサーバ名と、ある程度のディレクトリ情報を組み合わせたドメインネームを用いることで、任意の情報コンテンツを保持しているサーバにアクセス可能である。

以降、第 2 章では、ミラーサーバについての概要とその問題点について述べる。次に第 3

章では、サーバ選択手法が満たすことが望ましい性質をあげる。つぎに既存のサーバ選択手法を 3 つのモデルに分類し、各実装において概要を述べる。そして、それぞれのモデルの問題点を挙げる。第 4 章では、提案するサーバ選択手法における要求を挙げる。そして、本手法におけるサーバ選択の動作について説明し、最後にユーザ側から見たサーバ選択の動作に付いて説明する。第 5 章では、第 4 章で提案した手法を実装したプロトタイプシステムについて述べる。第 6 章においては、提案した手法に対して評価を行なう。第 7 章では、提案した手法をユーザが透過的に利用するシステムについて述べる。そして、第 8 章では、提案した手法の問題点について考察を行ない、最後に第 9 章で本論文のまとめを行ない、今後の課題を挙げる。

第 2 章

ミラーサーバ

本章では、まず、ミラーサーバが利用されるようになった背景について述べる。次にミラーサーバの概要を述べ、最後にミラーサーバの問題点を挙げる。

2.1 背景

近年のインターネットの急激な普及にともない、さまざまなサービスが提供されている。一方、利用者数も爆発的に増加して来ている。このため、サービスを提供しているサーバへのアクセスの集中や、その周辺のネットワークにおいてトラフィックが集中するといった問題が生じている。サーバやネットワークへのアクセスの集中により、サービスの処理時間に遅延が生じ、サーバの停止によるサービスの停止さえある。

この問題を解決するためには、多数のアクセスを処理することが可能で、スケーラビリティのあるサーバを構築する必要がある。単一のサーバマシンの性能には限界があり、また、その性能を拡張することも容易ではない。そこで、複数のサーバマシンを用いて、単一のサービスを提供することが多い。しかし、それらを局所的に配置しては、ネットワークトラフィックの集中が解消されない。よって、サーバマシンを広域に分散配置することが有効である。

2.2 マスターサーバとミラーサーバ

多くの場合、サービス内容の変更などは、管理の容易さから 1 台のサーバで行なっている。このサーバをマスターサーバ (Master Server) と呼ぶ。そして、複数のサーバはマスター

サーバをコピーして、同一のサービスを提供している。特定のサーバをコピーするサーバをミラーサーバ (Mirror Server) と呼ぶ。また、サーバやサービスをコピーすることを「ミラーする」または「ミラーリング」と呼ぶ。

なお、本研究で扱うマスターサーバとミラーサーバは、ファイルサーバに限定する。ファイルサーバは、UNIX のファイルシステムと同様に、ディレクトリを用いた木構造により、ファイルを管理している。このファイルとディレクトリを含めた構造のことをディレクトリツリーと呼ぶ。

マスターサーバに対してミラーを行なっているサーバを、1 次ミラーサーバと呼ぶ。1 次ミラーサーバのミラーを行なっているサーバを 2 次ミラーサーバと呼び、以降、3 次、4 次と続く。2 次ミラーサーバ見ると、1 次ミラーサーバがミラー元となるサーバであるため、マスターサーバと呼ぶ。よって、マスターサーバとミラーサーバには親子関係が生じ、ミラーサーバの接続関係は木構造を形成する (図 2.1)。

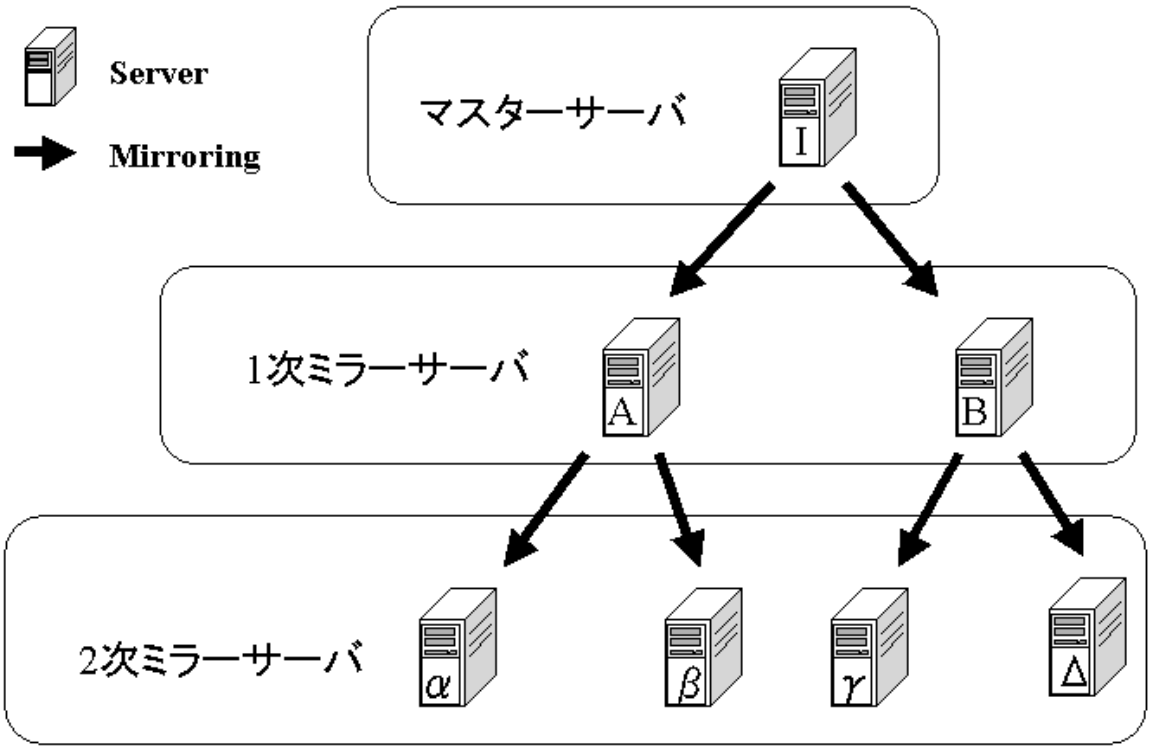

図 2.1　マスターサーバとミラーサーバの関係

2.2.1　マスターサーバ

通常、マスターサーバは、ミラーサーバを広域分散配置した場合でもアクセスが集中する傾向がある。これは、マスターサーバがサービス内容の変更を行なうため、最新のサービスを提供しているからである。そのため、マスターサーバはユーザに対して非公開な場合がある。この場合、ユーザはミラーサーバを利用するしかない。

2.2.2　ミラーサーバ

ミラーサーバは、ミラーのレベルにより以下の 2 つに分類される (図 2.2)。

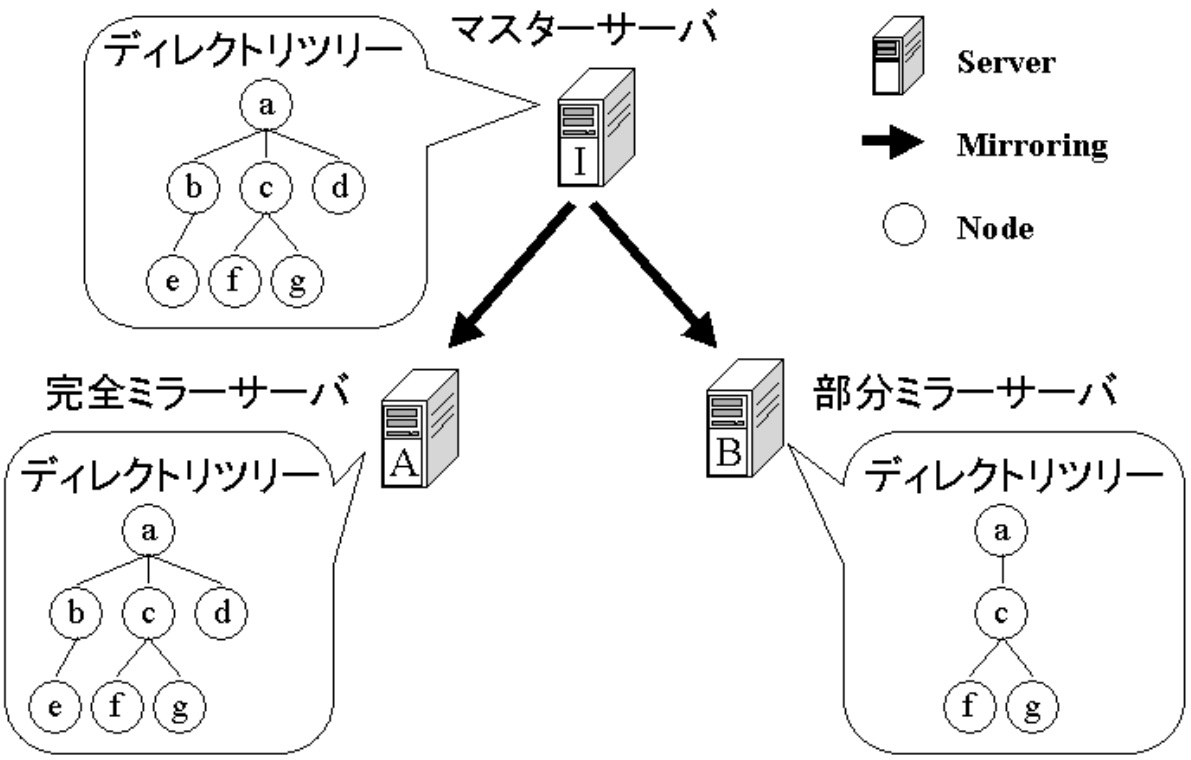

図 2.2　完全ミラーサーバと部分ミラーサーバ

- 完全ミラーサーバ

  図 2.2 の中で A のサーバのように、マスターサーバのディレクトリを完全にミラーするサーバを完全ミラーサーバと呼ぶ。完全にミラーしているため、ディレクトリツリーは

同一である。

- 部分ミラーサーバ

  図 2.2 中、B のサーバのように、マスターサーバのディレクトリの一部をミラーするサーバを部分ミラーサーバと呼ぶ。なお、マスターサーバのルートノードからの絶対位置は保持するもとのとする。図 2.2 の B のサーバのディレクトリツリーのように、必ずマスターサーバのルートノードから構成する。

ミラーサーバには、個人あるいはコミュニティの間だけで使用するため、一般公開していないものも存在する。

また、図 2.3 のように、ミラーサーバにおいてミラー対象のディレクトリとディレクトリツリーにおける相対位置が異なる場合がある。また、1 台のミラーサーバに複数のサーバをミラーする場合もある。

図 2.3 中、ミラーサーバ A は、マスターサーバ I のディレクトリツリーをノード B の下に完全ミラーを行なっている。また、ノード A の下にマスターサーバ II の完全ミラーも行なっている。

2.3 Ring Server プロジェクト

大規模なミラーサーバを積極的に活用したシステムに、Ring Server プロジェクト [*1]の Ring Server がある。

Ring Server プロジェクトでは、インターネットなどの高速ネットワーク環境を対象として、大規模なソフトウェアライブラリとソフトウェアの分散共同開発の支援を行なう共通基盤技術の研究開発を行なっている。実際に運用することによりネットワーク社会に貢献することを目的としているプロジェクトである。プロジェクトの一環としてミラーサーバ群である、Ring Server を運用している。

Ring Server は、複数のファイルサーバの集合体である。構成図を図 2.4 に示す。

[*1] http://www.ring.gr.jp/

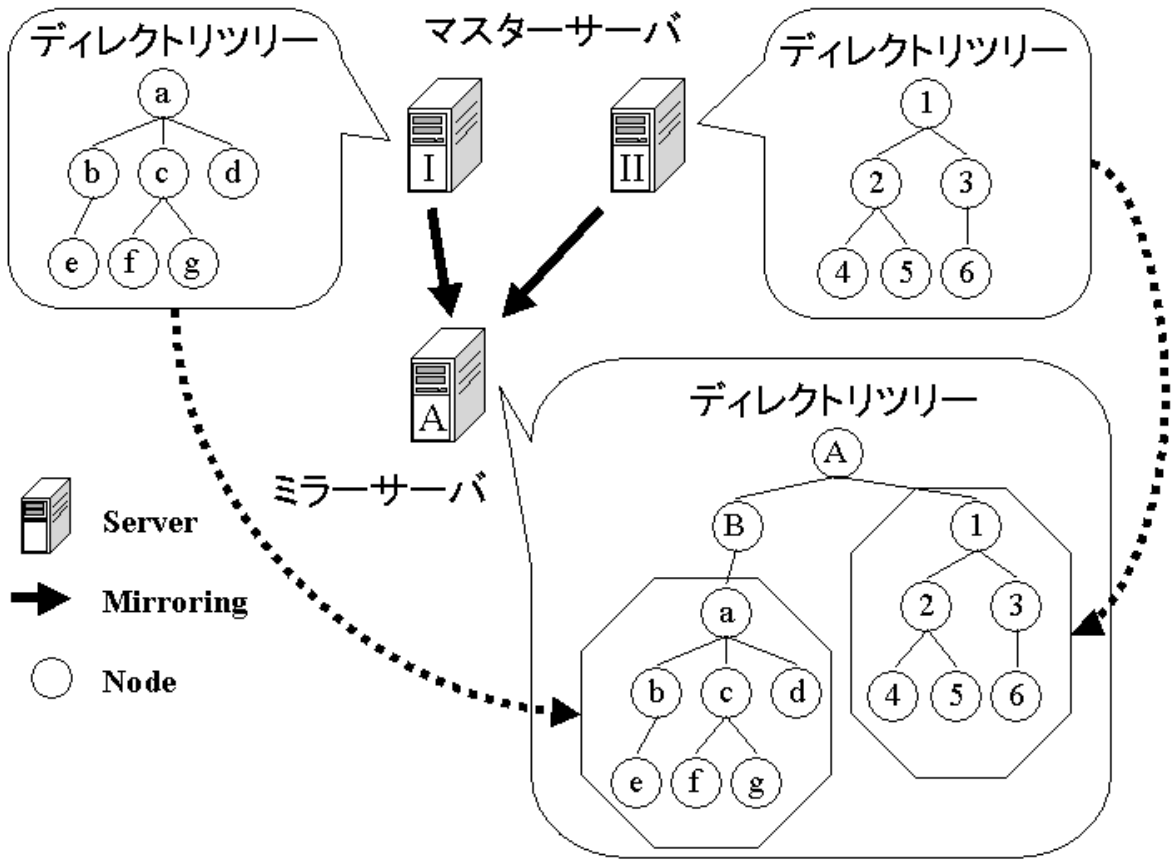

図 2.3　複数のサーバをミラー

Ring Server を構成する各サーバは、その役割によって以下の 2 つに分類できる。

- ベースサーバ (Base Server)

