マルチポイント ·マルチサーバ ·ニュースリーダの実装と評価

島袋 政辰　　　　河野 真治
琉球大学工学部情報工学科

マルチサーバ ·ニュースリーダは、複数のサーバからニュースを整合性良く取得するものであり、マルチポイントニュースサーバ は、一人のユーザが複数の計算機からでも rlogin することなく一貫してニュースを読むことを可能にするものである。これらに対応するような、新しいニュースシステムの開発について述べる。このニュースリーダは history データベースだけを保持する中間的なサーバを利用する。本論文ではデータベースの構成、サーバの実装と評価について述べる。

Multi-server Multi-point News System

Masatatsu Shimabukuro, Shinji Kono
Faculty of Information Engineering, University of the Ryukyus

Multi-server news reader can get news from net-news servers consistently. Multi-point news reader makes it possible to read net-news from different computers without remote login. We have developped multi-point multi-server news reader using an intermidiate server, which has history database. In this paper, we describe organization of the database and an implementation of the intermidate server and its evaluations.

1　はじめに

ネットニュースは、ニュースサーバが配布しているいろいろな情報をユーザが受け取りそれに対しての議論をユーザの間で行われるシステムである。簡単に新しい情報をいち早く手に入れられるので多くのユーザの交流の場になっている。

ユーザは今では複数の計算機を使用するのが普通であり、決まった計算機からニュースを読み出すとは限らない。また、現在では情報の多種多様化に伴い多くのニュースサーバが立ち上げられている。そこで、複数の計算機から remote login することなく一貫してニュースを読むことができるマルチポイント ·ニュースリーダや、複数のニュースサーバから記事の重複や混乱なくニュースを読むことのできるマルチサーバ ·ニュースリーダが必要であると考えられる。

複数のサーバが必要になる理由は、ネットニュースシステムは公開性が高いため、ローカルなニュースグループを他のユーザに読まれないように独自でニュースサーバを管理しなくてはならないからである。琉球大学でも world wide な comp を始めとする数多くのグループの他、大学全般に渡る ura、情報工学科の ie、さらに研究室のニュースグループ urcl がある。それを配布しているニュースサーバとして琉球大学では spn、研究室では hornets という計算機が管理している。

このような状況で、一つのサーバから他のサーバを含む形でサーバを運営するためには他のサーバのニュースをすべてコピーする必要がある。サーバ側では、配布するためのニュースを他のニュースサーバからニュースの内容を spool に保存し、そのニュースに対応しているメッセージ ID を history データベースに保存している。そのため、ニュースサーバが保持するデータはかなりのものになっていて、琉球大学でも history:spool=54M:115M のデータを保持している。小規模なローカルなニュースグループを作りたくても、fj(20M) とかを取ってくるディスクがないといった問題が起こってしまう。

ニュースサーバがローカルなニュースしか配布せ

ずに、他のニュースサーバに読みたいニュースグループがある場合はユーザがそのサーバに接続しなければ読むことができない。しかし、この場合は、二つのサーバ上の history データベースや記事番号は一致していないので、記事の重複や、記事を読む順番の混乱などの問題が生じてしまう。

また、現在は一人のユーザが複数の計算機にアカウントを持っているのが当り前となっている。しかし、ニュースの未読、既読情報 (.newsrc) はいつも読んでいる計算機にしか保存されず、別の計算機から読む場合は.newsrc を持ってくるか rlogin で.newsrc がある計算機に移らなければならない。

今のインターネットの世界では、複数のニュースサーバに対して複数の計算機 (ユーザ) に対応しているようなリーダが望ましい。本研究ではそのような複数のニュースサーバと、複数の計算機上のリーダの間に立つ中間的なサーバを導入する。中間サーバは、ただ中継をするだけではなく記事に関する大切な情報だけを保存し、必要に応じて複数のニュースサーバに接続する。また、.newsrc を読む計算機に関わらず提供する機能を持つ。

