第 14 部

Internet における情報検索機構

第 1 章

WIND WG における情報検索機構の研究

1.1 1995 年度の活動概要

WIDE WIND ワーキンググループ (WIND WG) は、WWW や anonymous ftp、NetNews などのように、ネットワーク上に分散して存在する各種の公開されている情報を対象として、効率よく情報検索を行う機構に関する研究を目的としている。

WIND WG は 1995 年度より活動を開始し、初年度は効率良い情報検索機構のモデムの検討と、情報の在処についての情報を収集する部分についての研究を行った。

具体的な研究内容は以下の通りである。

- 情報の在処やその種類などを示す情報を「情報 index」として定義し、それを収集し、整理し、検索する機構のモデルを検討した。
- 「情報 index」を収集する機構を検討し、プロトタイプの実装を行った。

第 2 章

WIND のモデルとプロトタイプの設計について

2.1 はじめに

現在インターネットでは WWW[121], Gopher[122], WAIS[123], NetNews, anonymous ftp など各種の「情報提供システム」を用いて、様々な情報が広く公開され利用されている。

その情報には有用なものが多く含まれるが、現状では必要な情報を入手するために利用すべき情報提供システムとアクセス先を知る事は容易にはできない事が多い。

この問題を解決するために、archie[124] や WWW Robot, Virtual Library のように、特定のシステムに対する情報のありかのデータを収集し、検索できるようにする機構がいくつか考案され、実際に利用されている。限られた条件内ではこれらの機構は有効に利用できる場合も多いが、検索方法が十分とは言えない。また、異なった情報提供システムの情報を検索する事はできない。

本報告では、各種の情報提供システムが保持している情報について、そのありかと内容を検索するための WIND (Wide Information Navigation Directory) システムのモデルとプロトタイプについて提案する。

2.2 WIND の目的

WIND は、各種の情報提供システムによって公開されている種々の情報を効率良く検索するためのシステムを目指す。

当面は、以下に示す、現在広く利用されている各種の情報提供システムに存在するあらゆる情報を効率良く検索するシステムの研究を行うことを目的とする。

anonymous ftp, WWW, Gopher, WAIS, NetNews, X.500, finger, telnet

これらのシステムが保持する情報の在処、情報の種類などを示すものを「情報 index」とし、それを使って必要な情報を得る事ができるようにする。

2.3　WIND のモデル

2.3.1　情報の検索

人が情報を探す時には、次のようなパターンが考えられる。

1. 探したい内容が比較的はっきりしており、その在処もほぼ分かっている。

   ⇒ 目的の場所に行き情報を得る

2. 探したい内容は比較的はっきりしているが、その在処は不明である。

   ⇒ 在処を知っている人に聞く

3. 探したい内容がそれほどはっきりしている訳ではない。

   ⇒ 助言をしてくれる人に相談をする

4. 特に必要に迫られてはいないが、ランダムに各種の情報を見たい。

   ⇒ その時々に触れる情報を見る

WIND では、このうち 1, 2, 3 の形の情報検索を支援するものとする。

2.3.2　全体の構成

WIND システムでは、システムを機能面から大きく以下の 3 つの要素に分ける事にする。

1. 「情報 index」の収集機構
   — 各種の情報提供システム上に存在する情報の情報 index をもれなく収集する。

2. 「情報 index」の整理機構
   — 収集した情報 index を検索しやすい形に整理する。

3. 「情報 index」の検索機構
   — 利用者から与えられた検索キーワード等により、情報 index を検索する。

WIND システムは図 2.1 のように、3 つの機構を備えたある管理範囲を担当するサブシステムが分散配置され、互いに協力する事で動作する。サブシステム同士の関係は、対等な場合もあれば、階層的な場合もある。

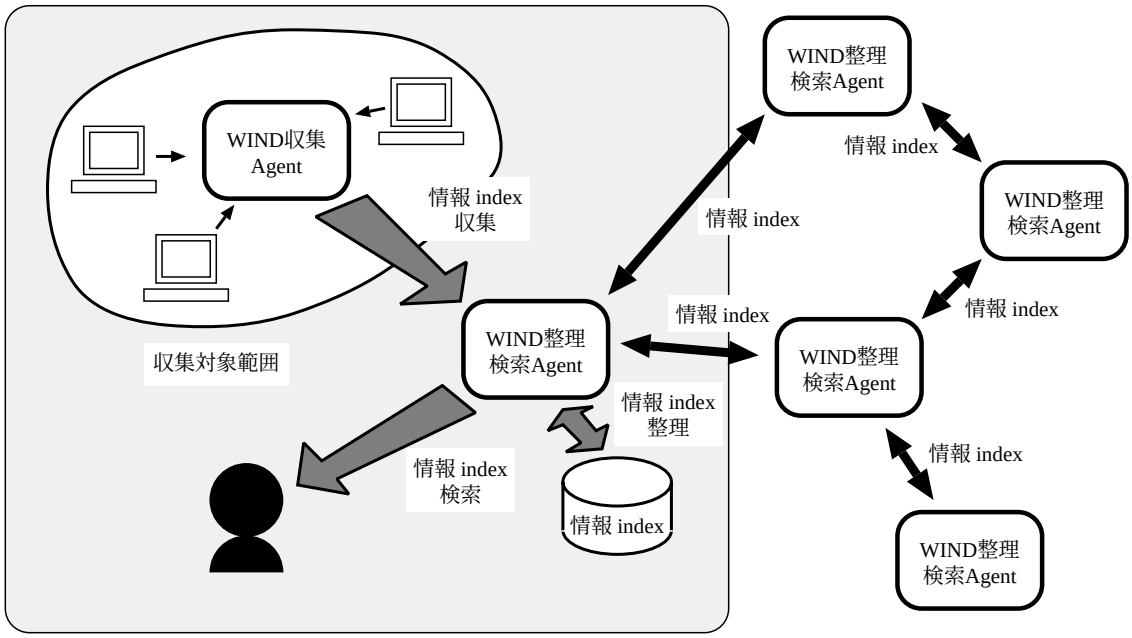

図 2.1: WIND のモデル

2.3.3 情報 index

情報 index は、情報提供システムが保持する情報の在処やその内容を表すものであり、情報 index を整理し検索するのに必要な項目を備えていなければならない。

