サーバー構築/運用術

第7回 ネット・ニューズ(前編)

Web掲示板やメーリング・リストにユーザーを奪われて衰退しつつあるネット・ニューズですが，依然有益な情報も流れていますし，ニューズ・サーバーの構築・運用はネットワークおよびサーバー管理を学ぶ上では避けて通れない技術と言えるでしょう。今回はネット・ニューズの仕組みについて紹介します。

(ケイ・ラボラトリー　仙石 浩明)

この連載を始めた直後*1に転職して設立準備を始め，連載の冒頭でも何度か設立過程などをお話したケイ・ラボラトリーですが，おかげさまでこのたび設立1周年を迎えました。無茶苦茶忙しかったこの1年半，連載を継続できたのは読者の皆様のおかげです。厚く御礼申し上げます。

5人で始めたケイ・ラボラトリーも，今や58人*2もの大所帯です。人を増やすにあたって私自身，何十人もの人を面接したのですが，中には私の連載の愛読者だと言ってくれる人*3もいて，ライターみょうりに尽きます。

人が増えただけでなく，いろんなことがありました。創業期の企業に特有と思われることもありましたし，もちろん一般の会社にありふれたこともあるでしょうが，いずれにせよ前職である大企業の研究所にとどまっていては経験できなかったことが多かったのではないかと思います。

困難なことにぶち当たるたびに，試行錯誤で，また周囲の皆様から教えられ助けられてなんとかやってきました。この先，ケイ・ラボラトリーが発展していくかどうかまだまだ予断を許しませんが，今までの経験を基に何とか対処していける気がしています。わずか1年半の経験なのですが，それだけの自信が得られるほど濃密な体験でした。

ネット・ニューズ

GnutellaやNapsterなど，ピア・ツー・ピア(P2P)が注目を集めています。P2Pは最先端の技術のように思われがち*4ですが，2つのコンピュータが対等な役割を果たすという意味ではインターネット自体が完全分散型のP2Pネットワークですし，インターネット*5とほぼ同時期に誕生したネット・ニューズ*6は，分散型の情報共有システムという点においても，現在もてはやされているP2Pによく似ています。

ネット・ニューズは，P2Pで接続された掲示板システム(BBS＝Bulletin Board System)で，どこか1つのBBSにメッセージを書き込めば，相互接続された他のBBSにも順次配送されていき，どのBBSでも同じメッセージを読むことができる仕掛けになっています。利用者側から見ればBBSの集合全体が1つの巨大なBBSであるかのように見えます。

---

*1　実際，この連載の第1回目(2000年4月号)は，「転職します。のっけから唐突ですが，おかげで鬼のように忙しい今日このごろです。」という文で始まっています。
*2　アルバイトなどをすべて含めると80人を超えます。100人を突破してしまうのも時間の問題かも知れません。
*3　そのうちの何人かは採用して，一緒に働いています。
*4　大昔からある技術に勝手なキーワードをつけて扇動するのはマスコミの常とう手段ですね。P2Pの前には，クライアント・サーバー・システム(CSS)が最先端技術であるかのように宣伝された時期もありました。当時は「時代はP2PからCSSへ」といった感じの宣伝があふれていました。歴史は繰り返すのでしょうか。
*5　1969年に実験が開始されたARPA(Advanced Research Projects Agency)ネットが起源とされています。TCP/IP技術が確立したのは1982年のことです。その後1986年にNSF(National Science Foundation)ネットが発足し，続いて世界各国のネットワークが相互に接続するようになって，全体としてインターネット(the Internet)と呼ばれるようになりました。
*6　1979年の誕生当時は，USENETと呼ばれていました。日本でも1984年にJUNETが発足しています。誕生当時は，メッセージを相互にやり取りできる，というネットワーク的側面に注目したネーミングだったわけです。ところがネットワークがメールやニューズだけのものでなく，telnetやftpそしてWorld Wide Web をはじめとするさまざまなアプリケーションが使われるようになって，ニューズは