そのサーバによりユーザが複数のサーバに接続する手間が省ける。また、ニュースサーバもローカルなニュースグループだけを配布し、別に他のニュースサーバのニュースを受け取る必要がないのでサーバ自体の資源も節約できるようになる。

2　ネットニュースの仕組み

2.1　ネットニュースとは

電子メールはプライベートにメッセージを交換するシステムであり、ネットニュースは掲示版を使って色々な人の意見によって議論を進めていく通信手段である。インターネット上で世界中のユーザが公開された形で議論を行うためのシステムである。

複数のニュースサーバがニュースを交換しあう、大きな論理的なネットワークを構成する。そのネットワークを一括して管理しているサーバはなく、各々のニュースサーバで管理しているため分散管理体制としてネットワーク全体を管理している。(図 1)

ニュースシステムを運用している計算機のことをニュースサーバといい、ユーザはそのニュースサーバが配布しているニュースグループを購読して参加することができる。

ネットニュースの基本機能はニュースリーダを使ってニュースを読むことである。その他に投稿、フォローアップ、リプライ、キャンセルがある。投稿はニュースの記事の発信で、配布範囲にしたがって各ニュースグループに投稿することができる。フォローアップは読んだ記事に対してニュース上で応答し、キャンセルは投稿した記事を取り消す。キャンセルできるのは投稿した本人しか出来ないようになっている。

投稿されたニュースをニュースサーバ間でメッセージ ID やニュースの本文を送受信し、history にそのニュースの保存期間、spool に本文を保存し、その保存されたニュースをユーザに配布するシステムになっている。

ニュースシステムとユーザの間での記事の送受信は色々あるが、現在ではネットワークを使った NNTP(Network News Transfer Protocol) が主流となっている。(図 2)NNTP はニュースリーダによってニュースを読む際にそのニュースに関する情報をやりとりする時に用いられる通信規約である。

また、ネットニュースの記事の形式に関する文書 RFC-1036 に基づいて説明する。電子メールの形式を定めた RFC-822 とよく似ているためユーザが扱いやすく、またニュースと電子メールの乗り入れが容易にできるという利点がある。ネットニュースの記事はヘッダ (header) とボディ(body) に分かれていてヘッダにはユーザに意味があるものとニュースシステムに意味するものがある。ボディはユーザが投稿した記事の本文が書いてあってニュースシステムは一切関与しないようになっている。

2.2　INND の仕組み

ネット・ニュースシステムの代表的なものに、Cnews、Bnews、INN というのがある。本研究では

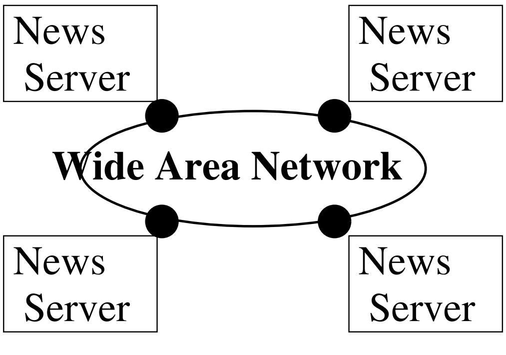


図 1: ネットニュースの全体像

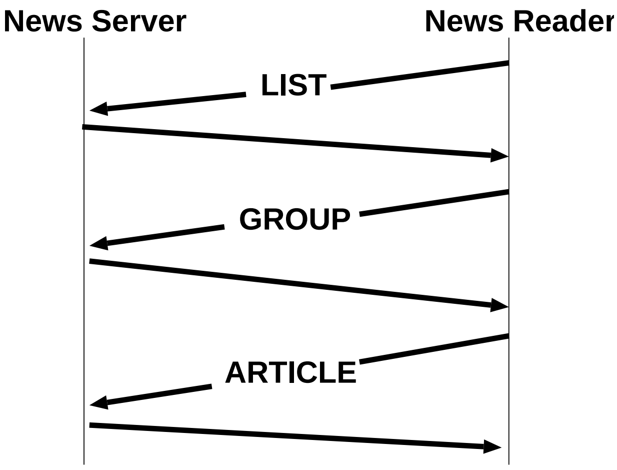


図 2: NNTP

INNというニュースシステムを利用しているのでINNについて説明をする。

(INN)InterNetNewsは完全なUsenetのシステムであり、パッケージの要はinndサーバである。ソケットはストリームで、ポートはnntp:119である。またIPアドレスはINNを起動しているサーバのIPアドレスを使う。