  それぞれ同一の内容を保持する完全ミラーサーバを Ring Server では、ベースサーバと呼ぶ。ベースサーバのうち、国内外の著名なライブラリのサーバと直接アクセスしてコピーするサーバをマスターサーバとバックアップサーバと呼ぶ。

- ミラーサーバ (Mirror servers)

  ベースサーバの部分ミラーサーバを Ring Server では、ミラーサーバと呼ぶ。部分ミラーサーバなので、ハードディスクのサイズは各サーバごとに違う。

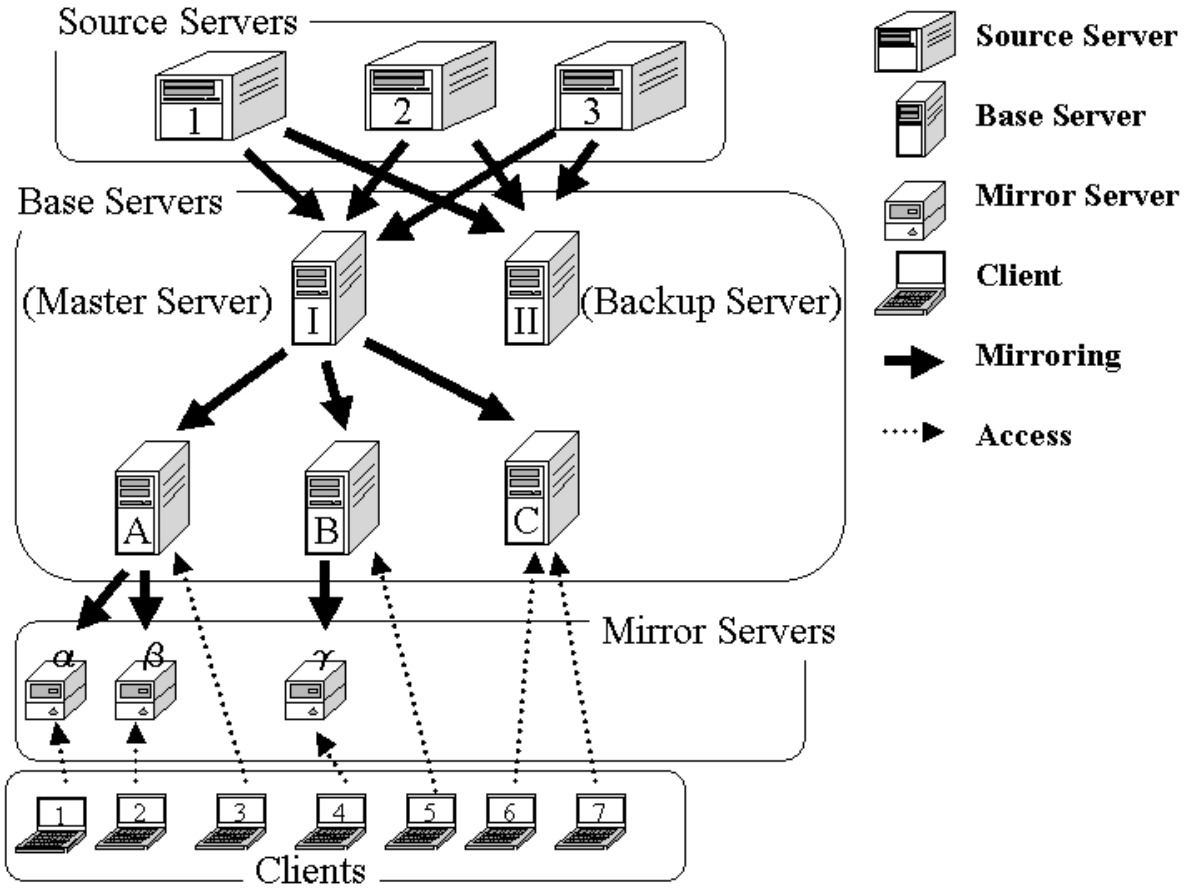

図 2.4　Ring Server の構成

2.4　ミラーサーバの問題点

ユーザが複数のミラーサーバが保持している情報コンテンツを取得しようと試みる際、以下の理由からミラーサーバを利用する方がよい。

- マスターサーバはアクセスが集中する傾向がある。
- ネットワーク的に近いミラーサーバが存在することが期待できる。

しかし、ミラーサーバを選択する前に、あらかじめユーザは情報を収集する必要がある。これは、本来、ユーザに対して不必要であるはずの負担を強いることになる。

以下にユーザがミラーサーバを選択する際に収集しておく必要のある情報を述べる。

2.4 ミラーサーバの問題点

- どのようなミラーサーバが存在するのか

  ミラーサーバの存在は、WWW、電子メール、ネットニュース等の媒体でアナウンスされる。これらのアナウンスはすべて手動で行なわれている。著名なミラーサーバであれば見つけることは簡単であるが、積極的にアナウンスを行なっていないミサーサーバを見つけるのは大変困難である。ミラーサーバの存在を確認する技術は存在しないため、ユーザが手動でその存在を確認しなければならない。

- ミラーサーバが何をミラーしているのか

  ミラーサーバは複数のサーバのミラーを行なってることが多い。しかし、ディスク容量やネットワーク帯域の制限から、すべてのミラーを行なうことは不可能である。また、ディレクトリの相対位置が異るため、あらかじめどのディレクトリにミラーしているかも調べる必要がある。これらの情報を集めるためには、実際にアクセスしてディレクトリの検査を必要とする。

- どのミラーサーバがネットワーク的に近いか

  ネットワーク的な距離は、経由するルータ数や帯域で決定する。たとえ、地理的に近くても、ネットワーク的に遠いことは十分に考えられる。また、ネットワーク上のトラフィックは刻一刻と変化するため、ネットワーク的な距離も同様に変化する。そのため、ネットワーク的な距離を把握することはかなり困難である。

- ミラーサーバの更新が遅れていないか

  多くのミラーサーバは、ある一定の間隔で更新を行なうので、更新間隔程度の遅延が生じる。また、ミラーサーバの管理者の設定ミスや、ディスク容量以上のデータの流入、ネットワークのリンク切れ等の原因によって、更新が止まってしまうこともある。よって、最新の情報コンテンツを取得するためには、一度マスターサーバにアクセスし、比較を行ない、最新の情報コンテンツであることを確認する必要がある。

以上に述べたように、ミラーサーバの利用にはユーザがさまざまな情報を収集しなくてはならないといった問題が存在する。

2.4　ミラーサーバの問題点

ユーザは、マスターサーバにアクセスすれば確実に最新の情報コンテンツを入手することができる。たとえアクセス処理やネットワークの輻輳により時間がかかっても、ミラーサーバを選択するための情報収集を省くことが出来る。

つまり、ミラーされたファイルが最新のものである保証がないことが問題なのである。

そこで、ユーザが任意の情報コンテンツを取得する際に、ユーザにとって最適なサーバを選択する手法が必要である。

第 3 章

ミラーサーバ選択手法

第 2.4 節で述べたように、ミラーサーバを利用するにはユーザがさまざまな情報を収集しなくてはならないという問題がある。

本章では、まずミラーサーバを選択する手法において満たすことが望ましい性質を述べる。次に既存のサーバ選択手法について、実装において分類し、それぞれの実装について概要を述べる。そして、それぞれの問題点を挙げる。以降でそれぞれの場合において詳細に述べる。

3.1　サーバ選択手法への要求

第 2.4 節の問題から、ミラーサーバ選択手法が満たすことが望ましい性質を挙げる。

3.1.1　ユーザの利便性

2.4 で述べたように、ミラーサーバの利用には多くの情報を収集する必要がある。しかし、マスターサーバを利用する場合を考えると、これらの情報収集は、ユーザに取って本来不必要な負担である。

そこで、出来る限り少ない負担でユーザがミラーサーバを利用できることが望ましい。

3.1.2　ファイルの最新性

ユーザがマスターサーバにアクセスを行なえば、常に最新のファイルが取得できる。しかし、ミラーサーバでは同期の遅延により、常に最新のファイルが取得できるとは限らない。

ユーザがミラーサーバを利用した際に、最新のファイルを取得できなかったとする。そして、ユーザがマスターサーバにアクセスをし直してファイルを取得することを考えると、ミラーサーバは無意味であり、かえってユーザに負担を強いてしまう。

そこで、サーバを選択する際に、最新のファイルを保持しているサーバを選択できることが望ましい。

3.1.3　ファイル取得の確実性

部分ミラーサーバやディレクトリツリーの同一性を保証しないミラーサーバにおいて、ユーザがアクセスしたサーバで確実に欲するファイルを取得できることが望ましい。また、完全ミラーサーバにおいても同期の遅延や、マシントラブルなどにより、一部のファイルが壊れていたり、存在しない場合があるので同様である。

3.1.4　スケーラビリティ

サーバを選択する際に、サーバ選択手法自体がが新たなボトルネックとなっては、ミラーサーバが無意味となる。そこで、サーバ選択手法は分散的に実装できることが望ましい。

3.1.5　サービス提供者非依存性

サーバ選択を行なうのはサービス提供者のみとは限らない。例えば、ある組織のネットワーク管理者が、その組織のネットワーク構成 (対外接続線の種類など) に応じてサーバ選択を行なうこともある。よって、サービス提供者とは独立してサーバ選択手法が利用できた方が、より適用可能範囲が広がる。

3.2　サーバ選択モデル

一般にユーザがクライアントマシンを使ってサーバにアクセスする際の流れはおおよそ以下である。

3.2 サーバ選択モデル

1. サーバ名の指定

   ユーザは、クライアントを使い、アクセスするサーバ名を指定して、アクセス要求を出す。

2. DNS による名前解決

   DNS サーバは、受け取ったサーバ名を IP アドレスに変換 (名前解決) しクライアントに渡す。

3. サーバにアクセス

   クライアントは DNS から受け取った IP アドレスを使用してサーバにアクセスを行なう。

よって、最新の情報コンテンツを保持する 1 つのサーバを自動で選択するモデルとして、以下の 3 つが考えられる。

- クライアントにおける実装
- サーバにおける実装
- DNS における実装

さらに、サーバ選択を行なう際の単位として、以下の 3 つに分類できる。

- ファイル単位
- ディレクトリ単位
- サーバ単位

本研究と本研究に関連する研究は、この中のいずれかに分類できる。

以降では、既存のサーバ選択手法において概要を述べ、第 3.1 節で挙げた性質を満たしているか評価を行なう。

3.2.1　クライアントにおける実装

クライアント、またはクライアントを使用するユーザが判断し、ミラーサーバを選択する実装である (図 3.1)。クライアントは自由に実装することが可能なため、サーバ選択の単位

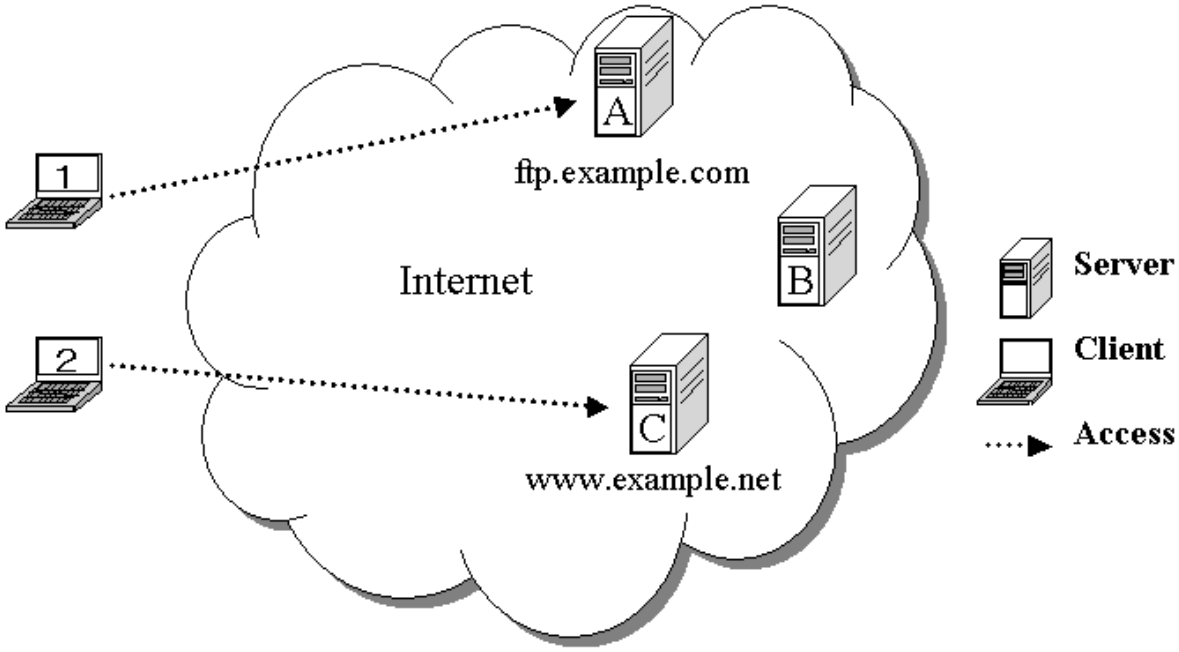

図 3.1　クライアントにおける実装

は、“ファイル単位”、“ディレクトリ単位”、“サーバ単位” のいずれも実装可能である。

しかし、当然のことながら専用のクライアントを利用することになるので、使用する OS や環境に制限ある。また、各クライアントで実装しなくてはならないため、ユーザの利便性やスケーラビリティを欠く。さらに、各ユーザが各マスターサーバに対して、ミラーサーバの存在と更新状況を把握しておかなくてはならない。この情報を一元管理することが出来ないため、各ユーザごとに調査のためのトラフィックが生じてしまう。

クライアントによるサーバ選択

クライアントにおけるサーバ選択手法の既存の実装として、GetRight[*1]、Smart Client [YCE+97] がある。GetRight、Smart Client ともに、ファイル単位の選択が可能であり、特定の単体ソフトウェアをミラーサーバ群から選択するために用いる。

- GetRight

  特定の単体ソフトウェアが存在するミラーサーバリストを入力とし、この中から近いところを選択する。
- Smart Client

  クライアントとして Java applet を用い、サーバ側の負荷情報などを受け取り、クライアントでのサーバ選択を実現している。しかし、クライアントは Java の機能が必要であるため、適応性に制限がある。