掲示板は分野別に分かれていて，それぞれはニューズ・グループと呼ばれます。「ニューズ」からの連想で，メッセージは「記事」と呼ばれ，メッセージを書き込むことは「投稿」，ある投稿記事に対して書き込むコメントは「後報（フォローアップ）」[*7]です。なお，ネット・ニューズにおいて，各BBSのホストはニューズ・サーバーと呼ばれます。「サーバー」であるのは利用者が記事の読み書きに用いるクライアント（ニュース・リーダー）に対してであり，ニューズ・サーバー同士は対等な関係，つまりP2Pです。

ニューズ・サーバー間で記事を配送するための相互接続の方法は何でもよく，ネット・ニューズ登場当時の1970年代は電話回線上のUUCP[*8]が主流でしたが，インターネットの普及にともないTCP/IP上のNNTP（Network News Transfer Protocol）が主流となりました。

配送経路の変遷

ネット・ニューズの発展，そしてインターネットの爆発的普及によって，ニューズ・サーバーの接続形態，すなわち記事の配送経路は，大きく変化してきました。

●UUCPによる配送

ネット・ニューズの登場当初は，ニューズ・サーバーは他のニューズ・サーバーに対し，数10分～数時間間隔でダイヤル・アップしUUCPで記事を相互に配送していました。電話回線を使っていたので遠距離だと電話代がかさむため，なるべく地理的に近いニューズ・サーバー同士を接続していました（**図1**）。

ダイヤル・アップのタイミングは，接続2者間だけで決められ，サーバー群全体としての同期などは特に考慮されていないので，運が悪いとサーバーから次のサーバーへ配送されるまで長い時間待たなければなりません。ネットワークの端から端まで伝わるまで何日もかかるということも珍しくありませんでした。

そこで，伝ぱにかかる日数を便宜上1週間[*9]と定め，例えばニューズ・グループにおける議論で合意を得るには，合意の呼び掛け記事がネットワーク全体に行き渡るために1週間，反対者がその記事を見て反論記事を記事を書き，その反論記事が戻ってくるのに1週間，つまり合計2週間は最低でも待たなければならない，などという合意が形成されました。

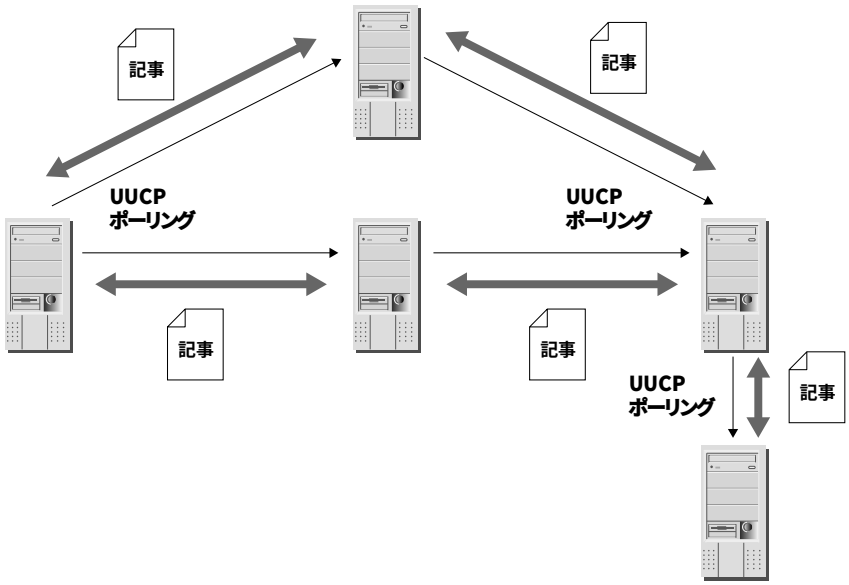

図1　UUCPによる配送

ネットワークの一アプリケーションという立場に転落し，アプリケーション的側面に注目したネーミングである「ネット・ニューズ」という言葉が生まれました。

＊7　パソコン通信では，コメントのことを「応答（レスポンス）」と呼んでいたため，これを縮めて「レス」というスラングが生まれました。インターネットの普及にともなって，それまでパソコン通信に慣れ親しんでいた人たちがネット・ニューズに参加するようになったとき，パソコン通信の世界だけで通用する「レス」をネット・ニューズの世界で何の説明もなく使ったため，いわゆる「レスって何ですか？」摩擦を引き起こしました。