INN自身はニュースを送ることはしないのでニュースを送り出すようにその他のプログラムに指示をだす。それで実際の配送はnntpsendによって行われている。これらの設定はnewsfeeds、nntpsend.ctl、hosts.nntp、expire.ctlファイルによって設定する。newsfeedsには配送するニュースグループ、パス名を設定し、nntpsend.ctlは配送する計算機のフルドメイン名を設定する。それによってnntpsendはニュースを配送するのである。hosts.nntpは配送してくれるニュースサーバのフルドメイン名を設定し、ニュースを受信する。expire.ctlはニュースの保存期間を設定する。

また、ニュースリーダから接続された場合、inndはnnrpdプロセスを生成(fork)し、接続をnnrpdに渡し、innd自体はニュースの配送に集中するのである。(図3)

3 マルチポイント・マルチサーバにする利点と欠点

マルチサーバでは、複数のニュースサーバに接続するためニュースの本文を持つ必要はない。なぜなら、ユーザからニュースの本文に関する要求にはニュースサーバに接続し、そこからニュース本文を取り出しユーザに答えることができる。ただ、ニュースの保存を知っておくためにメッセージIDの保持は必要である。

マルチポイントでは、アカウントを持っているどの計算機からニュースを読むとしても個人情報(未読、既読)は変わらない。しかし、今のNNTPに対応している既存のニュースリーダ(mnewsやgnuなど)は.newsrcから個人情報を手に入れているため既存のニュースリーダは使えない。しかし、マルチサーバとしては今までと同じように使うことはできる。

また、データベースの使用により読みたくない記事を配布しないようにすることができるが、(SPAM対策)データベースを使用するため起動時に巨大なデータを複数のニュースサーバから取得する必要があり時間がかかってしまう。その時間に関しては6章の実装で述べる。

これらの利点 欠点をまとめると以下の通りになる。

- 利点
  - マルチサーバに限れば従来のニュースリーダが使える
  - spoolを持たない。(データ量を減らすことができる)
  - SPAM対策として使える
- 欠点
  - マルチポイントでは既存のニュースリーダが使えない
  - 起動時に巨大なデータを複数のニュースサーバから取得する必要がある

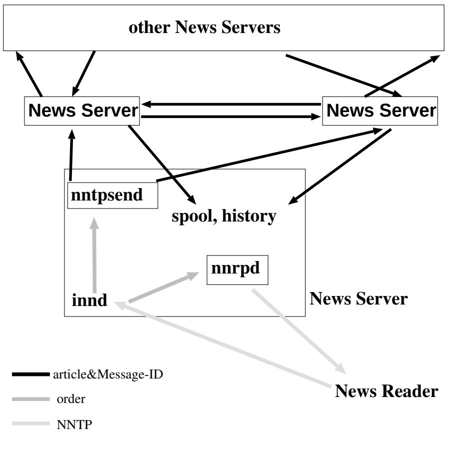


図 3: INN の動作

4　マルチポイント・マルチサーバについて

4.1　データベース

従来のニュースサーバにおけるデータベースにはhistoryがある。それに保存されている内容は以下の通りである。(表1)

値	意味
<6ck14e$ru5@dolphins.cr.ie.u-ryukyu.ac.jp>	メッセージID(key)
887982030˜-˜887949630	保存する期間
dolphins.test/19	ニュースグループ名/ニュースナンバー