3.2.2　サーバにおける実装

サーバにおける実装では、要求を受けつけたサーバが判断を行ない、サーバを選択する方法である (図 3.2)。

この方法は要求を受け付けるサーバの判断でさまざまな状況に対応することが可能となり、柔軟性が高い。ただし、このサーバ自身がボトルネックとなる可能性が高く、スケーラビリティに欠ける。

既存の実装

サーバにおけるサーバ選択手法の既存の実装として、FTP Mirror Tracker [NH00] がある。

FTP Mirror Tracker は、FTP サーバのディレクトリツリーを解析し、各ディレクト

[*1] http://www.getright.com/

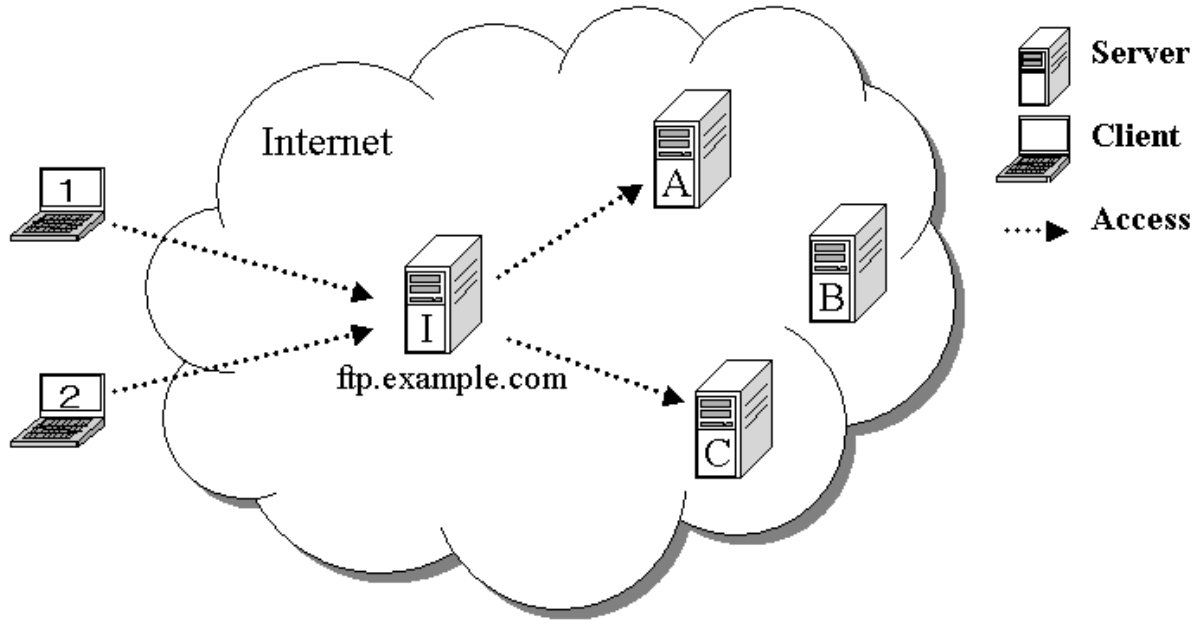

図 3.2　サーバにおける実装

リの内容を反映した一意の識別子を (MD5 [RRDS92]) を作成する。この識別子をもとに MySQL [*2]を用いてデータベースを構築する。ユーザから URL が入力されると、データベースを参照して、それと同じ中身を持つ複数の URL を返答する。

FTP Mirror Tracker は、ディレクトリごとの同一性しか判断できない。そのため、ファイル単位で相違が生じているディレクトリは利用できず、非効率である。

3.2.3　DNS における実装

DNS における実装では、要求を受けたサーバの名前解決を行なう際に、DNS サーバがサーバを選択する方法である (図 3.3)。

DNS は、インターネット上のほとんどのアプリケーションで利用される。よって、あらゆるアプリケーションでサーバ選択を行なうことが可能である。

DNS サーバは、分散的に動作するために、ボトルネックになりにくく、スケーラビリティに優れている。

[*2] http://www.mysql.gr.jp/

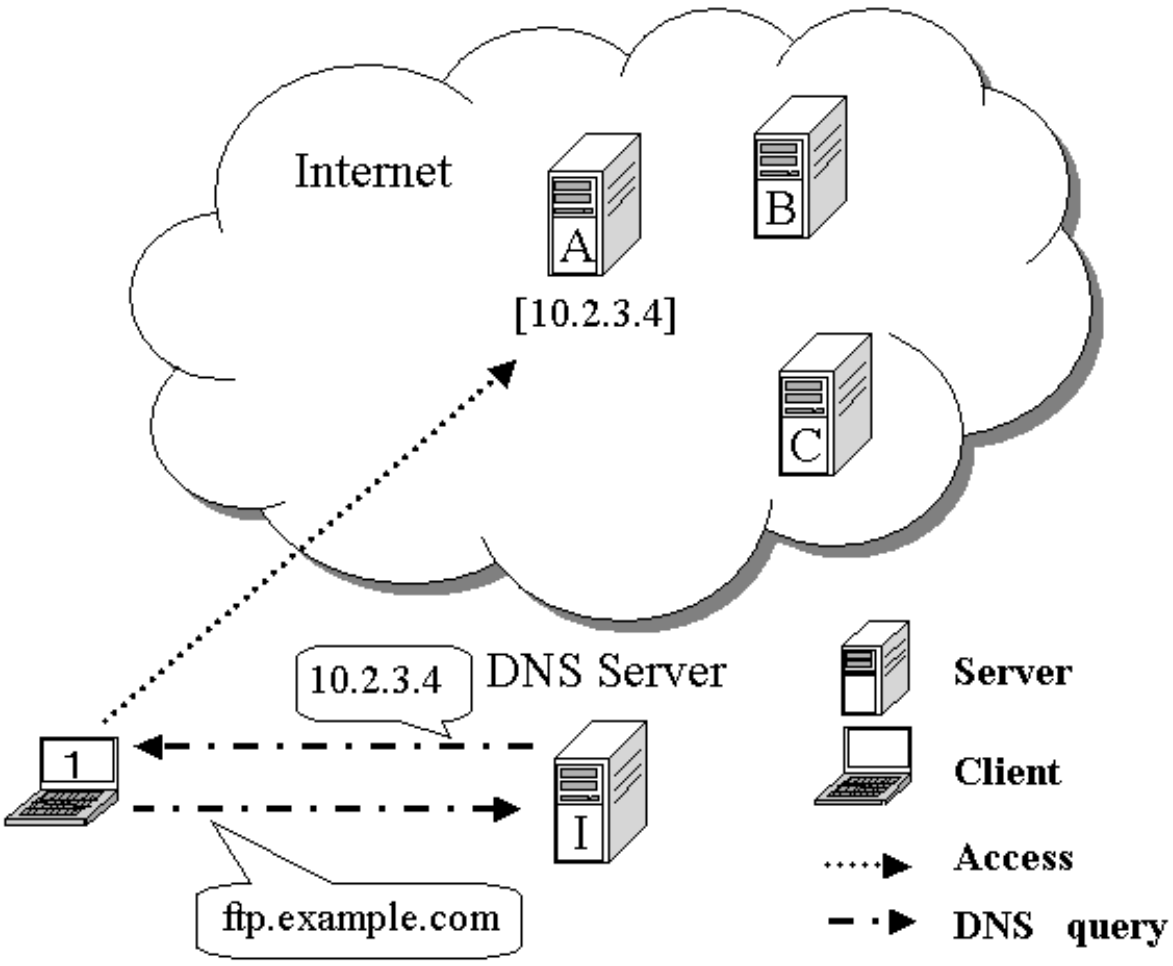

図 3.3　DNS における実装

しかし、DNS における実装では、サーバ単位でしかサーバ選択できないという問題がある。サーバの更新状況を把握する機能が存在していないため、選択したサーバの更新が遅れており、ユーザが古い情報コンテンツを取得してしまう可能性がある。

さらに、ミラーサーバがミラーを行なう場合、ディレクトリの相対位置は異なってよい。また、ミラーサーバは複数のミラーを行なうことができる。よって、ミラーサーバごとにディレクトリツリーは異なる。DNS における実装では、完全ミラーサーバを前提にしているので、ディレクトリツリーの異なるミラーサーバ群に対して対処できない。

既存の実装

DNS におけるサーバ選択手法の既存の実装として、TENBIN [下川 01][*3]、DNS Balance[*4]、DNS Trick[*5]などがある。

- TENBIN

  TENBIN は、九州大学で開発された多様な選択ポリシーを利用可能なサーバ選択機構である。これは、ローカルの DNS サーバを置き換える形で設置する。また、これとは別にサーバ群を定期的に調査するシステムを設ける。クライアントから対象サーバのサーバ名の名前解決要求があると、ローカル DNS サーバは、調査システムからの情報をもとに最適なサーバの IP アドレスを選択して返答する。この選択には多様なポリシーを利用可能であり、経路情報にもとづいた手法などが提案されている。

- DNS Balance

  DNS Balance は、ユーザの IP アドレスと、何らかの方法でサーバをランクづけした表をもとに、ユーザに適したサーバを選択する。この表は一定時間毎に読み込み直され、これにより 動的な負荷分散が可能である。

  また、この表は、サーバの負荷や CPU のロードアベレージ、ネットワークの負荷といったさまざまな情報をもとに作成可能である。

- DNS Trick

  DNS Trick は DNS サーバとクライアントの間のやりとりを監視する。1 つのサーバ名に複数の IP アドレスが割当たっていた場合に、複数の IP アドレスのサーバに対して調査を行なう。そして、次にクライアントからサーバ名に対して問い合わせがあった場合、最適なサーバの IP アドレスを返答する。よって、クライアントは最適なサーバにアクセス可能である。

  調査方法は現在 ping の応答速度と netselect の出力結果の 2 通りである。

[*3] http://www.tenbin.org/

[*4] http://openlab.jp/dns_balance/dns_balance.html

[*5] http://openlab.jp/dns_balance/dns_trick.html

3.3　各モデルにおける性質

今まで述べて来た既存の手法について、実装方法とサーバ選択の際の単位において表 3.1 にまとめた。

表 3.1　既存のサーバ選択手法

	サーバ	ディレクトリ	ファイル
クライアント			GetRight Smart Client
サーバ		FTP Mirror Tracker	
DNS	TENBIN DNS Balance DNS Trick		

各モデルにおける性質をまとまると、次のようになる。

- クライアントにおける実装

  クライアントにおける実装では、必ず専用のクライアントを導入しなくてはならず、ユーザの利便性が悪く、スケーラビリティも欠く。しかし、クライアントでの実装であるため、自由に実装できる点が利点である。

- サーバにおける実装

  要求を受けたサーバ自身が判断を行ない、サーバ選択を行なえるといった利点がある。しかし、このサーバ自身がボトルネックになる可能性が高く、スケーラビリティに欠ける。

- DNS における実装

  ユーザは、サーバ選択手法を利用するしないに関わらず、ドメインネームの名前解決のために DNS を使用するため、ユーザの利便性は高い。また、DNS サーバは分散的に動作するため、ボトルネックになりにくく、スケーラビリティも高い。しかし、サーバ単

位での選択しかできないため、同一性の保証されていないミラーサーバや部分ミラーのサーバを選択することが不可能である。よって、ファイルの最新性、ファイル取得の確実性は低い。

さて、ミラーサーバの利用の一番の問題は、ユーザが多くの情報を収集する必要があることは、第 2.4 節で述べたとおりである。つまり、サーバ選択手法において、ユーザの利便性の向上が優先すべきことである。

したがって、ユーザの利便性が高く、スケーラビリティが優れていることから、DNS における実装を行なったサーバ選択手法を提案する。

第 4 章

DNS を用いた部分ミラーサーバ群におけるサーバ選択手法

前章で述べたとおり、既存のサーバ選択手法は、第 3.1 節で挙げた性質のいずれかを満たすことができない。

そこで、本章では、これらの性質を満たす、サーバ選択手法を提案する。まず、本手法についての概要を述べる。次に本手法におけるサーバ選択の動作について述べ、最後に、ユーザ側から見たサーバ選択の動作について説明する。

4.1 DNS を用いたサーバ選択手法

DNS における実装では、サーバ単位での選択しかできないため、同一性の保証されていないサーバや、部分ミラーサーバに対してサーバを選択することが不可能である。

そこで、ドメインネームにディレクトリツリーの情報 (以降、ディレクトリ情報と呼ぶ) を付与し、ファイル単位によるサーバ選択手法を提案する。

4.1.1 特徴

提案する手法は、以下の 4 つの特徴がある。

- ファイル単位によるサーバ選択

  本手法では各ミラーサーバのディレクトリ情報を収集し、各ファイルに対してサーバ

選択を行なう。そのため、ディレクトリ情報の同一性が保証されていないサーバや部分サーバに対してもサーバ選択を可能とする。

- ファイル取得の確実性

  アナウンスされているサーバ名とディレクトリ情報を組み合わせることで任意のファイルを保持しているサーバを選択することが可能である。

- ファイルの最新性

  そして、互いのディレクトリ情報を比較し、最新のファイルに対する要求にだけ応答を返すことで、ファイルの最新性を保証する。

- ユーザの利便性とスケーラビリティ

  DNS を用いてサーバ選択手法の実装を行なうため、ユーザにサーバ選択手法を意識せずにミラーサーバを利用可能である。また、インターネット上のほとんどすべてのアプリケーションにおいて DNS は利用されているため、スケーラビリティが高い。

4.2　サーバ選択の動作

提案するサーバ選択手法について述べる。

提案する手法のおおまかな流れは次のようになる。

1. ミラーサーバのディレクトリ情報を収集

2. RDN を生成

3. RDN の最新性を検証

4. RDNDN を生成

5. DNS のレコードを更新

以下では、各ステップについて詳細を述べる。

4.2.1　ミラーサーバのディレクトリ情報を収集

各ミラーサーバからディレクトリ情報を収集する。

収集するディレクトリ情報は、ディレクトリツリーとそれに含まれる各ファイルの最終更新時間、ファイルサイズとする。

4.2.2　RDN の生成

収集したディレクトリ情報から RDN を生成する。

収集したディレクトリ情報から、ディレクトリツリーのノード (以降、ディレクトリノード) を逆順に並び替え、各ノード間を “.”(ドット) で区切って再構成する (図 4.1)。再構成したものを Reverse Directory Node(以降、RDN) とする。

また、RDN は “–” と “.”(ドット) 以外の記号の使用を認めない。これは、RDN をドメインネームに使用するためで、ドメインネームの生成規則による。

よって、本過程において “–” と “.” 以外の記号をすべて、“–” に変換する。

```
        /pub/FreeBSD   ⟶   FreeBSD.pub

/pub/GNU/g++.README    ⟶   g--.README.GNU.pub
```