＊8　「Unix to Unix CoPy」の略。UUCPについては本誌2000年7月号の連載「実践で学ぶ，一歩進んだサーバー構築・運用術」第4回「UUCPの活用」を参照してください。

＊9　現在のネット・ニューズでは，数秒もあれば世界中のニューズ・サーバーに伝わってしまいます。

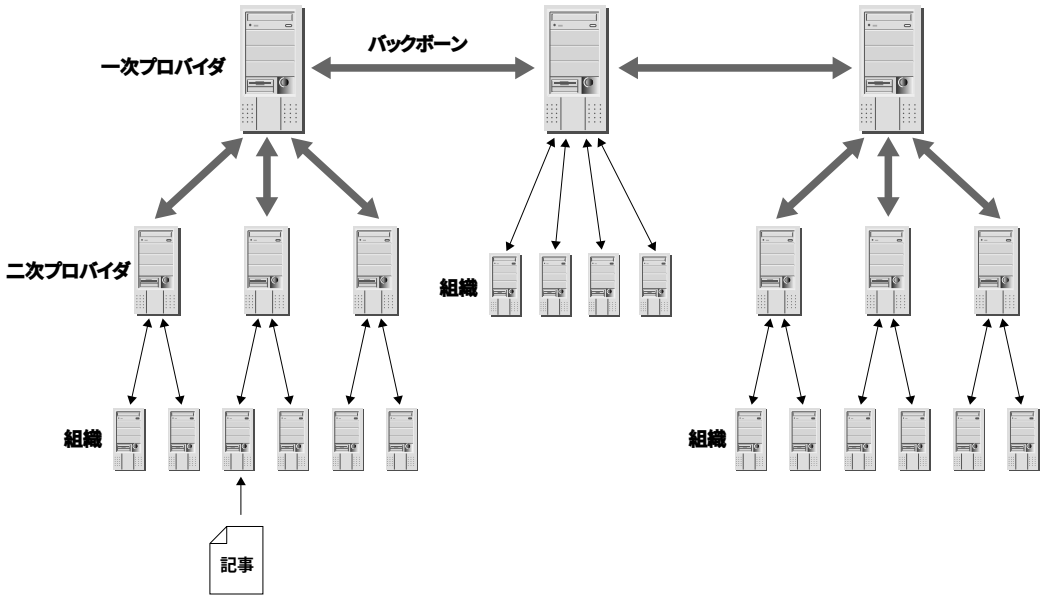

図2　インターネット・トポロジに沿った配送経路

●インターネット・トポロジに沿った配送経路

1991年に，インターネット・サービス・プロバイダによってインターネットへの接続サービスが提供されるようになる＊10と，広帯域の通信線を持ち，インターネットのバックボーン（背骨）的役割を果たす1次プロバイダと，1次プロバイダへのアクセスを小分けにしてその分安価にインターネットへのアクセスを提供する2次プロバイダや，さらに2次プロバイダへのアクセスを小分けにする3次プロバイダなどが現れ，**図2**のようなツリー構造のインターネット・トポロジができあがりました。

インターネットに接続しようとする組織あるいは個人は，これらのプロバイダのいずれかにアクセスすることになります。それぞれのプロバイダは，ネット・ニューズをサービス・メニューの一つとして提供し，それぞれが上位のプロバイダのニューズ・サーバーへ接続していたので，インターネット・トポロジに沿った配送経路ができ上がりました。ネット・ニューズがインターネット上のアプリケーションである以上，配送経路はインターネットのトポロジに一致することが理想と言えるでしょう。

●インターネットのトポロジを無視した配送経路

ネット・ニューズは，記事を中継する数多くのニューズ・サーバーによって成り立っているので，それらのサーバー資源を浪費しないよう，読者にとって有益な記事のみを簡潔に書くのがマナーとされてきました。ネット・ニューズの参加者が少ないうちは自浄作用が働き，記事の流量はさほど問題にはならなかったのです。