表1: 従来のhistoryデータベースの構造

マルチポイント・マルチサーバではニュースサーバが保持しているspoolを保持せずに、新しいhistoryデータベースだけを保持することにする。よって以下に示すのがマルチポイント・マルチサーバにおけるhistoryデータベースの内容である。(表2)

値	意味
<6ck14e$ru5@dolphins.cr.ie.u-ryukyu.ac.jp>	メッセージID(key)
dolphins	ホスト名
dolphins.test	ニュースグループ名
19	ニュースナンバー

表2: マルチサーバのhistoryデータベースの構造

メッセージIDをキーとして、どのニュースサーバに対応しているか知るためにニュースサーバ名、メッセージIDがどのニュースグループに属しているか知るためにコンテンツとして保持する。ニュースナンバーはナンバーの統一化を行なうために保持している。

またマルチポイントとしてはhistoryのデータとしてユーザ名(ID)を付け加え、読んだ記事に対してはユーザ名(ID)が保存されるといった仕組みになっている。

4.2　プロトコル

ニュースサーバとユーザからの接続に対するマルチポイント・マルチサーバの動きをプロトコルにし、そのプロトコルを図4に示す。

マルチポイント・マルチサーバを起動して最初にすることは、データベースに必要なデータをNNTPのLISTとSTATを使って全て取得することである。(それにかかる時間は6章の実装で述べる) ユーザ側からの命令に対してはニュースに関する情報以外はマルチサーバだけで行ない、ニュースに関する情報(ARTICLE、BODY、HEAD)は、そのニュースを保持しているニュースサーバに接続して、そのままユーザに流すといった流れになっている。

5　実装

5.1　マルチポイント・マルチサーバの作成

プログラム全体の入出力は、ユーザ側からは1450のポートを使って接続し、ニュースサーバにはもちろんNNTPポートである119を使って接続する。

ニュースサーバにLIST(NNTP)を送ってニュースグループのデータを受け取り、それに従ってSTAT(NNTP)を使い、1つ1つのメッセージIDをndbmを使ってhistoryファイルに保存する。(図5)

historyデータベースを作成する時、リストを作りその中にニュースサーバ名、ニュースグループ名、保持しているニュースで最も古いニュースナンバー、最も新しいニュースナンバー、モードを登録しておく。それはLISTやGROUP、LISTGROUPといったニュース本文には関係のないNNTPのプロトコルをマルチポイント・マルチサーバで同じように動作するためである。例えばユーザからLISTの命令がきた場合、リストされているデータをそのまま送ることができる。またGROUP newsgroupの命令の時は、リストからnewsgroupを探して、このサーバが作成したhistoryデータベースに保存されているnewsgroupのニュースの数とリストにある古いニュースナンバー、新しい