情報 index が 備えているべき項目は以下のものとなる。

- 情報の在処:
  - ホスト名、IP アドレス、パス名、ニュースグループ名など
- 情報提供システムの種類:
  - プロトコル名、サービス名など
- 情報の属性:
  - 最終更新日付、ファイル容量、版、変更履歴、アクセス制御情報、著作権情報、複製情報など
- 情報の内容:
  - 情報の種類、分野、キーワード、要約など

2.3.4　情報 index 収集機構

情報 index 収集機構は、次のような機能を備えているものとなる。

1. 複数のサブシステムが協調しながら、世界中の情報提供システムが保持している膨大な情報を効率良く調べ、情報 index を生成する仕組み。
2. 情報の更新を検知して、必要に応じて情報 index を更新する仕組み。
3. WWW を代表とする情報提供システムは、自由に運用開始/設定変更ができるため、その事を検知して洩れや不整合を生じさせないようにする仕組み。
4. 情報 index を収集機構側から集める仕組みおよび、各情報提供システム側から情報 index を提供してもらう仕組み。

1 は、情報提供システムが保持する情報から、情報 index を生成するのに必要な情報を取り出す必要があるが、これは情報の種類に応じて適切な方法を取ることができなければならない。また、その対象として多様な形で保存されているテキスト、画像、音声、各種データなどを取り扱う事ができる必要がある。

2 や 3 を実現するには、情報 index 収集機構側が監視するよりも、情報提供システム側から更新された事を通知して、情報 index を更新する方がよいと思われる。

2.3.5　情報 index 整理機構

情報 index 整理機構は、次のような機能を備えているものとなる。

1. 検索しやすい形で整理し、必要に応じてサブシステム間で情報 index を交換する仕組み。
2. 各整理機構間の関係を定義して、それぞれの Agent が管理している情報 index についての情報を、必要に応じて報告する仕組み。
3. あいまいな検索をできるように、柔軟な形で情報 index を保持するようにする仕組み。

2.3.6　情報 index 検索機構

情報 index 検索機構は次のような機能を備えているものとなる。

1. 情報 index を検索する際に、できる限り効率良く検索が行えるような分散型の検索の仕組み。
2. anonymous ftp や WWW のミラーのように複製が作成されている場合、ネットワークトラフィックを軽減するように適切な情報の在処を提示する仕組み。

3. 情報に「版」がある場合、最新または必要とされる「版」の情報の在処を提示する仕組み。

4. 必要としている情報があいまいな場合に、対話的に情報の内容を確認しながら絞り込んでいく仕組み。

1 は、情報の在処が分かっている場合と、在処は分からないが特定するためのキーワードがある場合の検索、およびあいまい検索ができる必要がある。

2 を実現するには、ネットワークのトラフィックやホップ数、応答時間などの情報を収集し、それを元に「最適度」を計るようにする必要がある。

さらに、3 を実現するためには、ネットワークの最適度、複製の有無、版の情報を元にして、適切な情報の存在場所を検索する事が必要となる。

2.4 プロトタイプの設計

WIND システムのモデルを検証するため、プロトタイプシステムを作成する。プロトタイプを作成するにあたって、ネットワークに存在する情報提供システムからもれなく情報 index を収集するためと、負荷の分散を計るため、WIND システムの各 Agent およびその間の情報 index の転送方法に OSI ディレクトリサービス (X.500)[125] を利用する。

このプロトタイプでは、情報 index の検索の際に「あいまい検索」をする事は考慮しない。

プロトタイプシステムの構成を図 2.2 のようにする。

2.4.1 情報 index

情報 index は、OSI ディレクトリサービスの「エントリ」として保持する事にする。

情報 index 用の「エントリ」の例を図 2.3 に示す。

2.4.2 情報 index 収集機構

情報 index 収集機構は OSI ディレクトリサービスの DSA の管理範囲を収集範囲とし、各情報提供システムから収集した必要な情報を元に、情報 index を作成して DSA 渡す。DSA は一旦そのままの形で自分が管理するエントリとして保存する。OSI ディレクトリの DSA 毎に収集機構を用意する事で、分散して情報 index を収集する事ができる。

プロトタイプでは、情報提供システム側および OS に変更を加えないで収集機構を構築するものとする。そのため、収集機構は「ポーリング型」で情報 index を収集するようにする。