図 4.1　RDN の生成

4.2.3　RDN の最新性を検証

RDN と更新情報リストとの比較を行なう。

- RDN が最新のものであるか、更新情報リストに存在しないものであれば、更新情報リ

ストを更新する。

- RDN が更新情報リストよりも古いものであれば、RDN を破棄し、2 に戻る。

4.2.4　RDNDN を生成

RDN より、RDNDN を生成する。

RDNDN は、RDN Domain Name の略で、RDN の末尾にユーザにアナウンスするサーバ名を追加したものを指す。

図 4.2 に RDN が “FreeBSD.pub” で、サーバ名が “mirror.server.org” である場合の RDNDN の例を示す。

```
Domain Name   :   mirror.server.org
        RDN   :   FreeBSD.pub
                  ↓
  FreeBSD.pub.mirror.server.org
```

図 4.2　RDNDN の生成例

4.2.5　DNS サーバのレコードを更新

RDNDN をディレクトリ情報を取得したサーバ名の別名として、CNAME レコードを作成し、DNS を更新する。

CNAME(canonical name) は、ドメインネームの別名を定義する DNS 資源レコードである。

通常、1 つのサービスを提供しているサーバのドメインネームは、ユーザに対してアナウンスしているため、サービスが終了するまで変更することはない。このサーバのマシントラブルやメンテナンスの際には、同じドメインネームで、別の IP アドレスを割り当てたサーバを用いることが多い。

よって、本過程では、RDNDN を各ミラーサーバの別名としてレコードを作成する。

図 4.3 に CNAME レコードのフォーマットを示す。

```
owner ttl class CNAME canonical-domain
```

図 4.3　CNAME レコード

以下で、CNAME レコードのそれぞれの要素について説明する。

- `owner`

  正規名の別名を指定する。本過程では、RDNDN を指定する。

- `ttl`

  レコード生存時間 (TTL:Time To Live) を指定する。中間のサーバがレコードをキャッシュしておく時間。中間サーバはキャッシュしているレコードの TTL を過ぎると、レコードを再度検索し直す。頻繁にレコードが変更する場合には短く設定する。

- `class`

  現在広く用いられているクラスは、IN(Internet) である。他にもクラスは存在するが、現在は IN 以外は広く用いられていない。

- `canonical-domain`

  owner の正規名を指定する。本過程では、各ミラーサーバ名を指定する。

図 4.4 に RDNDN が “FreeBSD.pub.mirror.server.org.” で、ミラーサーバが “mirror.example.com” の場合についての CNAME レコードを示す。

```
FreeBSD.pub.mirror.server.org 60 IN CNAME mirror.example.com
```

図 4.4　CNAME レコード

いくつかのミラーサーバで、ミラー対象が同じ場合、例えば、ミラーサーバ “mirror.example.com” と “ring.example.net” とで、それぞれ “/pub/linux” をミラー対象としていた場合、RDNDN が重複する。このように複数の CNAME レコードが存在する場合

には、複数の正規名を循環させて使用できる。この機能をラウンドロビン (round robin) と呼ぶ。

ラウンドロビン機能を用いることで、複数のサーバに要求を振り分けることができる。よって、サーバの負荷分散を実現できる。

図 4.5 に、本手法によるサーバ群の動作例を示す。

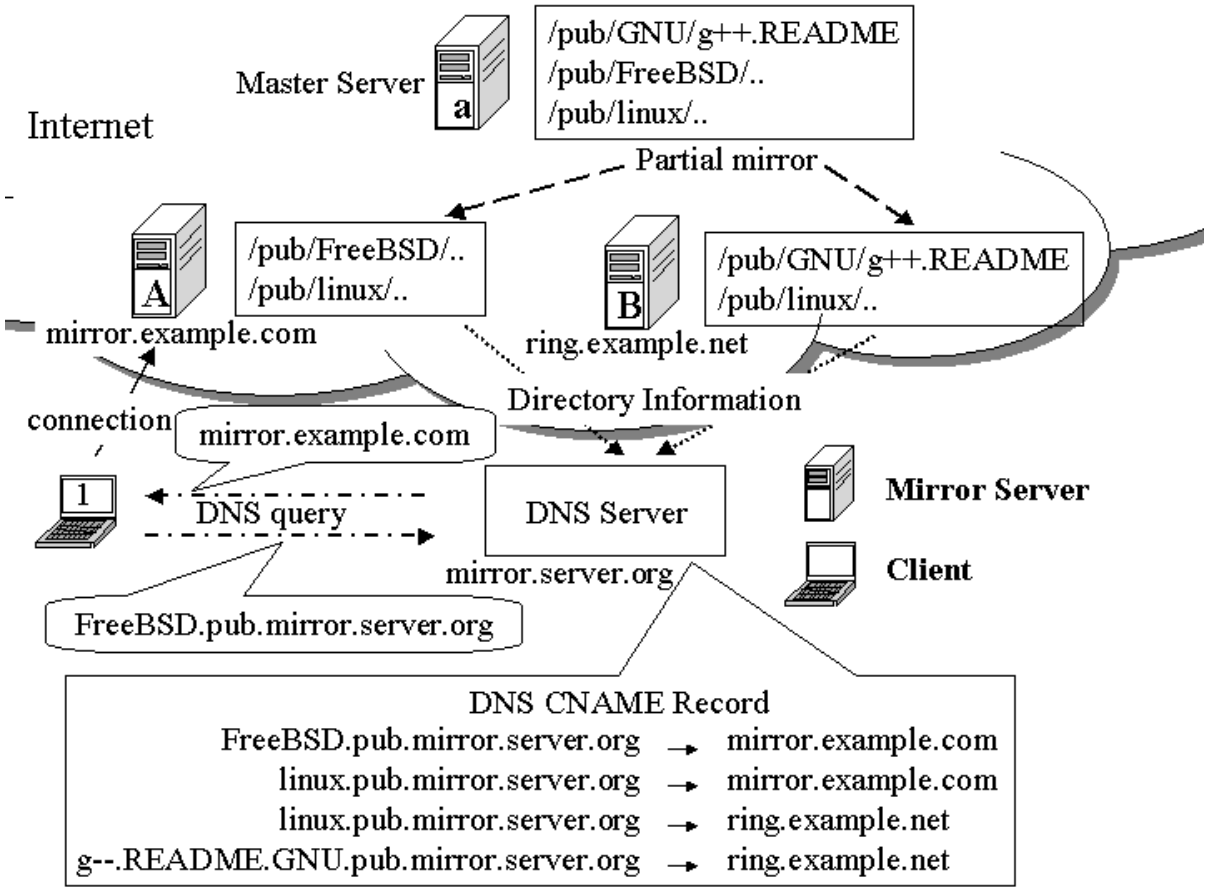

図 4.5　提案手法のサーバ群の動作

4.3　ユーザから見たサーバ選択

ユーザが、FreeBSD ディレクトリ以下にある情報コンテンツを “ftp [PR85]” を利用して取得しようとした場合を例に示す。

ユーザはサーバ名 “mirror.server.org” と、ディレクトリ情報”/pub/FreeBSD/” を知っ

ており、RDNDN(FreeBSD.pub.mirror.server.org) を利用したことがあるものとする。

4.3.1　サーバ名を指定

ユーザは、ftp クライアントを使って RDNDN にアクセス要求を出す (図 4.6)。

```
> ftp  FreeBSD.pub.mirror.server.org
```

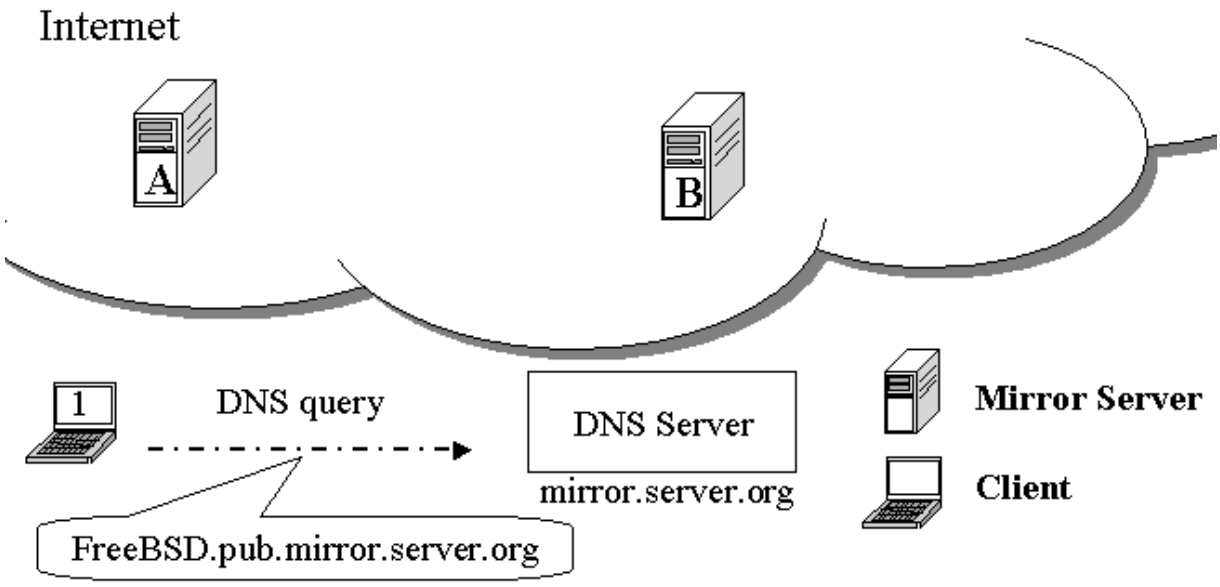

図 4.6　RDNDN へのアクセス要求

4.3.2　RDNDN の名前解決

DNS サーバ “mirror.server.org” は、RDNDN の名前解決を行なう (図 4.7)。“FreeBSD.pub.mirror.server.org” を引き、CNAME レコードから正規名 “mirror.example.com” をクライアントに返す。

4.3.3　サーバにアクセス

クライアント 1 は、4.3.2 で受け取ったサーバに対してアクセスを行なう (図 4.8)。

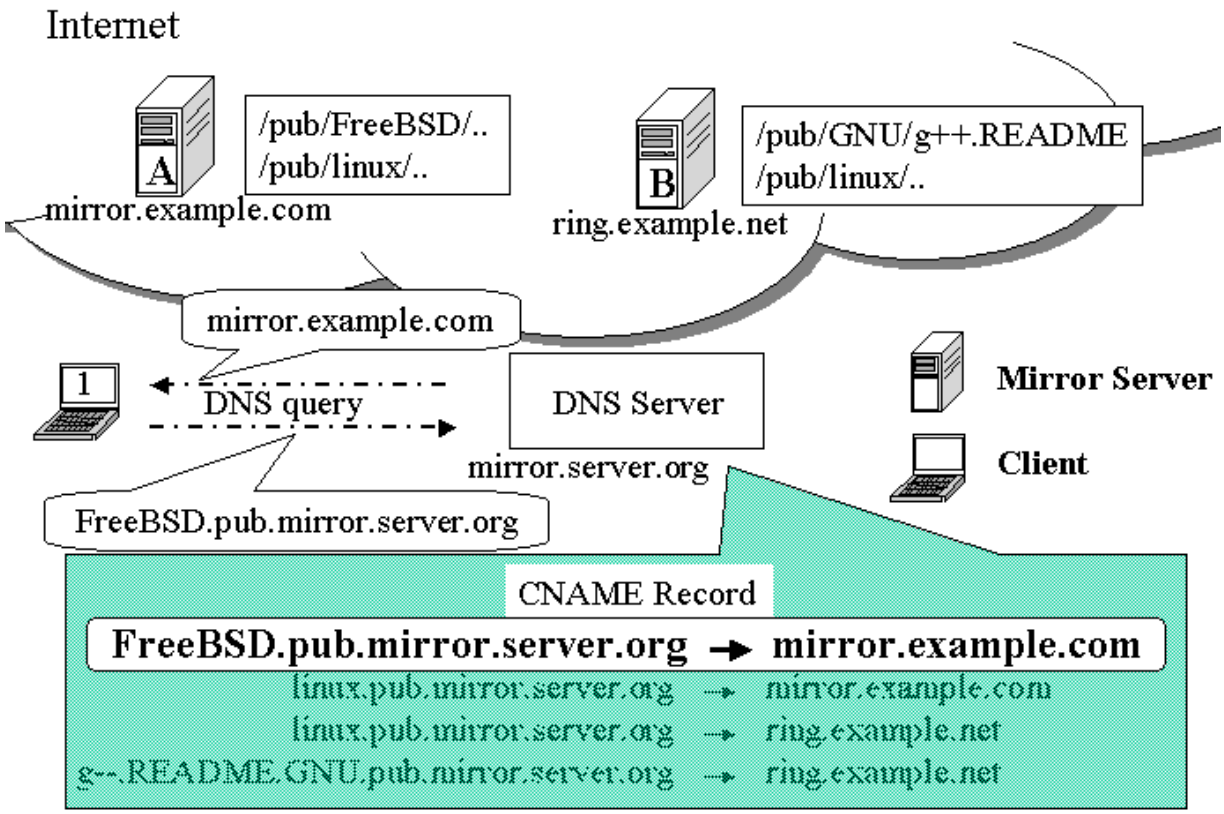

図 4.7　DNS による名前解決

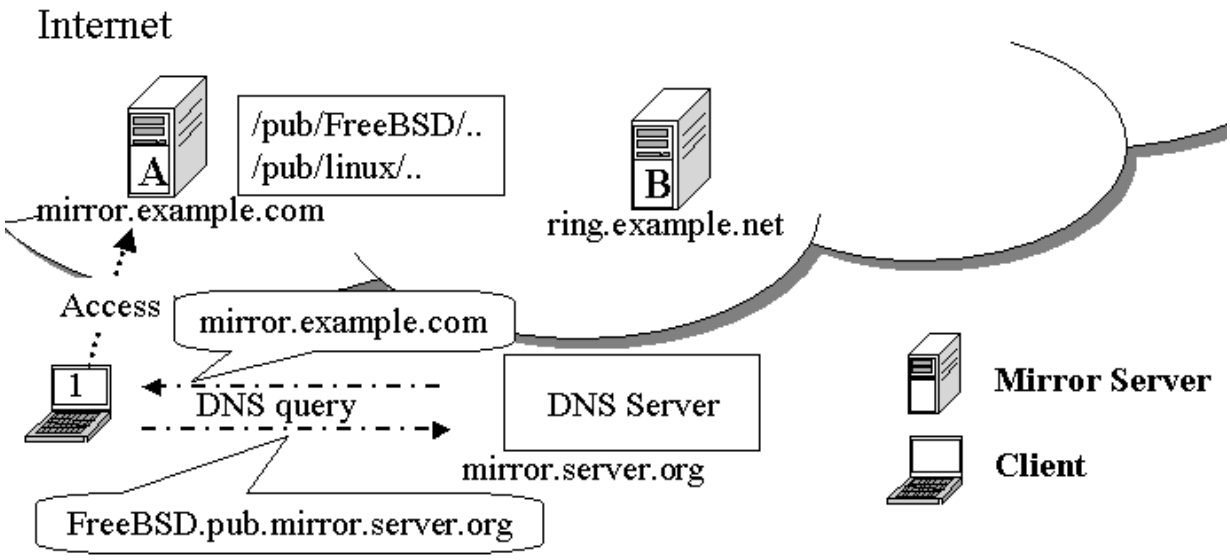

図 4.8　サーバにアクセス

ここでは省略しているが、4.3.2 では、サーバ名 “mirror.example.com” を受け取るため、実際には DNS に問い合わせるなどして、サーバ名を IP に変換する必要がある。