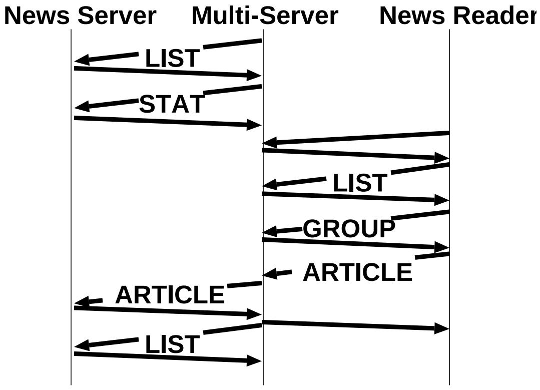


図4: マルチポイント・マルチサーバのプロトコル

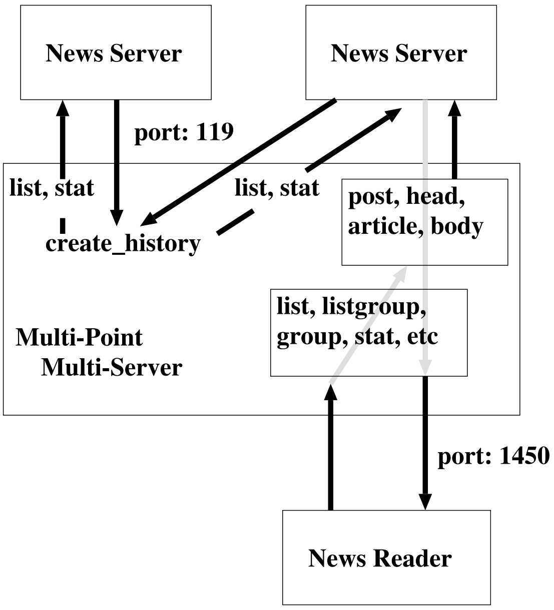


図 5: プログラム全体の入出力

ニュースナンバーをユーザに送るようにすればよい。(NNTP で GROUP をすると 保持しているニュースの数 最も新しいニュースナンバー 最も古いニュースナンバー ニュースグループ名が返ってくる) そのリストを使い history データベースを作成する。しかしマルチポイントマルチサーバは接続しているニュースサーバが複数あるため、複数のファイルポインタが必要になってくる。そのため、サーバにもリストを作りそれによって複数のサーバを使い分けることにしている。そのリストはニュースサーバ名、ディスクリプタ、ファイルディスクリプタが保持されている。

それらのリストを使って history データベースを作成する。ニュースグループ別のリストからニュースサーバ名を探しだし、そのニュースサーバ名からファイルポインタを探してそこに STAT(NNTP) をして、メッセージ ID を取り込む。(図 6)

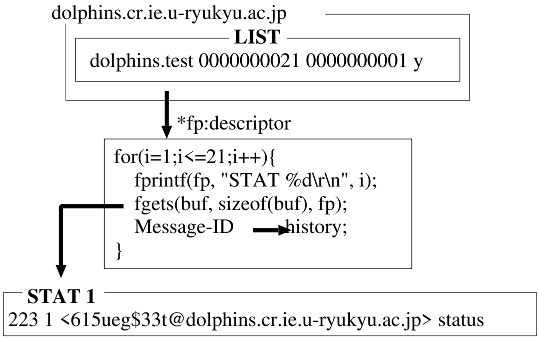
次にユーザからの NNTP のプロトコルに答えるようなプログラムを作成する。ニュースグループ別のリストと history データベースから実装することのできるプロトコルを作成すると実装できるのは LIST, GROUP, LISTGROUP, STAT, QUIT, MODE READER, HELP, LAST, NEXT である。これらのプロトコルはニュース本文を必要とせず、メッセージ ID やデータベースに保存されている情報だけで答えることが可能なので、ニュースサーバへのアクセスは無用である。例えば、STAT 2 は history データベースからニュースグループとニュースナンバーに対応するメッセージ ID を探して、そのメッセージ ID を返すのである。同じように LISTGROUP はニュースグループ別リストからニュースグループに対応するニュースナンバーを小さい数から返すようにすればよい。

今度はマルチポイント ·マルチサーバだけでは実装することのできない NNTP のプロトコルを実装する。実装できない NNTP のプロトコルとは ARTICLE, HEAD, BODY, POST である。それらのプロトコルは spool を持たないマルチポイントマルチサーバでは対応することができない。例えば ARTICLE 2 の場合、そのプロトコルの前に LISTGROUP または GROUP によってニュースグループを指定してそのニュースグループとニュースナンバーに対応するメッセージ ID をデータベースから探しだす。そして、ニュースグループに対応するニュースサーバに接続し、そのメッセージ ID を使ってニュースサーバから spool に関する情報を手にいれそのままユーザに手渡すのである。また、ARTICLE メッセージ ID の場合は、データベースからこのメッセージ ID を探しそれに対応するニュースサーバに接続し、spool に関する情報をユーザに渡す。(図 7)

ユーザからのアクセスについては UNIX の select を使って管理するようにした。なぜなら select は複数のファイルディスクリプタを管理することができるからである。それによって複数のユーザからの接続に答えるようにできている。