収集対象とするホストやサービスについては、あらかじめ OSI ディレクトリまたは DNS に問い合わせる事で決定するものとする。

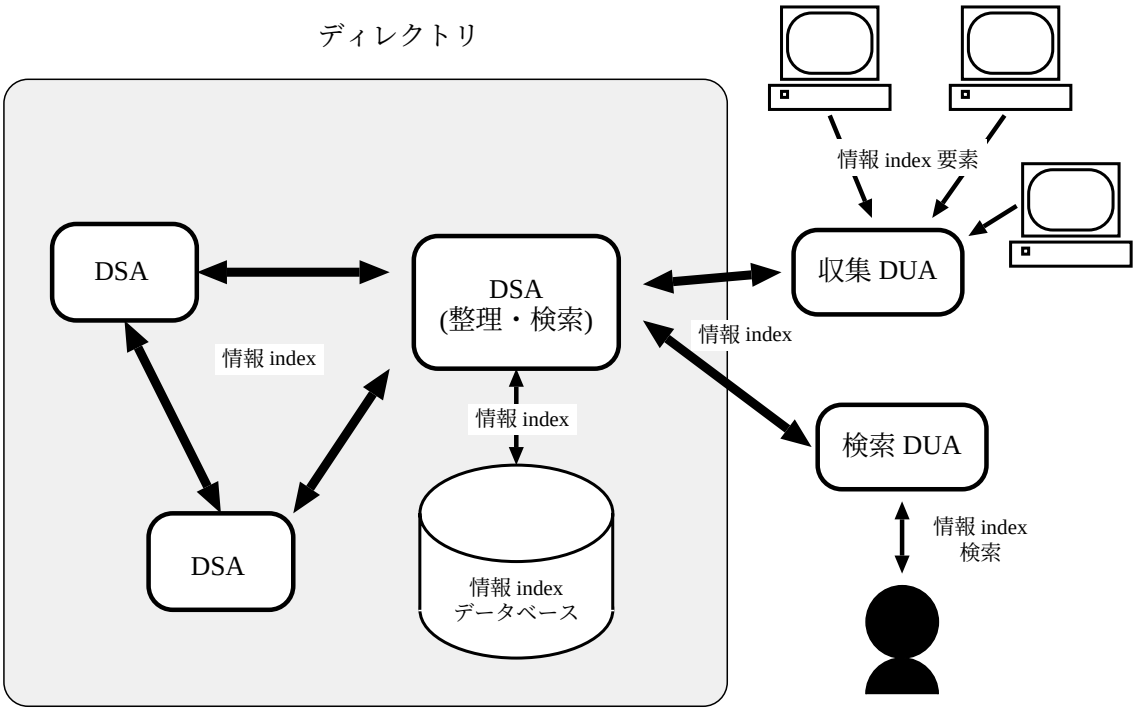

図 2.2: WIND のプロトタイプシステム

```
objectClass= top & indexObject & software
host= ftp.iis.u-tokyo.ac.jp
ipAddress= 157.82.96.67
protocol= ftp
place= /mail/CF/CF-3.4W2.tar.gz
size= 207362
date= Aug 24 23:01 1995
keyword= mail & sendmail & sendmail.cf & WIDE
```

図 2.3: プロトタイプの情報 index の例

2.4.3 情報 index 整理機構

情報 index 整理機構は、DSA が一旦保存した情報 index を取り出し、情報の属性や内容を元に整理し、検索に適した形態に配置し直す。

整理した情報 index の配置として、以下の 2 通りの方法を検討する。

1. 情報 index の属性や内容に従って、OSI ディレクトリの全体の DIT (Directory Information Tree) 上に分類して検索に備える。

2. 整理・分類した情報 index を、その DSA のエントリとして保存して、検索に備える。

1 の方法では、情報 index を分類するための DIT を別途構築することになる。この方法では、DIT を構築する際に全体のツリー構造を決定しなければならない。検索はツリー構造に即して行う事になる。

2 の方法では、DIT は通常の組織を元にした階層構造のもののままとなる。DSA 内に保存する情報 index は、多様なものとなるため、効率良く検索できる形で整理し格納する。

これ以外に、整理・分類した index を OSI ディレクトリとは完全に独立した分散データベースシステムに保存して検索できるようにする方法もありうるが、今回はとりあえず検討しない。

2.4.4 情報 index 検索機構

情報 index 検索機構は、上述の整理機構の種類によってその実現方法が異なる。整理機構が 1 の場合は、検索 DUA から与えられたキーワード等に従って、DUA が参照 (refferal) 型または連鎖 (chaining) 型で検索を行う。

2 の場合は、検索はマルチキャスト (multicast) 型で、情報 index を保持している全ての DSA を対象として行う事になる。

第 3 章

ネットワーク上での更新情報の収集について

3.1 目的

現在、インターネット上で情報提供システムによって公開されている情報のありかを見つけ出すシステムとして、archie, WAIS, WWW Search Engine などが知られているが、それらはあくまで「情報を自分から探して歩く」しくみである。この方法は現在使われている情報提供システムに何も手を加えずに実現可能であるが、多くの場合大きなネットワークトラフィックを発生させ、かつ情報提供システムのサーバの負荷となる事が多い。

本研究では、WIND システムの情報 index 収集機構を実現するため、情報提供システムのサーバ側から、情報の追加/変更/削除といった更新情報を流すことによって、情報を集める側の負担を軽減する方法を提案する。

ここでは、情報提供システムのサーバを改造する方法、サーバが記録する情報を利用する方法に加え、現在利用されているしくみに手を加えずに実現する方法も考える。現在利用されている情報提供システムのサーバがこの方式をサポートすることで、素早い情報の更新やネットワークトラフィック、サーバの負荷の軽減が実現できる。

3.2 対象情報提供システム

今回は、情報提供システムとして、anonymous ftp, NetNews, WWW を対象として情報 index 収集機構のプロトタイプの実装を行った。

3.2.1 FTP (File Transfer Protocol)

FTP はインターネットで使われるファイル転送プロトコルの一つである。本文中ではその FTP を使った Anonymous FTP という公開ファイル転送サービスを指して、単に FTP と記述することにする。

FTP は RFC 959 [126] で規定されている。ここでは、ファイル/ディレクトリ操作のための命令群が定義されており、対話的にファイル転送を行う事が可能になっている。

Anonymous FTP は特定のユーザ名でアクセスする事により、特定のディレクトリ以下のファイルを誰でも入手する事ができる仕組みであり、フリーソフトウェア等の配布に利用されている。

3.2.2 NetNews (Network News System)

NetNews は、コミュニケーションシステムであるとともに、不特定多数の人々に情報を提供するシステムである。情報を提供したい人は、その内容を「記事」として NetNews システムに対して書き込み (投稿 (post)) する。

投稿された記事は隣接 NetNews システムに対して配送され、世界中に伝達されていく。具体的には、各記事に全世界で一意な “Message ID” と呼ばれる番号を割り当て、その Message ID を持つ記事がまだ配送されていない場合に、隣接 NetNews サーバホストがその記事を受け取る。全ての NetNews サーバホストは互いに接続されているので、全ての記事が世界中に伝播する。ホスト間の通信は NNTP (Network News Transfer Protocol) [127] によって行われている。

3.2.3 WWW (World Wide Web)

WWW とは、ネットワークからアクセスできる情報を統一的に扱うことで、ネットワークを一つの情報空間とみなそうという考え方で構築されたシステムである。最終的には、分散ハイパーメディアシステムを構築することを目指している。

情報の所在を表すのに URI (Universal Resource Identifiers) [128] という記法を導入して、ネットワークからアクセスできる情報を、全てそれで記述する。URI にはプロトコル、ホスト名、(そのホスト内での) 情報の場所が記述されていて、それだけで情報を適切にアクセスすることができるようになっている。