第 5 章

プロトタイプシステムの実装

本研究では第 4 章で提案したサーバ選択手法について、プロトタイプを実装した。

本章では、実装したシステムの詳細なプロセスについて述べる。はじめに、実装要求を述べる。次に、実装環境を示し、実装したシステムの概要を述べる。最後にシステムについての詳細な説明を行なう。

5.1 サーバ選択における要求

実装するサーバ選択手法における要求は以下である。

- ファイルの最新性を保証
- ディレクトリツリーの異なるミラーサーバ群も対応
- スケーラビリティがある
- 既存のクライアントを用いてアクセス可能
- 導入コストが低い

5.2 実装環境

5.1 より、最小限のツールを使っての実装を試みた。

表 5.1 に示すような環境で実装を行なった。

表 5.1 実装環境

CPU	Pentium II 400MHz
Memory	128 MB
OS	FreeBSD4.6.2-RELEASE
DNS	BIND9-9.2.1
Perl	Perl5.005
gzip	gzip-1.2.4

5.3 概要

提案するサーバ選択手法の大まかな流れは次のようになる。

1. 設定ファイルを読み込む
2. 更新情報リストの読み込む
3. ディレクトリ情報を取得
4. RDN を生成
5. RDN と更新情報リストを比較
6. RDNDN を生成
7. DNS のレコードを更新
8. 更新情報リストを更新

以降でそれぞれの詳細について述べる。

5.4 設定ファイルを読み込む

本過程では設定ファイルを読み込む。

デフォルトでは、コマンドを実行するカレントディレクトリにある、“rdndns.conf” というファイル名を設定ファイルとして読み込む。また、コマンド実行時のオプションとして、

設置ファイルを指定することも可能である。

設定ファイルは平文で図 5.1 のフォーマットに従い記述する。

パラメータラベル　＝　値

図 5.1　設定ファイルフォーマット

設定可能なパラメータについて以下に記述する。

- **name**

  ユーザにアナウンスするドメインネームを指定する。主に、RDNDNS を実装してる DNS サーバのサーバ名を指定する。

- **server**

  サーバ選択の対象となるサーバ名を指定する。

- **lslr**

  すぐ前の “serever” ラベルで指定したサーバが保持する “ls-lR” ファイルを “/” からの絶対パスで指定する。

  デフォルトでは、“/ls-lR.gz” が指定される。

  このため、“name” ラベルを記述してない場合や、値が空の場合でも動作に影響しない。

なお、“name” と “server” のパラメータは必須である。また、行頭に “#” をつけると、コメント行とみなし、その行を無視する。

図 5.2 に設定ファイルの記述例を示す。

5.5　更新情報リストの読み込む

特に専用のデータベースを用いず、平文でのテキストファイルとして各ファイルの情報を管理している。

更新情報リストファイルのフォーマットを図 5.3 に示す。

各要素について以下で説明する。

```
# ユーザにアナウンスするドメインネーム
name     =   mirror.server.org
# サーバ選択で対象とするサーバ名
server   =   mirror.example.com
# 対象とするサーバが保持する ls-lR ファイルまでの絶対パス
lslr     =   /ls-lR.gz


server   =   ring.example.net
lslr     =   /ls-lR.gz
         ⋮
```

図 5.2　設定ファイルの記述例

```
RDN    最終更新時間    ファイルサイズ
```

図 5.3　更新リストファイルフォーマット

- RDN

  RDN を表す。詳細は、5.7 で述べる。

- 最終更新時間

  RDN が指すファイルの最終更新時間を表す。西暦　月　日　時　分　を “YYYYMMDDhhmm” のように 12 桁の数字で表す。最終更新時間が 2003 年 1 月 5 日 14 時 30 分であるなら、“200301051430” と表す。

- ファイルサイズ

  RDN が指すファイルのファイルサイズを表す。

5.6　ディレクトリ情報を取得

第 4.2.1 節でも述べたように、対象とするサーバのディレクトリ情報を取得する。

ls-lR ファイルについて以下で詳しく述べる。

- ls-lR ファイル

  ls-lR ファイルは、UNIX の基本的な “ls” コマンドにオプションとして、“-lR” をつけて実行した結果をファイルに書き出したものである。“l” オプションがカレントディレクトリにあるファイルの詳細な情報を出力する。さらに、“R” オプションでカレントディレクトリ以下のすべてのディレクトリとファイルを出力し、“ls -lR” コマンドを実行することで、実行したカレントディレクトリ以下のディレクトリ情報を取得できる。

  図 5.4 に ls-lR ファイルの出力結果を載せる。図 5.4 の 1 行目は “ls -lR” コマンドが実

```
 1 ./pub:
 2 total 44
 3 drwxr-xr-x 13 ftpadm staff 512 Jan 25 14:21 FreeBSD
 4 drwxr-xr-x 224 ftpadm staff 21504 Jan 25 14:22 GNU
 5                                  ⋮
 6 drwxr-xr-x 7 ftpadm ftpadm 512 May 10 2001 text
 7
 8 ./pub/FreeBSD:
 9 total 12552
10 drwxr-xr-x 6 ftpadm ftpadm 512 Jan 19 02:30 CERT
11 lrwxrwxrwx 1 ftpadm ftpadm 15 Feb 7 2001 CTM -> development/CTM
12                                  ⋮
```

図 5.4　ls-lR ファイルの出力結果

行されたカレントディレクトリを表している。

2 行目の “total” では、カレントディレクトリのファイルサイズの合計値が、512 バイトブロック単位で表示される。

3 行目からは、ファイルに関する情報を表している。左からそれぞれ、“ファイルモード”、“リンク数”、“所有者名”、“所有グループ名”、“ファイルのバイト数”、“月の短縮形”、“最終更新が行なわれた際の日付”、“時”、“分”、“パス名” である。なお、11 行目のパス名の後にある、“− > development/CTM” はファイルがシンボリックリンクファイルである。そのため、パス名の後に、“− > リンク先パス名” として記述されている。

カレントディレクトリのファイルの出力が終ると、空行が入る。その後で、“ls -lR” コマンドを実行したカレントディレクトリ以下のディレクトリについての表示に変わる。以降カレントディレクトリ以下のすべてのディレクトリ情報を表示する。

本手法では、ls-lR ファイルから各ファイルごとにファイル名、ファイルサイズ、最終更新時間とディレクトリ位置を取得してサーバ選択に用いる。

設定ファイルの “server” パラメータで指定されたサーバから、 FTP を用いて “lslr” パラメータで指定された ls-lR ファイルを取得する。

取得した ls-lR ファイルが “gz” や “Z” で終る場合には、“gunzip/uncompress” を行なう。

また、ls-lR ファイルの取得に失敗した場合には、再度 FTP による取得を試みる。

再び取得できない場合には、指定されたサーバで何かしらのトラブルがあったものと判断する。そして、DNS から該当するサーバのレコードを削除する。第 5.1 節より、本手法では各サーバでファイルサーバのファイル管理に使用される “ls-lR” ファイルを収集することとする。

5.7 RDN を生成

第 4.2.2 節でも述べたように、取得したディレクトリ情報をもと RDN の生成をおこなう。

5.6 で取得した ls-lR ファイルを分析する。ls-lR ファイルから各ディレクトリの絶対パスと、その直下にあるファイルのファイル名、最終更新時間、ファイルサイズを取得する。取得した最終更新時間とファイルサイズについては 5.8 で取り扱う。

本過程では、ディレクトリの絶対パスとファイル名から次の手順で RDN を生成する。

1. 取得したディレクトリの絶対パスを、“/” で分割する。
   この際に “/” は削除される。
2. 分割したディレクトリ名を逆順に並べ替える。
3. ディレクトリ名の区切りとして “.”(ドット) を用いる。
4. 取得したファイル名の末尾に、3 で再構成したものを追加する。
5. 4 で生成したものに含まれる “–” と 1 つの “.”(ドット) 以外の記号をすべて “–” に変換する。

5.8　RDN と更新情報リストを比較

第 4.2.3 節で述べたように、ファイルの最新性を保証するために、RDN と更新情報リストを比較する。

本過程では 5.7 で生成した RDN と RDN に関する最終更新時間とファイルサイズを用いて、更新情報リストと比較を行ない、ファイルの最新性を保証する。

最終更新時間が遅い方が最新のものであると評価する。

また、最終更新時間が更新情報リストと同一の場合は、ファイルサイズが大きい方を優先する。多くの場合、ミラー対象のファイルサイズの変動は考えられない。そして、ファイルサイズが小さい場合は、ミラーリングの際にファイルの取得に失敗したことが考えられる。よって、RDN と更新情報リストの最終更新日が同一で、RDN の方がファイルサイズが小さい場合は RDN を破棄する。

手順を以下に示す。

1. RDN が更新情報リストに含まれているか、チェックを行なう。
   - 更新情報リストに RDN が含まれているなら、更新情報リストから RDN、最終更新時間、ファイルサイズを取得する。
   - 更新情報リストに RDN が含まれていないなら、RDN は新たに追加されたファイ

ルと見なし、4 へ。

2. RDN の最終更新時間と 1 で取り出した最新更新リストの最終更新時間との比較を行なう。
   - RDN の最終更新時間の方が更新情報リストの最終更新時間よりも新しいなら、4 へ。
   - RDN と更新情報リストの最終更新時間が同じであるなら、3 へ。
   - RDN の最終更新時間の方が古いなら、RDN を破棄し、比較を終了する。
3. RDN のファイルサイズと 1 で取り出した最新更新リストのファイルサイズとの比較を行なう。
   - 最終更新時間が同じで、RDN のファイルサイズの方が大きいなら、4 へ。
   - 更新情報リストのファイルサイズの方が大きい、もしくは等しいなら、RDN を破棄し、比較を終了する。
4. RDN が最新のものであることから、更新情報リストの該当する RDN の情報を更新する。

5.9　RDNDN を生成

第 4.2.4 節で述べたように、RDN の末尾にユーザにアナウンスを行なうサーバ名を追加し、RDNDN を生成する。

5.10　DNS のレコードを更新

第 4.2.5 節で述べたように、5.9 で生成した RDNDN をドメインネームとして登録を行なう。

本実装では、DNS の更新に RFC2136 で提唱された DNS の動的更新 [VTRB97] を使用する。DNS の動的更新では、サーバが権威を持つゾーンの資源レコードを、許可された更新者が追加および削除が可能である。

5.10 DNS のレコードを更新

本過程では、BIND [AL02] に標準添付されているコマンドラインプログラム “nsupdata” を使って DNS の動的更新を行なう。nsupdate は、1 行ずつのコマンドを読み込んで、それを使って更新メッセージに変換する。コマンドは、標準入力または、nsupdate の引数として指定したファイルから読み込まれる。

nsupdate で使用できるコマンドは、次の 6 つがある。大別すると更新の前提条件になるコマンドと、実際にレコードを更新するコマンドに分類できる。

- 更新の前提条件となるコマンド
  - `prereq yxrrset domain-name [class] type [rdata]`

    指定されたドメインネームが所有者で、かつ、指定されたレコード型の RR セットが存在することを、前提条件とする。
  - `prereq nxrrset domain-name [class] type`

    指定されたドメインネームが所有者で、かつ、指定されたレコード型の RR セットが存在しないことを前提条件とする。
  - `prereq yxdomain domain-name`

    指定されたドメインネームが存在することを前提条件とする。
  - `prereq nxdomain domain-name`

    指定されたドメインネームが存在しないことを前提条件にする。
- 更新に関するコマンド
  - `update delete domain-name [class] [type] [rdata]`

    指定されたドメインネームを削除する。`type` を併せて指定した場合には、指定された RR セットを削除する。さらに rdata も併せて指定された場合には、domain-name、type および rdata と一致するレコードを削除する。
  - `update add domain-name ttl [class] type rdata`

    指定されたレコードをゾーンに追加する。レコード型と資源レコード特有のデータに加えて、TTL も指定しなければならない。クラスの指定はオプションであり、

デフォルトは IN になる。

引数となる domain-name、ttl、class、type、rdata について、次で述べる。

- `domain-name`

  ドメインネームを入力する。本実装においては、RDNDN を指定する。

- `ttl`

  TTL(レコードの生存時間) を指定する。本実装においては TTL を短く設定する。これは、中間のネームサーバがアドレスをキャッシュしていてもキャッシュはすぐにタイムアウトし、中間のネームサーバは、再びドメインネームを検索するからである。本実装においては、一定時間でレコードを更新するため、キャッシュによる誤動作を回避するのが目的である。

- `class`

  デフォルトでは IN(Internet) になる。他にもクラスは存在するが、現在は IN 以外は広く用いられていない。

- `type`

  資源レコードのタイプを指定する。本実装においては、第 4.2.5 節より、“CNAME” と指定する。

- `rdata`

  type によるそれぞれの資源を入力する。本実装においては、第 4.2.5 節より、各ミラーサーバ名を指定する。

今回の実装では、レコード削除のために “update delete” を、追加のために “update add” を使用する。

各 RDNDN に対するコマンドは、対象とするミラーサーバごとに、レコードの削除ファイル “delete.mirror-name ” と追加ファイル “add.mirror-name ” を作成する。mirror-name には各ミラーサーバ名を指定する。