＊10　当時はまだ利用料金が高額で，利用者は一部の企業などに限られました。2次，3次プロバイダが現れて安価な接続サービスが提供され，インターネットのユーザー数が爆発的に増加したのは，インターネット元年と呼ばれる1995年ごろのことです。

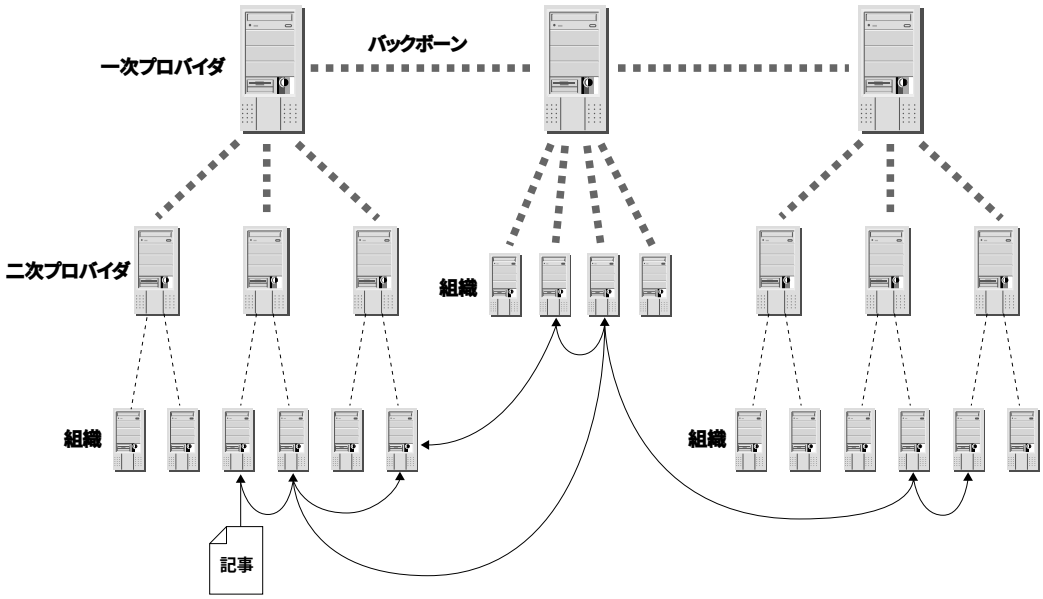

図3　プロバイダに頼らない配送網

ところが，インターネットの普及がさらに進み，ありとあらゆる人たちが参加するようになって，参加者の中にはモラルが低い人たちも増えてきました。ネット・ニューズを利用するだけで貢献しようとしないただ乗りユーザーが増えてきただけでなく，多くの参加者にとって無意味なデータを大量に投稿する行為が横行しました。その中には，著作権法やその他の法令に違反する画像/音楽データやソフトウエアの違法コピーなども含まれていました。

文字のみからなる記事に比べれば，画像やソフトウエアなどのデータは極端に大きいので，ネット・ニューズ全体の流通量は一気に激増しました。プロバイダが運用するニューズ・サーバーの中には，流通量の激増に対応できずに記事の流通が滞るところも出てきました。技術力に定評があったプロバイダでさえ，大量の記事の洪水に対して有効な対策を講じることができず＊11，記事が伝わるのに何日もかかるという事態になったのです。

こうなってくると，プロバイダのニューズ・サーバーを利用する限り，最新の記事を読むことができず，投稿されてから何日も経ってから記事が届くといった有り様で，ニューズ・グループで行われる議論への参加もままなりません。そこで**図3**のように，プロバイダを介さず＊12，組織間で直接記事を流通させる配送網が作られました＊13。

もちろん，プロバイダでさえ，さばききれないネット・ニューズの記事すべて＊14を組織間で流通させることは困難なの

＊11　インターネットの爆発的普及にともなってネットワーク技術者が不足していたことも理由の一つだったのでしょう。
＊12　もちろん，データ自体はプロバイダの回線を流れるのですが，プロバイダが提供するニューズ・サーバーは利用しないということです。
＊13　私が運営するサイトGCDに流入する記事の配送経路をグラフ化したものが，http://www.gcd.org/news/feedmap/Welcome.ja.htmlで参照できます。
＊14　2001年現在，1日当たり約50Gバイト，すなわち1秒当たり600Kバイト以上の流量があります。ここ数年間，毎年約30%増の勢いで増え続けています。