WWW 用の情報は、HTML (Hyper-Text Markup Language) という SGML に似た構造を持った、ハイパーテキスト文書を書くための言語で書かれる。HTML で書かれた情報は、WWW 用に作られたブラウザソフトで表示することができる。

現在の WWW は、主にこの 2 つの技術と既存のアプリケーションソフトやシステムの組合せで構成されている。Mosaic, Netscape に代表される使いやすいブラウザソフトが出現したことで、分散ハイパーテキスト環境を一般の人が手軽に使えるようになった。

WWW の情報は HTTP (Hyper-Text Transfer Protocol)[129] によって転送される。

3.3 各情報提供システムの特性

情報の取り扱われ方は、それぞれの情報提供システムによってかなり違っている。また、情報がどこから情報提供システムに入力されるか、どのようにシステムから削除されるの

か、などの点にも大きな違いがある。効率的な情報の更新の検知を行うためには、この違いは無視できない。

ここでは各情報提供システムについて、保持されている情報がどのように取り扱われるかを述べる。

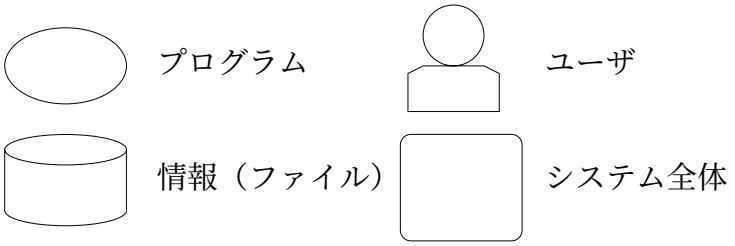

図 3.1: 凡例

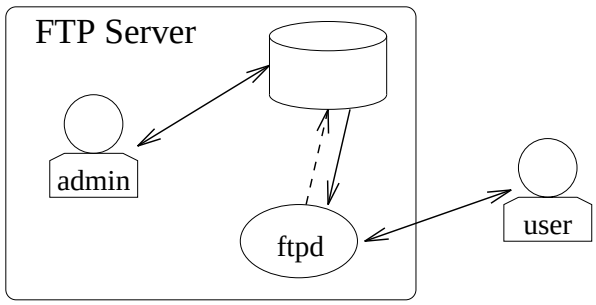

図 3.2: FTP システム

3.3.1　FTP

FTP サーバは普通 ftpd と呼ばれる。FTP ではファイルの更新は、ftpd を通してファイルをディレクトリに put するか、local マシンのユーザがファイルを直接ディレクトリにコピーすることで行なわれる。ftpd を通して更新されたファイルは ftpd によって検知可能であるが、ユーザがファイルの更新を行なった場合、ftpd はそれを察知できない。また、ファイルの削除はローカルマシンのユーザが行うのが普通である。

なお、FTP では、ローカル FTP サーバの状態を、遠隔 FTP サーバの状態と同じに保つための mirror と呼ばれる仕組みがあり、mirror を介して定期的にファイルを取得したり削除したりする事がよく行われている。この場合は、ファイルの更新は mirror によって検知する事ができる。

FTP では、mirror などにより、全く同じ情報が別の FTP サーバホストにコピーされて、そこで新たに公開される事が普通に行われており、情報の一意性が保証されていない。

3.3.2 NetNews

NetNews は情報の更新検知と言う点から見れば、非常に単純なシステムである。情報は新たなものがどんどん増える一方であり、古い情報は消える場合もあるし、消えない場合もある。NetNews に流れる大量の情報のうち、どの程度が保存すべき/検索できるべき情報であるかの調査はなされていないが、おそらくそれほど高い割合ではないのではないと思われる。

NetNews のサーバシステムは Bnews や Cnews, INN などが知られているが、いずれも「記事」を管理するプログラムやデーモンが存在しており、ここではファイル転送の立場からそれらをまとめて nntpd と表記する。NetNews の情報更新/追加は全て nntpd によって行われ、nntpd 以外が NetNews システムに情報を更新/追加することはない。また、削除はそのためのコマンドやスクリプトを使って行われ、nntpd は記事の削除には関与していない。また、情報の一意性については、Message ID によって確認する事ができる。

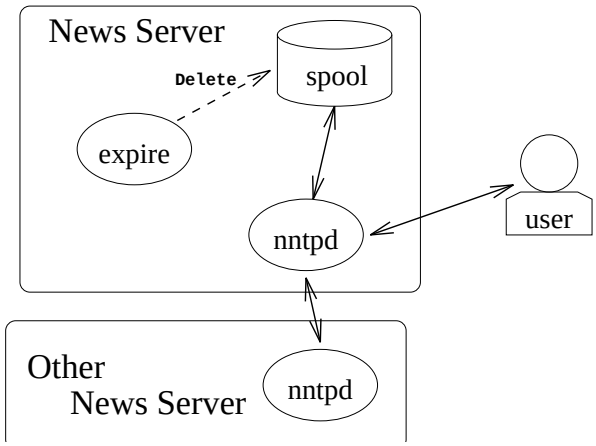

図 3.3: NetNews システム

3.3.3 WWW

WWW は幅広い運用形態が考えられるサービスである。WWW ブラウザによって NetNews や FTP の情報を見る事ができるが、HTTP で転送される WWW の情報そのものも NetNews や FTP の代わりに使われるようになってきている。

WWW の場合、NetNews 同様、更新を行うユーザが非常に多いという特徴もある。各自のホームディレクトリ以下に自分の世界を作り上げることができ、その更新は各ユーザが任意のツールを使用して行う。NetNews と違い、WWW サーバはその情報にアクセスするまではファイルがそこにあるかどうかすら気にしない。つまり、WWW サーバに手を入れるなどの方法で、前もって更新を感知することは非常に困難であるということになる。