ミラーサーバ名を “ring.example.net” とし、“ring.example.net” は、“/pub/FreeBSD”、

“/pub/GNU”、“/pub/X”、“/pub/doc” のディレクトリ情報を保持している。そして、ユーザにアナウンスするサーバ名を “mirror.server.org” とした場合のレコード削除ファイルとレコード追加ファイルの例をそれぞれ図 5.5 と図 5.6 に示す。レコード追加ファイルで指定する TTL は、60 とする。

```
update delete FreeBSD.pub.mirror.server.org.  IN CNAME ring.example.net
update delete GNU.pub.mirror.server.org.  IN CNAME ring.example.net
update delete X.pub.mirror.server.org.  IN CNAME ring.example.net
update delete doc.pub.mirror.server.org.  IN CNAME ring.example.net
```

図 5.5　レコード削除ファイル “delete.ring.example.org” の例

```
update add FreeBSD.pub.mirror.server.org.  60 IN CNAME ring.example.net
update add GNU.pub.mirror.server.org.  60 IN CNAME ring.example.net
update add X.pub.mirror.server.org.  60 IN CNAME ring.example.net
update add doc.pub.mirror.server.org.  60 IN CNAME ring.example.net
```

図 5.6　レコード追加ファイル “add.ring.example.net” の例

作成したファイルを使って、nsupdate を実行する。先に、レコードの削除を行なう。この際にレコード削除ために読み込むファイルは、今回作成した “delete.ring.example.net” ではなく、前回作成した “delete.ring.example.net.old” を使用する。これは、現在 DNS に登録しているレコードは、前回追加したレコードであるため、今回作成したレコード削除ファイルを使用すると、前回の更新時との差分のレコードが残ってしまう。したがって、今回の更新の際に削除された RDNDN のレコードも残っているため、今回作成したレコードではなく、前回作成したレコードファイルを用いてレコードの削除を行なうのである。

最後に、レコードの追加を行ない、DNS 資源レコードの更新が完了する。

5.11　更新情報リストを更新

次回、実行時に参照できるように更新情報リストをファイルに書き出す。第 5.4 章で述べたフォーマットに従い、書き出しを行なう。

5.12　プロトタイプシステムの評価

本提案を実装したプロトタイプシステムについての評価を行なう。

5.12.1　実験内容

実装したシステムを表 5.2 の ls-lR ファイルを用いて実験を行なった。

表 5.2　ls-lR ファイル

ミラー対象	ファイルサイズ	保持ファイル数
/pub/GNU	2179860	30603

DNS サーバとアナウンスするサーバ名を “ring.kikuken.org” とし、選択対象のサーバを “mirror.example.com” とした。

単一のサーバを対象としての実験

設定ファイル

本実験では、図 5.7 に示す設定ファイルを用いた。

プロトタイプシステムの実行

設定ファイル “rdndns.conf” があるカレントディレクトリにおいて、プロトタイプシステムのコマンド “rdndns” を実行した。

```
# ユーザにアナウンスするドメインネーム
name     =   ring.kikuken.org
# サーバ選択で対象とするサーバ名
server   =   mirror.example.com
# 対象とするサーバが保持する ls-lR ファイルまでの絶対パス
lslr     =   /ls-lR
```

図 5.7　実験に用いた設定ファイル “rdndns.conf”

```
> ./rdndns
```

なお、実行するユーザは、“nsupdate” コマンドを実行する権限を持つものとする。また、“rdndns” の実行は ftp によるファイル取得をしないものとする。

プロトタイプシステム “rdndns” の実行画面を図 5.8 に示す。

サーバ選択手法の利用

ここでは、“rdndns” コマンド実行後に、DNS サーバである、“ring.kikuken.org” にドメインネームの問い合わせを行なう。

ドメインネームの問い合わせには、“dig(Domain Information Groper)” を用いた。図 5.9 に “dig GNU.pub.ring.kikuken.org.” を実行した結果を示す。

詳細は省くが、図 5.9 の 11 行目に “ANSER SECTION:” とあり、12 行目に “GNU.pub.ring.kikuken.org” の CNAME として “mirror.example.com.” が返されていることが分かる。

以降、“/pub/GNU/” 以下のファイルについて任意に選択した、次のファイルに対して問い合わせを行なった。

- “/pub/GNU/GNUinfo/Audio/README”

```
========================================

Configure file:  rdndns.conf

DNS Server FQDN : server.ring.

Mirror Server :  ring.example.com

ls-lR path :  /ls-lR.gz

End of get servers


ftp ftp://mirror.example.com./ls-lR

lslr = ls-lR

========================================

nsupdate delete.ring.example.com.old


nsupdate add.ring.example.com.

========================================
```

図 5.8　“rdndns” 実行画面

問い合わせの結果を図 5.10 に示す。

- “/pub/GNU/gcc/gcc-2.95/gcc-g++-2.95.tar.gz”

  問い合わせの結果を図 5.11 に示す。

  なお、ここでは、RDN に変換の際に、“+” を”−”に変換しなければならない。

  したがって、RDNDN は、“gcc-g—2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org”

  となる。

この実験においてサーバ選択手法の利用において予想通りの動作が行なわれている。

```
 1 > dig GNU.pub.ring.kikuken.org.
 2
 3 ; <<>> DiG 8.3 <<>> GNU.pub.ring.kikuken.org.
 4 ;; res options:  init recurs defnam dnsrch
 5 ;; got answer:
 6 ;; ->>HEADER<<- opcode:  QUERY, status:  NXDOMAIN, id:  2
 7 ;; flags:  qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
 8 ;; QUERY SECTION:
 9 ;; GNU.pub.server.ring, type = A, class = IN
10
11 ;; ANSWER SECTION:
12 GNU.pub.ring.kikuken.org.  1M IN CNAME mirror.example.com.
13
14 ;; AUTHORITY SECTION:
15 example.com.  2h52m58s IN SOA dns1.icann.org.  hostmaster.icann.org.  (
16                                2002121700 ; serial
17                                2H ; refresh
18                                1H ; retry
19                                2W ; expiry
20                                6H ) ; minimum
21
22 ;; Total query time:  7 msec
23 ;; FROM: kiku003.kikuken.info.kochi-tech.ac.jp to SERVER: default --
   172.21.43.59
24 ;; WHEN: Mon Feb 3 11:57:03 2003
25 ;; MSG SIZE sent:  37 rcvd:  128
```

図 5.9 サーバ選択手法の利用

```
> dig README.Audio.GNUinfo.GNU.pub.server.ring.


;; ANSWER SECTION:
README.Audio.GNUinfo.GNU.pub.ring.kikuken.org.  1M IN CNAME
mirror.example.com.
```

図 5.10　“/pub/GNU/GNUinfo/Audio/README” の問い合わせ

```
> dig gcc-g---2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org


;; ANSWER SECTION:
gcc-g---2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org 1M IN CNAME
mirror.example.com.
```

図 5.11　“/pub/GNU/gcc/gcc-2.95/gcc-g++-2.95.tar.gz” の問い合わせ

5.12.2　対象サーバが重複したファイルを保持する場合

同一のディレクトリツリーを保持する 2 つのサーバを対象として実験を行なう。

DNS サーバとアナウンスするサーバ名を “ring.kikuken.org” とし、選択対象のサーバを “mirror.example.com” と “ring.example.net” とした。

設定ファイルを変更し、“rdndns” を実行した。本実験で用いた設定ファイルを図 5.12 に示す。

サーバ選択手法の利用

サーバ群が/pub/GNU/以下に保持している任意のファイルにおいて、RDNDN を用いて、問い合わせを行なう。

```
name    =  ring.kikuken.org
server  =  mirror.example.com
lslr    =  /ls-lR
server  =  ring.example.net
lslr    =  /ls-lR
```

図 5.12　実験に用いた設定ファイル “rdndns.conf”

ここでは 5.12.1 で用いたファイルに対して同じ問い合わせを 2 回を行なった。これは、DNS サーバのラウンドロビン機能により、正規名が循環することを確かめるためである。以降で、それぞれの問い合わせ結果を示す。

- “/pub/GNU/GNUinfo/Audio/README”

  1 回目の問い合わせの結果を図 5.13 に示す。1 回目の問い合わせの結果を図 5.14 に

```
> dig README.Audio.GNUinfo.GNU.pub.server.ring.


;; ANSWER SECTION:
README.Audio.GNUinfo.GNU.pub.ring.kikuken.org.  1M IN CNAME
ring.example.net.
```

図 5.13　“/pub/GNU/GNUinfo/Audio/README” の問い合わせ

  示す。

- “/pub/GNU/gcc/gcc-2.95/gcc-g++-2.95.tar.gz”

  ここでは、RDN に変換の際に、“+” を”−”に変換しなければならない。したがって、RDNDN は、“gcc-g—2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org” となる。

  1 回目の問い合わせの結果を図 5.15 に示す。

  2 回目の問い合わせの結果を図 5.16 に示す。

```
> dig README.Audio.GNUinfo.GNU.pub.server.ring.


;; ANSWER SECTION:
README.Audio.GNUinfo.GNU.pub.ring.kikuken.org.  1M IN CNAME
ring.example.net.
```

図 5.14　“/pub/GNU/GNUinfo/Audio/README” の 2 回目の問い合わせ

```
> dig gcc-g---2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org


;; ANSWER SECTION:
gcc-g---2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org 1M IN CNAME
mirror.example.com.
```

図 5.15　“/pub/GNU/gcc/gcc-2.95/gcc-g++-2.95.tar.gz” の 1 回目の問い合わせ

```
> dig gcc-g---2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org


;; ANSWER SECTION:
gcc-g---2.95.tar.gz.gcc-2.95.gcc.GNU.pub.ring.kikuken.org 1M IN CNAME
ring.example.net.
```

図 5.16　“/pub/GNU/gcc/gcc-2.95/gcc-g++-2.95.tar.gz” の 2 回目の問い合わせ

5.12.3　評価

第 5.12.1 節と第 5.12.2 節の結果より、プロトタイプシステムは想定した通りの動作をした。

しかし、このプロトタイプシステムは、“rdnds” コマンドの実行にかなりの時間を費す。第 5.12.1 節の実行時間を Perl の “times 関数” を使って測定したところ、187.688 秒であった。約 30 分もかかっている。もちろん、実験環境にもよるのだがそれでも時間がかかりすぎる。

以降では、プロトタイプシステムではなく、提案手法において想定される場合の評価を行なう。

第 6 章

提案手法における評価

本章では、提案したサーバ選択手法についての評価を行なう。

6.1 評価内容

提案する手法について、サーバ選択の対象とするミラーサーバのレベルと、ミラーサーバのミラー対象において、それぞれの場合を想定し評価を行なう。

- 選択対象をミラーサーバに限定
- 選択対象を部分ミラーサーバと完全ミラーサーバと両方
- 選択対象を完全ミラーサーバに限定

以降でそれぞれの場合において詳細を述べる。

6.2 選択対象を部分ミラーサーバに限定

サーバ選択の対象するミラーサーバ群を、すべて部分ミラーで構成したと想定する。

この場合には、各サーバの保持するディレクトリツリーにより、次の場合が想定できる。

- 保持するファイルが重複していない
- 保持するファイルが重複している

以降でそれぞれの場合において詳細に述べる。

6.2.1　保持するファイルが重複していない

各ミラーサーバの保持するファイルが重複していない場合、つまり、RDN がすべて重複しない場合を想定する。

サーバ群の動作

図 6.1 に、各ミラーサーバの保持するファイルが重複していない場合のサーバ群の動作を示す。

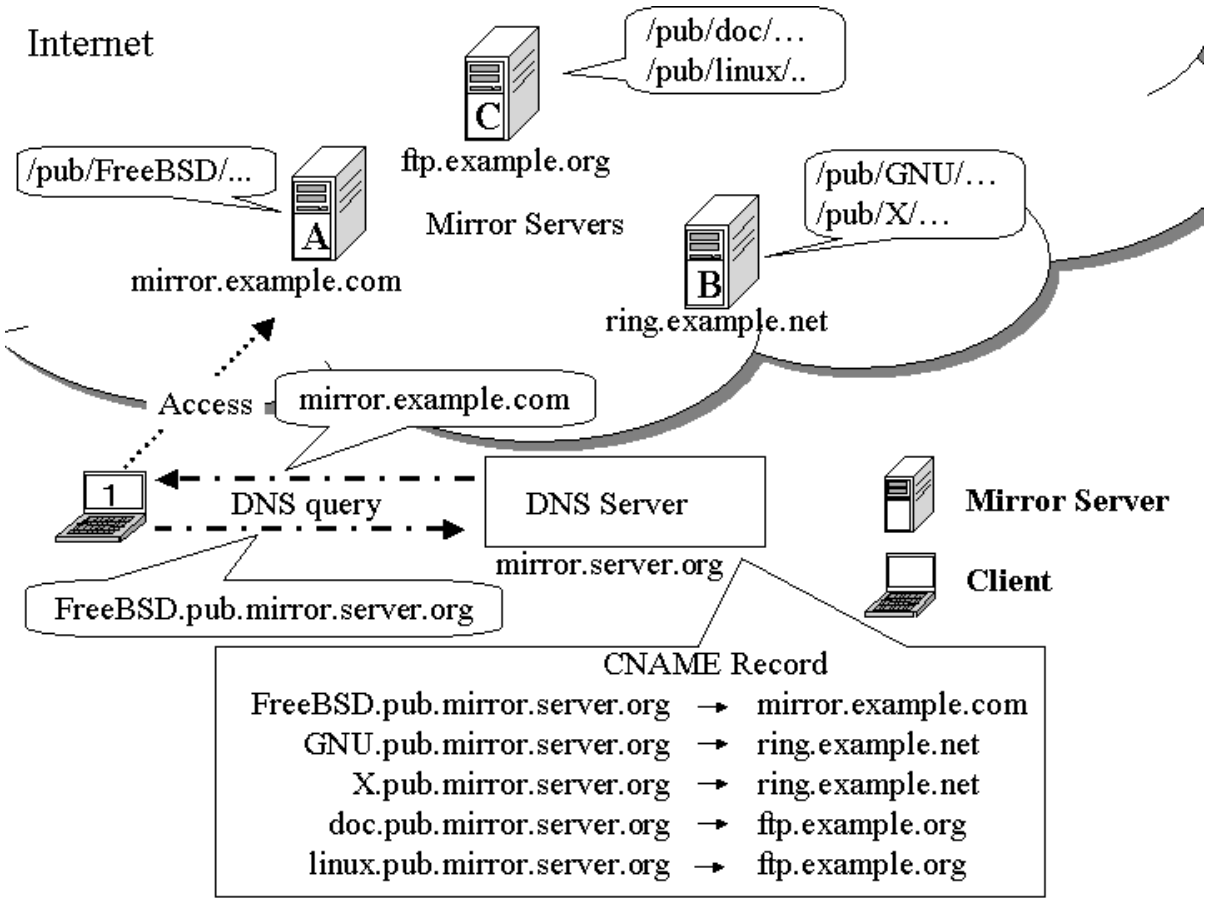

図 6.1　保持するファイルが重複していない場合

ユーザがクライアント 1 を使い、“/pub/FreeBSD” 以下のファイルを取得しようと試みた場合、“/pub/FreeBSD” 以下を保持するミラーサーバは、“mirror.example.com.” しか存