で，必要なニューズ・グループだけを流通させることになります。例えば，日本語が主に使われているニューズ・グループであるfjだけなら1日当たりわずか数Mバイト＊15ですから，個人が運営するサイトでも余裕で配送できます。

配送の仕組み

図4にネット・ニューズの記事の例を示します。メールと同様，ヘッダー部と本文とから成ります。「From: 」「Subject: 」「Date: 」「Message-ID: 」「Lines: 」などのフィールドはメールと同様です。「Newsgroups: 」はニューズ・グループの指定で，この例では私が運営しているサイトGCDのローカル・ニューズ・グループ「gcd.test」を指定しています。

Path: フィールド

「Path:」は，この記事がどのニューズ・サーバーを経由して配送されたか記録するためのフィールドです。この例では，記事を投稿したサーバーgcd.org上でネット・ニューズを読んでいるので，gcd.orgと投稿者のユーザー名sengokuだけが記録されていますが，もしこの記事を別のサーバーで読んだとしたら，Path: フィールドは例えば**図5**のようになっていることでしょう。

この例から分かるように，ニューズ・サーバーを経るごとに左側に経由したサーバー名が追加されていきます。この場合だと，まずgcd.orgに投稿された後，news.ksw.feedmania.orgへ配送され，次にnewsfeed.mesh.ad.jpに配送され，t-newsgw1.odn.ne.jpに届いた，ということを表わします。

それぞれのサーバー名は「パス・ホスト名」と呼ばれます。インターネットのホスト名＋ドメイン名によく似ていて，両者が一致するホストも多いのですが，ホスト名＋ドメイン名とは異なるパス・ホスト名を持っているサーバーもあります。

そういったサーバーは，UUCPで接続されているサーバーか，あるいは昔UUCP接続されていた経緯を持つサーバーであることが多く，UUCPのノード名をパス・ホスト名としているようです。

実を言えば，Path: フィールドの形式は，UUCPでメールやファイルなどを転送する際の指定方法＊16と同じです。例えば，**図6**のようにユーザーsengokuが投稿した記事が，サーバーfooへ届いたときのPath:フィールドが「foo!bar!gcd!sengoku」だったとすると，大昔のUUCPネットワークにおいては，サーバーfooであて先として「bar!gcd!sengoku」

```
Path: gcd.org!sengoku
From: Hiroaki Sengoku <sengoku@gcd.org>
Newsgroups: gcd.test
Subject: test
Date: 4 Aug 2001 14:13:55 +0900
Message-ID: <9kg0aj$2tn$1@asao.gcd.org>
Lines: 5

仙石です。これはテストです。


#7342.                                                    仙石 浩明
http://www.gcd.org/sengoku/        Hiroaki Sengoku <sengoku@gcd.org>
```

図4　ネット・ニューズ記事の例

```
Path: t-newsgw1.odn.ne.jp!newsfeed.mesh.ad.jp!news.ksw.feedmania.org!gcd.org!sengoku
```

図5　別のサーバーで読んだときのPath: フィールド

＊15　実を言えば，fjの流量は1999年ごろをピークに減少に転じています。現在の流量は10年前の水準を下回っています。Web掲示板やメーリング・リストなど，コミュニケーションの場がこのほかにもたくさんできてしまったので，fjの流量の減少は必然なのかも知れません。
＊16　メールを送る際の中継サーバーを「!」で区切って明示するこの宛先指定方法は「!」（「bang」と発音）形式と呼ばれ，いまだにこの形式のアドレス指定を受け付けるメール・サーバーも存在するようです。