WWW のサーバは普通 httpd と呼ばれている。httpd は、情報の更新/追加/削除には全く関わっていない。ユーザは HTML という言語で書かれたファイル (情報) をローカルホストで編集して、更新/追加/削除する。httpd でこれらの挙動を監視することは多くの場合不可能である。ただし、httpd からプログラムを呼び出す CGI (Common Gateway Interface) や Server Parsed HTML というインターフェイスを使う事で、httpd を通して情報の更新が可能ではある。

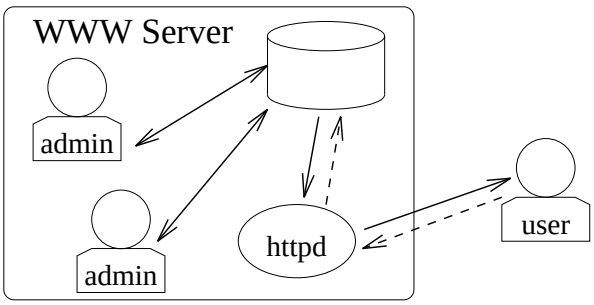

図 3.4: WWW システム

3.4　更新情報収集機構の設計

現在良く知られている更新情報の収集方法には、FTP では archie, mirror、WWW では robot がある。NetNews の場合は、NetNews システム自体が更新情報収集にかかわることはあまりなく、ユーザが必要な NetNews の記事を保存するなどの方法で情報を収集する。また最近は、各種の最新情報を流したり、保存すべき情報を流すニュースグループなども作られ、NetNews システムの指定された記事を削除する機能や、特定のニュースグループの記事を別の場所に保存する機構を使って、最新の正しい情報が NetNews サーバ内に残るような工夫もされている。

archie, mirror サービスは、FTP サーバが作成する、`ls-lR` というそのサーバが持っている全てのファイルのリストを元にして、手元に持っている古い情報と比べてみて、どのファイルが更新されているかを調べている。

WWW の robot は、手辺り次第にリンクを辿ったり、WWW サーバの名前やアクセスポートを推測して接続したりといった、ad hoc な方法でサーバにたどり着いて、そのサーバが持つ情報をかき集めてくるという手法を取っている。

これらの方法では、更新されていないファイルの情報などの無駄なデータがネットワークを流れる。ネットワーク全体でのデータ量は増える一方なので、このような無駄なトラフィックは減らさなければいけない。また、インターネットの一般化によってますます情報が増え、欲しい情報がを探すためのしくみは重要なため、そのようなものを提供するために、情報の更新検知は欠かせない機能である。

3.4.1 モデル

ここでは、次のようなモデルを提案する。

ユーザがその時点で最新の情報を探す手段を提供するために、更新情報は重要である。重要な情報を多く蓄えているサーバほど大規模であり、そのようなサーバはクライアントのリクエストも多い。このような状況では、複数のクライアントがサーバの更新情報を欲しがり、おのおのでそれを作ろうとして、似た情報が何度もアクセスされることが想定される。このことから、更新情報をサーバ側で用意しておくことが望ましい。サーバから流す情報を最小限にすれば、サーバ/ネットワーク共に負荷を軽減することができる。

この情報をどのように配るかについては 2 通り考えられる。

- クライアントからのリクエストに応じて更新情報を送る。
- 適当な時間間隔で、ファイルの更新があった時にネットワークに流す。

更新情報を取り出す部分と取り出された更新情報を管理する部分は分割する。これによってシステムによる管理方法の違いを吸収する部分を完全に分離できる。こうしておけば、新しいシステムが増えた時の対応が楽になり、更新情報を管理するサーバは別のマシンで動かせば良いのでサーバマシンの負荷を軽減できる。

この要求を満たすモデルは図 3.5 のようになる。収集 Agent が情報提供システムから更新情報を取り出し、管理 Agent がそれを管理する。

以下では、この収集 Agent、管理 Agent それぞれの設計について述べる。

3.4.2 収集 Agent

更新情報を取り出す手法として、次のようなものが考えられる。

1. 保持している全てのデータを走査して情報 index を作り出す。(探索型)
2. 管理者やサーバからの書き込み/追加/削除を検知して、その都度情報 index に追加していく。(検知型)
3. (なにか他の目的で) サーバが作った情報を利用する。(利用型)

探索型 (1.) はたいていのシステムでは使用可能だと思われる。たいていの情報提供システムは、ある程度まとまった場所に情報を保持しているので、調べなければならない場所が限定される。時間も負荷も一番かかる方法であるが、完全なデータが得られる。

検知型 (2.) は、他の方法に比べて効率的である。更新検知が一番早く、マシンにかかる負荷も少ない。サーバを通して更新された情報はサーバが面倒を見ると言うのは常識的な発想であるが、これを実現するためにはサーバを変更する必要がある。

利用型 (3.) は、サーバがすでに log などの何らかの情報を作っている場合、それを流用するという考え方である。サーバが情報更新/削除全てに関与し、その記録を取っているよ