在しない。そのため、ユーザが “FreeBSD.pub.mirror.server.org.” にアクセス要求を出すと、DNS サーバは、“mirror.example.com.” を返し、クライアント 1 は “/pub/FreeBSD” 以下を保持する “mirror.example.com.” にアクセスを行なう。

特徴

ミラーサーバ群の保持するファイルが重複しない場合、サーバ選択は行なえるが、アクセスしたサーバが保持するファイルが最新のものである保証は出来ない。これは、他に比較できるミラーサーバが存在しないため、ファイルの最新性は個々のサーバの同期にかかっている。

また、ファイル取得の確実性は、あるサーバがダウンしたりトラブルがあった場合には、そのサーバが保持するファイルの取得は不可能である。

この構成は、部分ミラーサーバ群のミラーによるファイルの最新性を信用している場合や、ある程度の同期の遅延を許可する場合に有効である。

また、部分ミラーサーバ群よりも、むしろマスターサーバ群である場合に有効である。

6.2.2　保持するファイルが重複している

各ミラーサーバの保持するファイルが重複している場合、つまり、RDN が重複する場合を想定する。

サーバ群の動作

図 6.2 に各ミラーサーバの保持するファイルが重複している場合のサーバ群の動作を示す。

ユーザがクライアント 1 を使い、“/pub/linux” 以下のファイルを取得しようと試みた場合、“/pub/linux” 以下を保持するミラーサーバは、“mirror.example.com.” と “ftp.example.org.” が存在する。この場合、ユーザが “linux.pub.mirror.server.org.” に

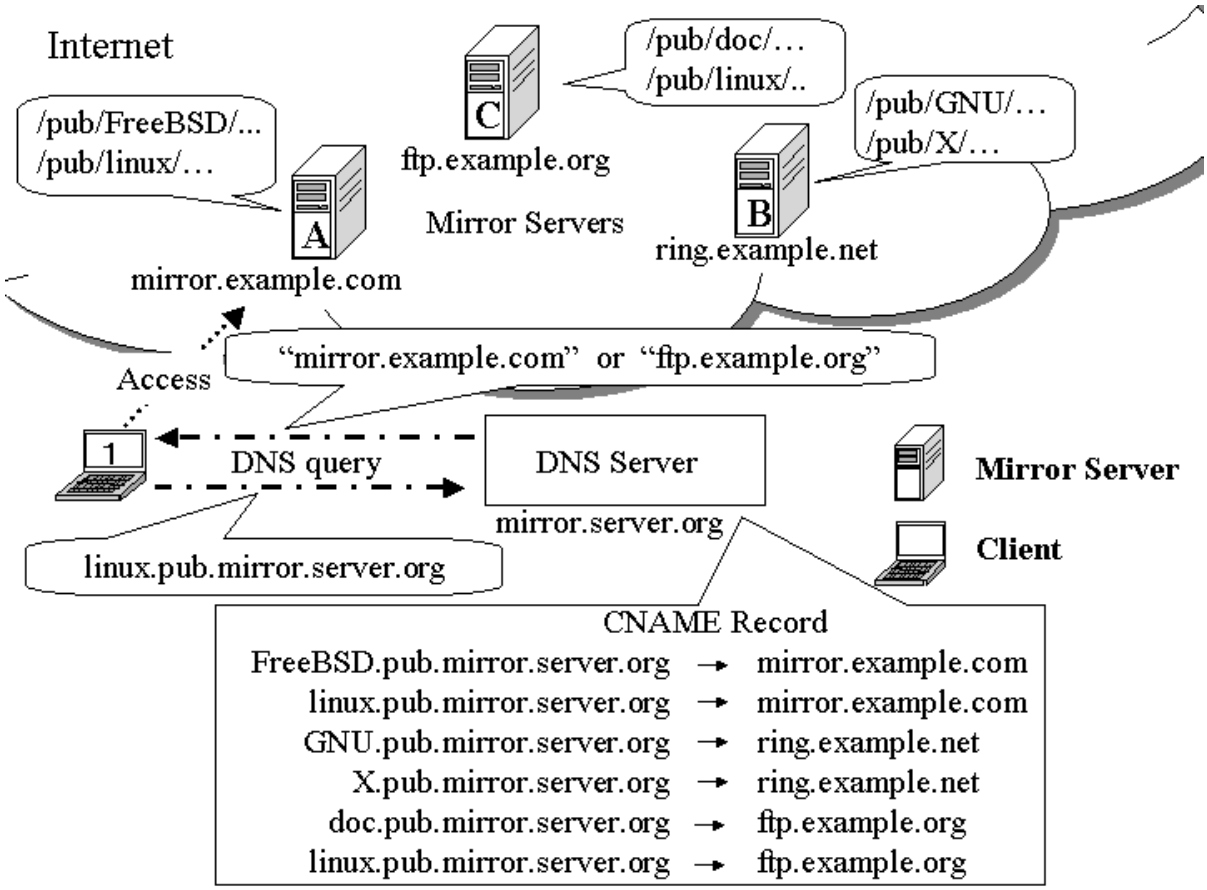

図 6.2　保持するファイルが重複している場合

アクセス要求を出すと、DNS サーバは、“mirror.example.com” と “ftp.example.org.” を循環させ返答する。したがって、クライアント 1 は名前解決のたびに違うアドレスを受け取り、“/pub/linux” を保持する “mirror.example.com” か “ftp.example.org.” にアクセスを行なう。

特徴

ミラーサーバ群の保持するファイルが重複している場合、重複しているファイルにおいては、最新のファイルのあるサーバを選択する。重複しているファイルが同じものであれば、DNS サーバは、ラウンドロビンを使い、そのファイルを保持する正規名を循環して返答する。

重複してないファイルに関しては、6.2.1 と同様の動作を行なう。

この構成は、部分ミラーサーバ群の中でアクセスが集中するファイルを保持している場合に、別のサーバでファイルを保持することで、負荷分散が実現できる。

また、重複するファイルを保持するサーバのどれかが稼働していれば、ファイルの取得が可能である。

6.3　選択対象を部分ミラーサーバと完全ミラーサーバで構成

サーバ選択の対象するミラーサーバ群を、部分ミラーサーバと完全ミラーサーバで構成したと想定する。

6.3.1　サーバ群の動作

サーバ選択の対象を部分ミラーサーバと完全ミラーサーバの両方で構成した場合のサーバ群の動作を図 6.3 に示す。

この場合、DNS は、第 6.2.2 節の保持するファイルが重複している場合と同じ動作を行なう。

特徴

第 6.2.2 節の保持するファイルが重複している場合と同様に、重複しているファイルにおいては、最新のファイルを保持するサーバを選択する。

第 6.2.2 節よりも、多く重複するファイルがあるため、どれかのミラーサーバでトラブルがあっても、ファイルを取得出来ない可能性は第 6.2.2 節よりも低下する。

しかし、一般に完全ミラーサーバにおいて同期の遅延が部分サーバよりも起こりやすいことが言える。

この構成は、異なるマスターサーバを完全ミラーしているサーバ群においても同様の構成となる。重複しているファイルもあれば、重複していないものも存在するからである。

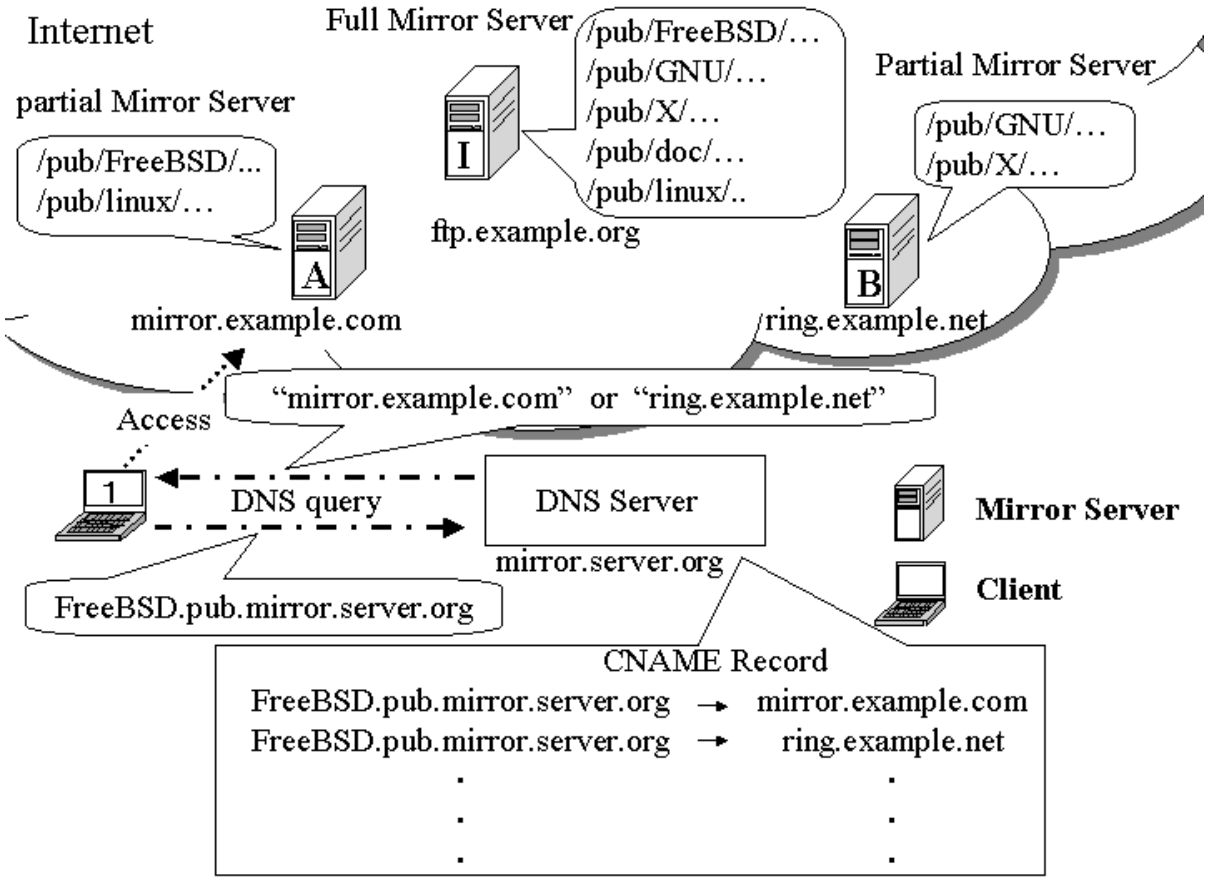

図 6.3　選択対象を部分ミラーサーバと完全ミラーサーバで構成

6.4　選択対象を完全ミラーサーバに限定

サーバ選択の対象するミラーサーバ群を、完全ミラーサーバに限定して構成したと想定する。

6.4.1　サーバ群の動作

完全ミラーサーバのみで構築した場合のサーバ群の動作を図 6.4 に示す。

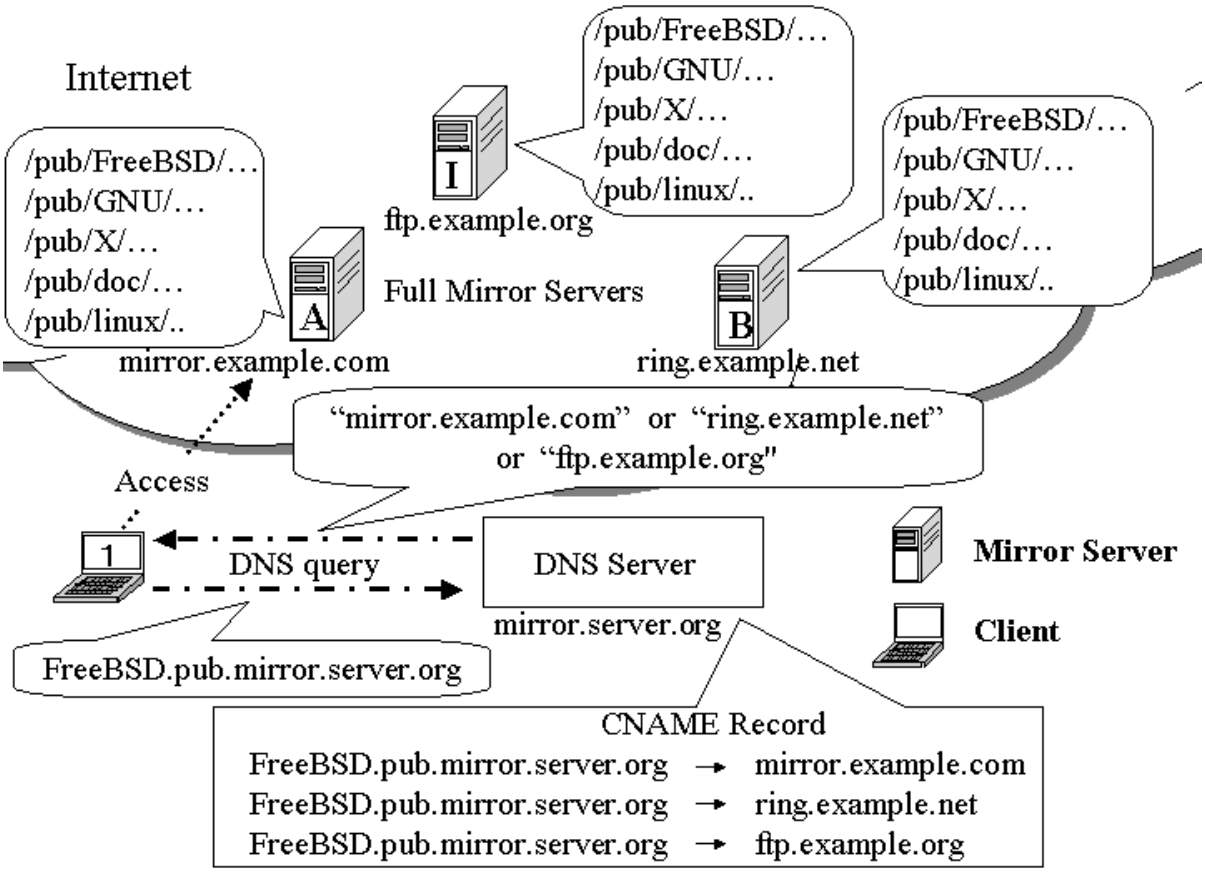

図 6.4　選択対象を完全ミラーサーバに限定して構成

特徴

すべてのサーバが任意のファイルに対して選択対象となり、対象サーバ内でのファイルの最新性が保証される。それぞれのミラーサーバの同期に遅延にが生じても、任意のファイルを保持していないサーバにはアクセスが行なわれないため、この構成はファイルの最新性を保証する。