6　実装に対する評価

研究室にあるニュースサーバ (hornets, dolphins) とマルチポイント ·マルチサーバのデータ量の差を比べる。ニュースサーバのデータ量は history データ



```
dolphins.test 0000000021 0000000001 y
```

```
for(i=1;i<=21;i++){
    fprintf(fp, "STAT %d\r\n", i);
    fgets(buf, sizeof(buf), fp);
    Message-ID    →history;
}
```

```
223 1 <615ueg$33t@dolphins.cr.ie.u-ryukyu.ac.jp> status
```

図 6: history ファイルの作成手順

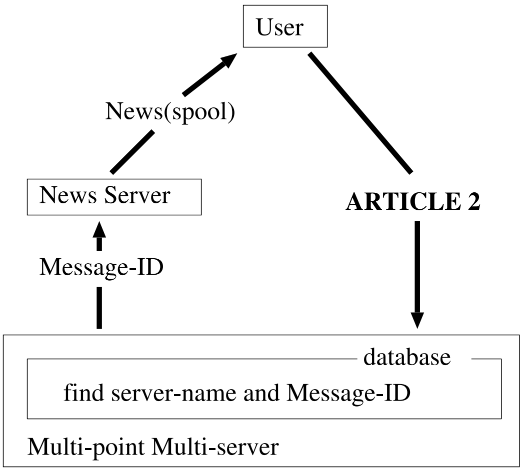


図7: spoolに関するプロトコルの実装

ベースとspoolのデータ量を示し、マルチポイント・マルチサーバは独自のデータベースのデータ量を示す。(表4)見ても分かるようにデータ量が減っているのが分かる。表3と表4でのデータ量の調べた方法は、ニュースサーバの場合historyファイルを”ls -l”で調べ、spoolは”du”を使って調べた。またマルチポイント・マルチサーバはデータベースを作るのにndbmを使っているので実際のデータ量の約1/4を表には載せている。

ホスト名	history データベース (byte)	spool(k)
hornets	1171595	28965
dolphins	3204	61
合計	1174799	29026

表3: ニュースサーバのデータ量

サーバ	データ量
ニュースサーバ (表3の合計)	30200799(byte)
マルチポイント・マルチサーバ	about 573440(byte)

表4: データ量の差

また、ニュースを読むまでの時間はニュースサーバに接続するのと、マルチポイント・マルチサーバに接続するのではさほど時間はかからないが、起動にはかなりの時間を必要となる。表4だけのデータを取り込むだけでも4分5秒の時間がかかった。(timeを使って調べた時間) なぜならマルチポイント・マルチサーバはニュースサーバに登録されているメッセージIDを全て取り込んでいるのでニュースの量が多ければ多いほど起動に時間がかかってしまうのである。琉球大学におけるニュースサーバのデータ量は1章でも述べたようにhistory:spool=54M:115Mである。そのようなニュースサーバからデータを受け取るとするとデータ量だけでも40倍近い時間がかかるだろう。

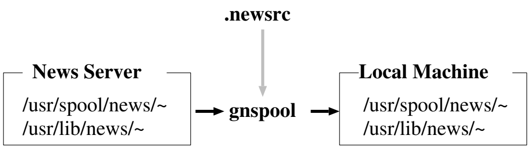


図8: gnspoolの動き

その問題の解決方法の1つとして、データベースに保存するニュースグループを指定するようにし、その他のニュースグループに関しては起動後にforkして取り込むといった形にしてもよいだろう。

ニュースの更新についてはユーザがニュースを読んでいるときに更新するようにしているが、この点に関してもまだ色々試すことができるのでニュースの更新についても最も最適な方法を見つけ出したい。

マルチポイントに関しては既存のニュースリーダは使えないといった欠点があるのだが、既存のニュースリーダも使えるようにしたい。その方法としてはニュースリーダを立ち上げる前にユーザ側で.newsrcを作成するようなプログラムをはさんでサーバ側から.newsrcを作るようにする。しかし各計算機に同じような.newsrcを作るのは能率が悪いので、(.newsrc:70k *ls -lによる) サーバだけでなくニュースリーダも開発するのが望ましい。