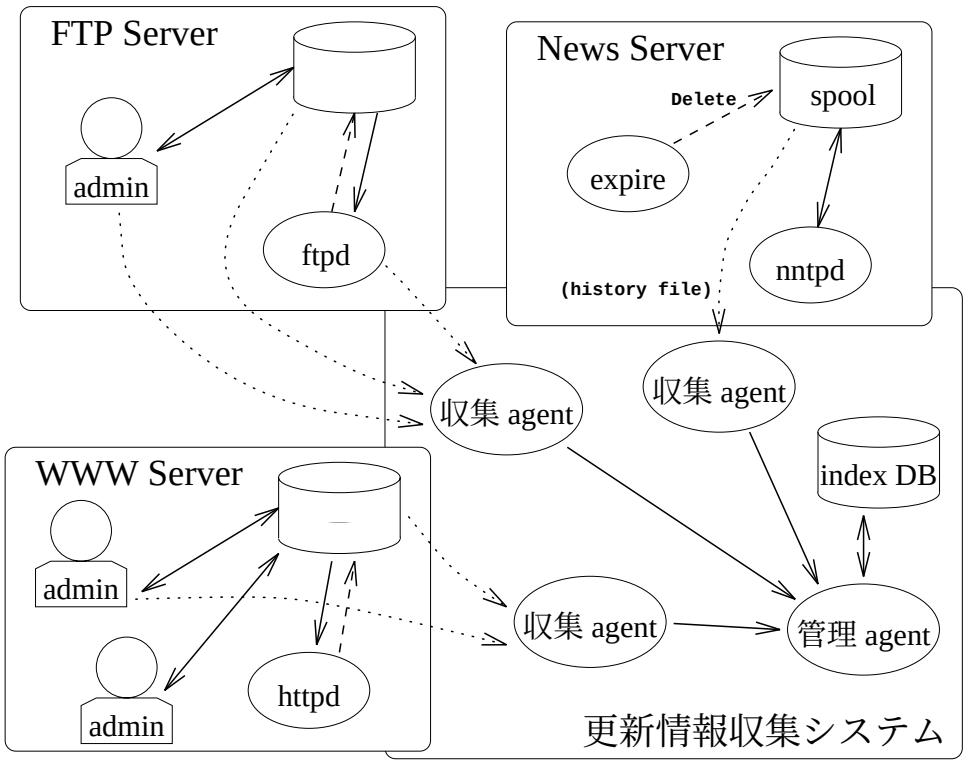

図 3.5: 更新情報収集システムのモデル

うな場合に非常に有効である。管理者は全く手間がかからず、マシンにかかる負荷も少ない。ただし、そういう情報を作らないシステムもあるため、この方法が常に使えるとは限らない。

検知型について、システム (OS やファイルシステム等) で対処してしまうことも考えられるが、既存のシステムに手を加える必要がある。管理者が情報更新をサーバに通知する手段もあり、それはある程度自動化できるので、管理の手間としては現状とほとんど変わらない。

3.4.3 探索型収集 Agent

FTP, WWW, NetNews は全て情報をファイルとして格納し、ディレクトリ階層で分類している。また、その情報はたいてい、あるディレクトリ下にまとめられている。これ以下のファイルの情報を全て走査すれば、そのシステムの持っている情報が網羅できる。

このことから、今回取り上げた各システムでは、探索型の方法が汎用的に使えることが分かる。また、ディレクトリ階層を使って情報を管理していれば、その他のシステムを扱うことも可能である。

探索型収集 Agent を実現するための、ファイル走査をするプログラムを、以下 `fscan` と表記することにする。

3.4.4 検知型収集 Agent

管理用のコマンドを用意して、管理者が情報 (ファイル) の更新作業をする時にそのコマンド群を使ってもらうというのは、現実的でたいした手間もかからないと考えられる。これらのコマンド群は、管理者が普段使っているコマンドと同じ、または似た仕様になっていることが望ましいため、`cp`, `mv`, `rm` などの既存の UNIX コマンドに似た検知型収集 Agent 用のコマンドを用意する。これらのコマンドは、ファイルを所定の場所にコピーし、その情報を管理 Agent へ流す動作を行う。

また、更新したファイルを管理 Agent に知らせるコマンドを用意する。このコマンド自体はそれ以上の処理は行わない。これは `touch` コマンドに似たコマンドであると言える。

これらのコマンドを、以下 `fcp`, `fmv`, `frm`, `ftouch` と表記することにする。

これらは検知型の方法のうち、管理者による情報更新を全て捕まえる手段を提供する。主に管理者によって情報が更新されるシステムに向いている。また、`ftouch` コマンドはほとんどのシステムで汎用的に使用できる。

3.4.5 利用型収集 Agent

利用型の方法を検討するためには、現在使われている各システムの実装を考慮する必要がある。ここでは、それぞれのシステムについて順番に、そのシステムのサーバがどんな情報を管理しているのか調べ、この方法を適用できるかを検討する。

FTP システムではサーバを介してファイルの更新が行われることがある。このファイル更新は、通常、ファイル情報をを全て走査するかシステムに手を入れることでしか検知できない。wu-ftpd [130] というサーバでは、ファイルの更新を `xferlog` ファイルに記録するので、それを処理することで情報更新を検知できる。

NetNews システムでは、サーバを介してのみファイルの追加が行われる。ファイルの削除を行うのは専用のプログラムである。つまり、NetNews システムへの情報の入/出力は全て自動的に検知可能である。ただし、NetNews システムがその情報を保持していなければ、nntpd の部分的な変更が必要となる。現在主に使われている Cnews, INN は、`history` ファイルに更新情報を記録するので、これを利用する事が可能である。

WWW システムで使用できる情報更新の検知方法は運用形態に応じてさまざまなものが考えられる。たとえば、大勢のユーザが各々自分の持っている情報を更新することがあるし、管理者が全ての情報を管理する場合もある。このため、WWW システム一般に対する更新検知方法というものは存在しない。

そこで、WWW システムに対しては、すでに述べたものも含めて考えられる全ての方法を提供することが必要である。

情報はアクセスされてはじめて意味があるので、情報へのアクセスを監視して情報更新を検知する方法も考えられる。さらに、情報提供者が新たな情報を提供する時には、それがうまくできているかを自分でアクセスして確かめる可能性が高いので、httpd が作る `access_log` ファイルを監視して、今までアクセスされたことが無い情報が現れた場合、新たな情報は追加されたとみなせる。また、すでにあるファイルが変更された場合も `access_log` に残るので、情報の変更も知ることができる。この方法では情報の削除を検知することができないが、運用形態によってはこれで十分な場合もあると考えられる。

3.4.6　管理 Agent

収集 Agent から収集された更新情報は、local ネットワークの更新情報管理 Agent に集められる。これは収集された更新情報を整理/管理するものである。収集 Agent は更新された可能性のある情報 index をひたすら集めるだけなので、実際には更新されていないファイルの情報も含まれている。管理 Agent では、その記録を使える形に分析/整理しなければならない。

管理 Agent は、前回までの情報をファイルに記録しておき、新しく来た情報を記録されたものと見比べ、現在の全ファイルの正確な情報 index を作り出す。

情報の所在を一意に示す方法としては、URI が一般的に使われているので、それを利用する。情報が更新されていることは、ファイルの作られた時刻やサイズなどで判断する。これらが一致した場合、そのファイルは更新されていないと判断できる。追加された情報は前回までの記録には無いので、一目瞭然である。