第 7 章

提案手法利用者支援システム

提案手法をユーザが利用するには、URL から RDNDN を生成する必要がある。しかし、RDNDN を生成する作業は手間がかかり、ユーザに本来なら必要としない負荷を強いることになる。そこで本章では、WWW ブラウザを用いて、通常の URL 入力を行なうだけで、RDNDN を利用できるシステムを提案する。まず、本システムの概要を述べ、次に動作の流れを述べる。そして、実装について述べる。

7.1 概要

本システムは、WWW サーバがリクエストを受けた URL から RDNDN を生成する。そして、WWW ブラウザのリフレッシュ機能の転送先として、ホスト名を RDNDN に置き換えた URL を用いるものである。本システムを用いることで、ユーザが負担を強いることなく RDNDN を利用可能である。

7.2 動作の流れ

本システムの動作の流れを次に示す。

- URL から RDNDN を生成
- URL のホスト名を RDNDN に置換
- リフレッシュ機能の転送先として URL を指定

以降で各動作について詳細を説明する。

7.2.1　URL から RDNDN を生成

クライアントからリクエストされた URL から、ホスト名とファイルやディレクトリへのパスを抽出し、RDNDN を生成する。

7.2.2　URL のホスト名を RDNDN に置換

第 7.2.1 節で生成した RDNDN をリクエストされた URL のホスト名と置き換える。

図 7.1 は、受け取った URL“http://mirror.example.com/pub/FreeBSD” から RDNDN を生成し、URL のホスト名を RDNDN に置き換えた例である。

```
http://mirror.example.com/pub/FreeBSD

                ↓

http://FreeBSD.pub.example.com/pub/FreeBSD
```

図 7.1　URL のホスト名を RDNDN に置換

7.2.3　リフレッシュ機能の転送先として URL を指定

第 7.2.2 節でホスト名を RDNDN に置き換えた URL を WWW ブラウザのリフレッシュ機能の転送先に指定する。そして、クライアントへの応答として、リフレッシュ機能を付加した HTML 文章を返す。

WWW ブラウザのリフレッシュ機能は、HTML の Meta タグを使うことで簡単に利用できる。Meta タグは、HTML の Head 部分に記述するタグで、その文書に関する情報である「メタデータ」を指定するためのタグである。

リフレッシュ機能を利用するには、Meta タグの http-equiv 属性に “refresh”、content 属性に “URL” を記述する。

図 7.2 は、ホスト名を RDNDN に置き換えた URL を用いて、Meta タグとして記述した例を示している。

```
<meta http-equiv=‘‘refresh’’ content=‘‘http://FreeBSD.pub.example.com/pub/FreeBSD’
```

図 7.2 リフレッシュ機能の転送先に URL を指定

図 7.2 を付加した HTML 文書をクライアントへの応答とする。クライアントの WWW ブラウザは受け取った HTML 文書を解析し、リフレッシュ機能を実行し、RDNDN に置換した URL を用いてリクエストを送り直す。

よって、ユーザは WWW ブラウザに通常の URL を入力することで、RDNDN を利用することが可能である。

7.3 実装

本システムは、WWW サーバに apache を用いて実装を行なった。また、第 7.2 節で説明した動作の処理は CGI を用いた。CGI を用いた理由として、リフレッシュ機能が実装してある WWW ブラウザならどんなブラウザでも実行可能なシステムの構築を目指したからである。

7.3.1 apache の設定

ユーザがリクエストした URL をから RDNDN を生成するために、ファイルやディレクトリへの各パスごとに、リフレッシュ機能を付加した HTML 文章を用意するのは、大変な手間である。そこで本実装では、HTTP のエラー時に表示するページとして、第 7.2 節で説明した処理を行なう CGI“url2rdndn-path.cgi” を実行することで実現した。

具体的には、ファイルやディレクトリへのパスが見つからないときに表示する “404 NotFound” の Error ページの代わりに、“url2rdndn-path.cgi” を実行するのである。

apache の設定ファイル、httpd.conf の Customizable error response の ErrorDocument を図 7.3 のように “url2rdndn-path.cgi” へのパスに変更する。

```
# Customizable error response (Apache style)
ErrorDocument 404 /missing.html
                    ↓
# Customizable error response (Apache style)
ErrorDocument 404 /url2rdndn-path.cgi
```

図 7.3 apache の設定ファイルの変更

以上より、WWW サーバがコンテンツを持っていない時に、“url2rdndn-path.cgi” を実行するシステムが実現できた。なお、ユーザには URL で指定するホスト名として本システムが動作するサーバ名をアナウンスしておく必要がある。また、ユーザが URL を入力ミスしたときでも、“url2rdndn-path.cgi” が実行を行なうが、RDNDN の名前解決が出来ないため、DNS エラーとして WWW ブラウザが処理を行なう。よって、“url2rdndn-path.cgi” が再帰的に動作することはない。

本システムを用いることで、ユーザは WWW ブラウザを用いて通常の URL 入力を行なうだけで、提案するサーバ選択手法を透過的に利用することが出来る。

第 8 章

考察

本研究で提案したサーバ選択手法には、いくつかの問題点がある。本章では、提案した手法における問題点について考察する。

8.1 利用時におけるディレクトリ情報の収集

ユーザはある程度のディレクトリ情報を収集しなくては、サーバ選択を行なうことができないという問題がある。本手法では、ドメインネームにディレクトリ情報を付加することで、任意のファイルの最新性を保証するサーバを選択している。このため、ユーザがある程度のディレクトリ情報を収集していない場合、アナウンスしているサーバ名だけではサーバ選択が出来ない。

これは本手法にとってユーザに強いる最低限の負担であると考える。サーバ選択を行なうに当たり、ユーザは、何かを欲してサーバにアクセスを行なうわけである。その過程において、ユーザはある程度のディレクトリ情報を取得すると考える。この情報は、マスターサーバにアクセスを行なう場合にも必要な情報であるため、ユーザの負担が増えるわけではない。

解決策としては、部分ミラーのサーバ群の中にマスターサーバを完全にミラーするサーバを 1 つ以上用意しておくことである。これにより、アナウンスしているサーバ名だけで、ユーザからアクセスがあった場合に、完全ミラーサーバにアクセスさせることで、ディレクトリ情報を収集していなくても、任意のファイルを取得できる。

8.2　ドメインネーム生成規則による一意性の消失

DNS のドメインネーム生成規則により、英字の大文字と小文字とを区別できない。また、英数字以外の記号は “-”と、ドメインの区切りとしての “.”以外使用できない。よって、本手法においてはディレクトリノードの “–” と 1 つの “.” 以外の記号について、すべて “–” に変換した。しかし、これらのために変換前と変換後でのファイルの一意性を損なってしまった。

したがって、RDN と更新情報リストとの比較の際に情報コンテンツを破棄してしまう可能性が起こりえる。これは、2002 年 10 月 25 日に IESG[*1]により RFC 発行の承認をうけた「IDN(Internationalized Domain Name)[*2]」などの多言語ドメインネームを用いることで解決できると考える。多言語ドメインネームを用いることで、“–” や “.” 以外の記号の使用や、一意性を残しての変換が期待できる。

8.3　ドメインネームの文字数制限

ドメインネームに使用できる各ラベルの文字数は 63 文字である。また、ドメインネーム全体で使用できる文字数の上限は、255 文字である。したがって、ディレクトリの各ノードに使用できる文字数と RDNDN に使用できる文字数の上限がそれぞれ、63 文字と 255 文字ということになる。

つまり、文字数の上限よりも長いノードや、より深いディレクトリ構造をもつファイルには対応できないといった問題点がある。この問題は、ディレクトリノードの一意性を維持しての省略を行なうことで解決できる。しかし、ユーザが更にディレクトリ情報を収集する必要があるといった問題が残るため、現在の RDNDN をディレクトリノードと、キーワードといった形をとることで、解決できると考える。ファイルの一意性が保証されている場合に、ファイル名とアナウンスしているサーバ名だけでのアクセスが可能である。ファイル名

[*1] http://www.ietf.org/iesg.html

[*2] http://www.ietf.org/html.charters/idn-charter.html

に一意性が保証されない場合には、ディレクトリノードの一部をキーとして用いることで一意性を保証できると考える。

8.4　複数の別名の禁止

CNAME は、「ドメイン名と他のデータを同時に持てない」という規則がある。このため、CNAME の複数の利用はネームサーバにより、違反と見なされる。

今回、実装に用いた BIND9 では、複数の CNAME の利用は違反と見なすため、ラウンドロビンで正規名を循環するには、別途設定が必要である。しかし、今後この方法がソフトウェアとしてサポートされていくとは限らないため、CNAME レコードを用いずに、IP で指定する A レコードを用いることで解決する。

しかし、第 4.2.5 節で述べたように、1 つのサービスを提供しているサーバにおいては、ドメインネームの変更よりも IP アドレスの変更の方が多いため、DNS の管理者によるモニタリングが必要である。

8.5　シンボリックリンク

本手法における実装では、ファイルがシンボリックリンクである場合に対して、対応を行なっていない。これは、シンボリックリンクにより、特定のファイルまでのディレクトリ構成が複雑化しているため、今回の実装では対応できない。ミラーサーバのディレクトリ情報を ls-lR ファイルではなく、ディレクトリノードを自由に検索可能なデータベースにすることで対応可能である。

第 9 章

まとめ

最後に本章では、本研究のまとめと今後の課題について述べる。

9.1 まとめ

インターネットでは、各種サーバの負荷分散を目的としてミラーサーバが用いられている。しかし、ユーザがミラーサーバを利用するには、ミラーサーバに関する多くの情報を収集する必要があるという問題がある。そこで、サーバ選択手法が満たすことが望ましい性質を挙げ、既存のサーバ選択手法において評価を行なった。

そして、ユーザの利便性の向上を最優先に考え、DNS を用いた部分ミラーのサーバ選択手法を提案し、実装を行なった。

本手法は、ユーザが 1 つのサーバ名と、ある程度のディレクトリ情報を組み合わせたドメインネームを用いることで、任意の情報コンテンツを保持しているサーバにアクセスできるというものである。この場合、サーバ選択の対象となるサーバ群は、ディレクトリ情報の同一性が保証されないサーバや部分ミラーサーバであっても構わない。

また、本手法を実装したプロトタイプシステムにおいては、サーバ選択の対象となる個々のサーバから、毎回すべてのディレクトリ情報を取得する。そして、すべてのミラー対象ファイルにおいて最新性の検証を行なっているため、処理に時間的コストがかかるという問題がある。

さらに、WWW ブラウザを用いることで本手法をユーザが透過的に利用出来るシステムを提案し、実装を行なった。

9.2　今後の課題

今後の課題を次に挙げる。

9.2.1　CNAME レコードの更新頻度の最適性

ファイルの最新性やファイル取得の確実性を保証するために、どの程度の頻度で対象サーバのディレクトリ情報の収集を行ない、DNS サーバのレコードを更新すべきなのか、確かめられていない。頻度が多いとネットワークトラフィックが増加してしまい、頻度が少ないとファイルの最新性やファイル取得の確実性が保証できくなる。

そこで DNS サーバのレコード更新頻度の最適性を検証する必要がある。

9.2.2　性能の改善

本研究では、提案した選択手法の最低限の機能しか実装していない。よって、多くの性能改善を行なう必要がある。

以下に改善しなければならない重要課題を挙げる。

ディレクトリ情報の取得と管理の効率化

本実装において、ディレクトリ情報は “ls-lR” ファイルを用いて行なった。しかし、本システムの実行のたびに、各ミラーサーバから ls-lR ファイルを取得し、各ファイルの最新性の検証を行なうのは非効率である。

そこで、ディレクトリ情報の管理は、専用のデータベースを用いてディレクトリツリーとそれぞれのファイルの情報を管理し、効率化をはかる必要がある。

DNS レコードの差分更新

本実装においては、一度前回の DNS のレコードの削除を行なってから、新たにレコードを追加してきた。しかし、多くの場合、ミラーサーバのディレクトリツリーの大幅な変更が頻繁に起こることは考えにくい。そこで、DNS レコードの更新では、前回のディレクトリツリーの差分を用いて更新を行なう必要がある。

謝辞

本研究を行なうにあたり、Ring Server プロジェクトの方々には、鈴木 秀幹さんをはじめ、プロジェクトに関わる多くの方にミラーサーバの構築から運営などに関するアドバイス等をいただきました。

九州大学の下川 俊彦さんには、DNS サーバ選択機構の “TENBIN” について御指導、ご協力をいただきました。

株式会社インテック・ネットコアの中川 郁夫さんには、経路情報サーバである “RADIX” について御指導、御協力をいただきました。

また、田渕 理恵さん、西内 一馬さん、舟橋 釈仁さん、豊島 修平さんをはじめ、本研究室の方々には本研究へのアドバイスやご協力、激励をいただきました。

ほか本研究に関わったすべての人に感謝の意を表します。

最後に指導教員の菊池豊助教授には、本研究活動のすべてにおいて御指導、御協力いただきました。ここに感謝の意を表します。

また、本研究は、高知県情報生活維新の平成 14 年度ワーキンググループ重点プロジェクトとして助成を受けています。

参考文献

[AL02] Paul Alibtz and Cricket Liu. DNS & BIND 第 4 版. オライリー・ジャパン, February 2002. 高田 広章, 小島 育夫 監訳. 小舘 光正 訳.

[NH00] Alexei Novikov and Martin Hamilton. Ftp mirror tracker: A few steps towards urn, September 2000.

[PR85] J. Postel and J. Reynolds. File transfer protocol (ftp), October 1985. RFC959.

[RRDS92] R. Rivest and Inc. RSA Data Security. The md5 message-digest algorithm, April 1992. RFC 1321.

[VTRB97] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE), April 1997. RFC2136.

[YCE+97] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D.Culler. Using smart clients to build scalable servers, 1997. USENIX 1997, Annual Technical Conference.

[下川 01] 下川俊彦, 吉田紀彦, 牛島和夫. 多様な選択ポリシーを利用可能なサーバ選択機構. 電子情報通信学会論文誌, No. 9 in D-I Vol.J84-D-I, pp. 1396–1403, September 2001.