7 gnspoolとDNASとの違い

7.1 gnspoolとは

.newsrcに従って未読の記事をローカルディスクに取り寄せるためのツールである。この時、ユーザがネットニュースをローカルディスクで利用したことがない場合はかなりのニュースデータを取り寄せることになるので注意が必要である。(図8)

未読のニュースをローカルディスクに取り寄せるためNNTPを使用せずにニュースシステムにアクセスするようなニュースリーダはニュースサーバをアクセスしているのと全く同じように読むことができるのである。

しかし、個人の未読ニュースしか取り寄せないた

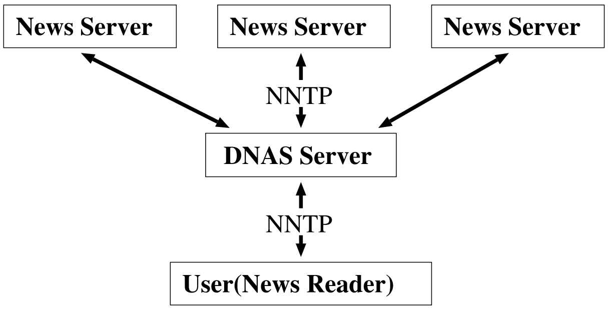


図 9: DNAS の動き

め複数のユーザで使うと結局ニュースサーバとあまり変わらずほとんどのニュースを取り寄せることになってしまう。

7.2 DNAS とは

DNAS(Distributed NNTP servers Service) は、ネットワーク上を分散しているニュースサーバに対して NNTP を通じて同時アクセスが可能なサービスである。(図 9)

ユーザ側への接続も NNTP を使っているので今までの NNTP 対応のニュースリーダを使うことができる。NNTP は記事の read、post に必要な ARTICLE、BODY、GROUP、HEAD、HELP、LIST、NEXT、POST、QUIT、LAST、XHDR、STAT についてのみサポートしている。

また、DNASD の内部状態を答える DNASSTAT コマンドがついている。

7.3 gnspool と DNAS との違い

gnspool の場合は.newsrc から未読の記事を探しだし、その未読の記事を取り込むシステムになっている。しかし、.newsrc に従うため複数のユーザに対して使えない。

また、DNAS の場合はニュースを読むときにニュースサーバに接続して、ニュースを読むことができる。しかし、読んだニュースはローカルディスクに保存されるようになっている。

マルチポイント·マルチサーバはメッセージ ID を保存していて、そのメッセージ ID を使ってニュースサーバにニュースに関する情報を手に入れる所が gnspool と DNAS とは違う。

8 まとめ

本研究では、資源の確保やユーザ·インターフェースを考えた「マルチポイント·マルチサーバ」を導入した。今までのニュースサーバに対する問題を解決案としては新しいものである。

今回開発したサーバによって、今までのニュースシステムの仕組みといった視点から、ニュースデータを取り込むようなことはしなくてもよく、ローカルなニュースを配布するだけでよくなった。それによってニュースサーバ自体の資源もかなり確保できるようになった。

また、ユーザの立場においても読みたいニュースのためにニュースサーバを変えて接続する必要もなくなった。しかし、サーバと呼ぶにはまだ能率が悪いのが現状である。

これからの課題としては、7 章で述べた問題を解決することが最重要課題である。また、それら以外にも修正していく点は多数あるだろう。

参考文献

[1] 安室 貴史, “動的木構造を用いたニュース·リーダの開発,” July 3 1995.

[2] 山口 和紀, 千 旭, 中村 敦司, 新城 靖, 西山 博泰, 古瀬, 一隆, 石川 佳治, 佐々木 重雄, 林 謙一, 荻原 一隆, 金谷 英信, 鈴木 孝幸, 黒石 和浩, “The UNIX Super Text. 技術評論社” 1992.

[3] Rich Salz, “Inter Net News: Usenet transport for Internet sites,” 1992.