3.5 プロトタイプの実装

以上のモデルの実現するプロトタイプを UNIX システム上へ実装した。実装にあたっては、システム間の違いを吸収できるようにするため、プログラム言語として Perl [131] を使用した。

情報の取り出しを行うのは、図 3.5 の収集 Agent である。収集 Agent には 3.4.2 で述べた数種類のコマンドを用意した。

3.5.1 探索型収集 Agent

探索型収集 Agent 用の `fscan` コマンドは指定されたディレクトリ下の全てのファイルの情報を走査し、そのファイルの更新時間とファイルサイズを記録する。以下では FTP システムへの適用を例にあげて述べる。

ディレクトリを再帰的に下降してファイルを検索するプログラムは、perl のライブラリ、`find.pl` を使う。このパッケージは、与えられたディレクトリ以下の全ファイルの情報を走査しながら、条件に当てはまるファイルを操作することができる。

以下が、条件にあてはまるファイルを見つけ出し、そのファイルの情報を調べている部分である。

```
# 必要なファイルを判別するサブルーチン
sub wanted {
    local($tmp, $type, $size, $time);

    # 普通のファイルか？
    $type = 'f' if (-f $name);
    # ディレクトリか？
    $type = 'd' if (-d _);
    # ファイルでもディレクトリでもなければ抜ける
    return unless($type);
    # file size を調べる
    $size = -s _;
    # timestamp を調べる
    $time = (stat(_))[9];
    # 見つけた情報をリストに格納する
    $tmp = sprintf("%s\t%s\t%d\t%d", $name, $type, $size, $time);
    push(@files, $tmp);
}
```

現在走査しているファイルは `$name` に格納されている。これは `find.pl` の仕様である。

`$name` が通常のファイル、もしくはディレクトリであることを確認して、そのファイルのファイルサイズと更新時間を切り出している。

`@files` には、tab で区切られた file name, file type, file size, timestamp の列が格納されていく。これらは次のようなものである。

file name : 絶対パスで表されるそのシステム内でのファイルの場所。

file type : ディレクトリ '`d`' 又は通常のファイル '`f`'。

file size : ファイルのサイズ。バイト単位。

timestamp : ファイルの更新時間。`time_t` 型の値である。普通、UNIX システムでは `long` 値 (4 bytes) で表される。この値は、グリニッジ標準時刻の 1970 年 1 月 1 日午前 0 時からの経過秒数である。

具体的には以下のようなデータ列になる。

```
/public/ftp/pub d       512     818244970
/public/ftp/pub/GNU     d       512     805518935
/public/ftp/pub/GNU/japanese    d       512     805531395
/public/ftp/pub/GNU/japanese/gdb_jman.tar.Z     f       197063  805531142
/public/ftp/pub/GNU/japanese/gs261d12.tgz       f       73727   805531396
/public/ftp/pub/GNU/japanese/gs261j10.tar.gz    f       155290  805529721
/public/ftp/pub/GNU/mule        d       512     813579442
                                ⋮
```

次に、このデータ列を URI に変換する。この作業は収集 Agent が行う。これは、管理 Agent は収集 Agent が動くシステムの local な情報を知らないためである。URI への変換は、変換テーブルを用意して単純に置き換える操作で行うことができる。

ftpd は anonymous user が login してきた時には、システムを安全なものにするために、まず `chroot` システムコールで anonymous user 用のディレクトリを root (`/`) ディレクトリとし、それ以下のファイル以外へのアクセスを禁止する。したがって、対象としている FTP システムの URI を作り出すためには、anonymous user 用のディレクトリを root ディレクトリとみなして、URI に必要な プロトコルとホスト名の組を付け足せば良い。

上の例では anonymous user 用のディレクトリが `/public/ftp`、プロトコルとホスト名の組が `ftp://ftp.kk.info.kanagawa-u.ac.jp` である。また、UNIX システムでは path が一意に決まらない場合が存在するため、この対応関係を複数指定する事ができるようにする。この例では、`/public/ftp` ディレクトリは他に `/.tmp_mnt/ftp`, `/.mnt_pt/ftp` という別名を持っているので、次のような変換テーブルを用意すれば良い。

```
/public/ftp     ftp://ftp.kk.info.kanagawa-u.ac.jp
/.tmp_mnt/ftp   ftp://ftp.kk.info.kanagawa-u.ac.jp
/.mnt_pt/ftp    ftp://ftp.kk.info.kanagawa-u.ac.jp
```

前述のデータに対してこの変換を行うと、以下のようになる。(一部省略)

```
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub d       512     818244970
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub/GNU     d       512     80551 ...
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub/GNU/japanese    d       512   ...
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub/GNU/japanese/gdb_jman.tar.Z   ...
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub/GNU/japanese/gs261d12.tgz     ...
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub/GNU/japanese/gs261j10.tar.gz  ...
ftp://ftp.kk.info.kanagawa-u.ac.jp/pub/GNU/mule        d       512   ...
                                    ⋮
```

このデータには URI とファイルのその他の情報が含まれている。また、管理 Agent に渡す前に、fscan を実行した時間を更新時間と同じフォーマットで記述しておく。これを管理 Agent に渡せば、fscan プログラムの処理は終了する。

3.5.2 検知型収集 Agent

検知型収集 Agent 用の fcp, fmv, frm コマンドは、それぞれ cp, mv, rm とほぼ同じ働きをする。これらのコマンドの働きは元になったコマンドとほとんど変わらず、元のコマンドに付け加えなければならない動作はどれもほとんど同じなので、ここでは fcp についてのみ記述する。

fcp の使用法は次の通りである。

1. `fcp [ -i ]` *filename1 filename2*

2. `fcp [ -i ]` *filename* ... *directory*

基本的に cp と同じなので、ここでは特に説明しない。以下、*filename1, filename* を *source*、*filename2, directory* を *destination* と書くことにする。

fcp はまず指定されたファイルを読み込み、指定の場所に書き出すという、cp と全く同じ動作をする。この先には時間がかかる処理がいくつかあるので、この段階でユーザへ処理を返し、fork システムコールを使って、残りの処理は子プロセスで行う。

次に、fcp は更新されたファイルの情報を整理 Agent に送る。更新されたファイルの情報は URI 形式にして整理 Agent に渡すため、引数の情報から URI を作り出す必要がある。URI を作るには、相対 path で指定された *destination* を絶対 path に変換する必要がある。UNIX には指定したファイルの絶対 path を得る機能はないので、次のような方法をとる。

1. カレントディレクトリの絶対パスを調べる。

2. *destination* にファイルが指定されていれば、そのファイルがあるディレクトリに、ディレクトリが指定されていれば、指定されたディレクトリに移動する。

3. 移動したディレクトリの絶対パスを調べる。

4. 最初に調べておいたカレントディレクトリに戻る。

具体的には次のようなコードがこれを行っている。

```
# カレントディレクトリの絶対パスを調べる
use Cwd;
$curdir = cwd();

# 指定されているのはディレクトリか？
if (-d $d) {
    # ディレクトリならリストにしまう
    push(@dirs, $d);
} else {
    # ファイルである
    # カレントディレクトリのファイルか？
    if ($d =~ /\//) {
        # カレントディレクトリの絶対パスをリストに入れる
        push(@dirs, $curdir);
    } else {
        # ファイルなら、最後のファイル名の部分を取り除く
        $d =~ s/^(.*)\/.*$/\1/;
        push(@dirs, $d) if (-d $d);
    }
}
# ディレクトリが存在しない？ ので捨てる
# 元のディレクトリに戻る
chdir($curdir);
```

ファイルの絶対パスを URI に変換する手法には `fscan` と同じものが利用でき、これにファイルサイズと更新時間の情報を付加して、管理 Agent へ送る。

管理 Agent へのデータの受渡しは、socket を使ってネットワーク越しに connect し、得られたデータを文字列のまま送ることで行う。これで子プロセスの処理は終了する。

3.5.3 管理 Agent

取り出された更新情報は、URI 形式 + α の形で、ローカルネットワーク上にある管理 Agent に送られてくる。管理 Agent は、情報 index を集めてデータベースを作成し、これを `index_db` というファイルに格納している。管理 Agent は socket の決まった port を開いて待機し、収集 Agent から connect 要求があった場合、`fork` して子プロセスで後の処理を行う。

まず、収集 Agent からの情報を全て socket から読み出し、順番にリストに格納していく。次に `index_db` から、それまでの情報 index を取り出して新しい情報のリストと比較し、以前の情報 index に記録されていない URI は追加された情報であるとみなす。すでに URI が記録されていた場合、情報の更新は、ファイルの更新時間が新しいか、ファイルサイズが異なることで判断する。

情報 index を更新/追加し、`index_db` に書き戻して管理 Agent のデータベース更新処理は終了する。

現在のところ、管理 Agent はクライアントの問い合わせに対して、該当する更新情報を返す機能のみ実装されている。

クライアントは整理 Agent に決まった port でアクセスし、更新時間の値を渡す。整理 Agent は `index_db` から、その更新時間以降に更新されたファイルの index を取り出して送る。これでクライアントは全ての更新情報を得ることができる。

3.6 評価

評価の際にもっとも問題になるのは収集 Agent の処理効率である。実際の評価は Sun SS5 上で行った。

`fscan` プログラムを実行した結果、3,500 ファイル, 500 MBytes のデータを含むディレクトリでは約 30 秒、38,000 ファイル, 1,200 MBytes 程度のディレクトリになると約 7 分 20 秒、`ls -lR` で同様の操作を行った時、500 MBytes では約 26 秒、1,200 MBytes では約 5 分 40 秒であった。ファイルシステムの構造に依存するので一概に言うのは難しいが、一般に、ファイル数が増えるとほぼ線形に走査時間も増える。実際にはファイルシステムの構成上受けるオーバヘッドなどで、パフォーマンスは線形よりも少々悪くなる。

現在運用されている FTP サイトのディスク容量は増えていて、中には 100 GBytes 以上のディスクを使用しているサイトもある。ディスク容量が増えると、ファイル数がそれ以上に増えることは一般的に知られており、そのようなサイトでファイルを全走査する方式を使うにはかなりの時間が必要になる。

この結果をみる限り、perl を使ったことによるオーバーヘッドは問題にならない程度である。今回は `opendir, readdir` システムコールを使っていないため、これらの命令を使えばさらなる高速化が期待できる。

一方 fcp, fmv, frm, ftouch コマンドを使った時は、そのオーバーヘッドはほとんどない。

3.7　結論

今回提案したシステムを使えば、既存のシステムに手を加えることなく更新情報を比較的低いコストで収集し、ネットワークの負荷を軽減できることが確かめられた。

課題として、本文中で触れたいくつかのパフォーマンス改善策の実装/評価があげられる。また、wu-ftpd の xferlog ファイルの監視、httpd の access_log ファイルの監視などの収集 Agent の実装、更新情報を発見された時点でネットワーク上に流してしまう方式の評価なども今後の課題である。

利用頻度が比較的低いため、今回は Gopher, WAIS などのシステムについては触れていないが、これらについても調べる必要がある。

第 4 章

今後の活動

今年度は情報 index 収集機構のプロトタイプの実装を行っただけであるため、今後は、情報 index 整理機構および、情報 index 検索機構のプロトタイプシステムを ISODE パッケージの OSI ディレクトリサービス QUIPU を使って実装し、各機構についてモデルおよび実装方法について検証する予定である。

検証項目としては以下のものを予定している。

4.1 情報 index

- 項目内容
- 保持形態

4.2 情報 index 収集機構

- 収集対象ホストおよびサービスを選択する方法
- 情報 index を構成するのに必要な項目の取得方法
- 更新に対する監視方法

4.3 情報 index 整理機構

- 両方の情報 index 整理方法の妥当性
- 整理に伴うネットワークトラフィック

4.4 情報 index 検索機構

- 検索方法の妥当性

- 検索時のネットワークトラフィック
- 検索に必要な時間