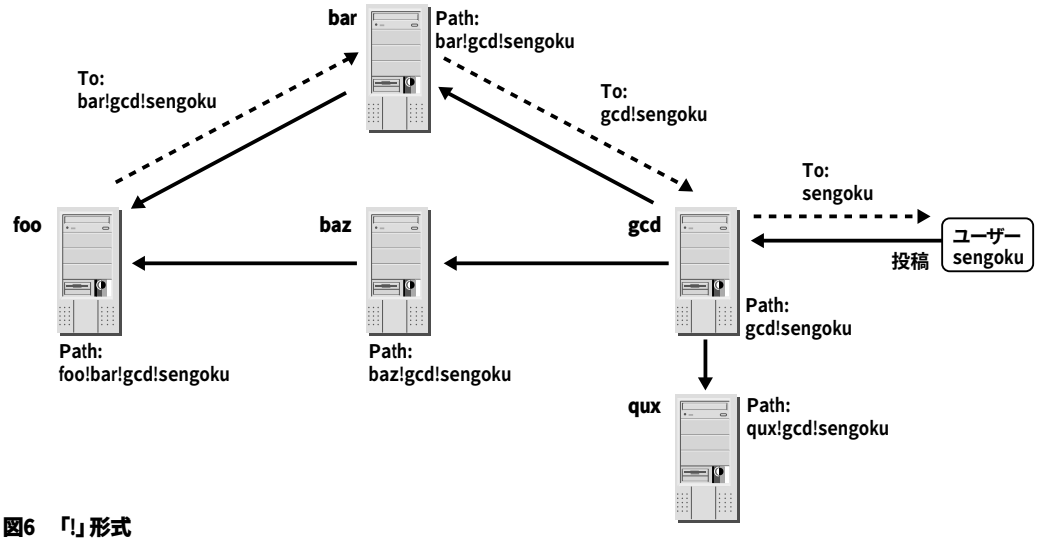

図6 「!」形式

を指定してメールを送信すると，bar, gcd経由でユーザーsengokuにメールを届けることができました。

Path: フィールドは単に配送経路を記録するだけでなく，ニューズ・サーバーがその他のニューズ・サーバーへ記事を配送するか否かを判断するのにも使われています。例えば，**図7**のようにgcdに投稿された記事が bar, baz経由でfooに届いたとします。この時，Path: フィールドは「foo!baz!bar!gcd!sengoku」となっていることでしょう。Path: フィールドに barが既に含まれているので，foo はこの記事がbarを経由して配送されたことを知ります。するとfooはこの記事をbarへは配送しません。

一般に，ニューズ・サーバーは，これから配送しようとする記事のPath: フィールドに，配送先のニューズ・サーバーのパス・ホスト名が含まれている場合は配送しません＊17。この仕掛けにより，同じ記事が同じサーバーを何度も訪れるような無駄を避けることができます。

Message-ID: フィールド

Message-ID:フィールドはメールにもありますが，メールではオプションなので無くても構わないのに対し，ニューズでは記事を検索する際のキーとしての役割を果たしています。つまりニューズ・サーバーは，同じMessage-IDを持つ記事は同一の記事であると見なして受け取らないのです。

例えば配送経路が図7のようになっているとき，サーバーgcdに投稿された記事はbar経由でbazへ届く場合と，gcdから直接bazへ届く場合とがあります＊18。bar経由の記事が先に届いた場合，同じMessage-IDを持つ記事がgcdから届くとbazは重複して届いた記事と見なして捨ててしまいます。

捨てられると分かっている記事を配送するのは無駄なので，どうすればこの無駄を回避できるかを考えます。

＊17　従って，もしネットワーク上に同じパス・ホスト名を持つニューズ・サーバーがあると，一方のサーバーを経由した記事はもう片方のサーバーには届かないことになり，記事の欠落が発生してしまいます。パス・ホスト名はインターネット上で唯一である必要があるわけですから，インターネット上で唯一であることが保証されているドメイン名を用いたパス・ホスト名を付けることが望ましいと言えるでしょう。

＊18　一見，bar経由の方が遠回りのように感じられるので，gcdから直接届く方が先になるのではないかと思われるかも知れませんが，barが広帯域の回線でインターネットに接続されていて，しかもサーバーが高性能だったりすると，ほとんどの記事がbar経由で先に届く，ということも起り得ます。さらに言えば，gcdからbar，foo経由でbazに届く記事もあるかもしれません。

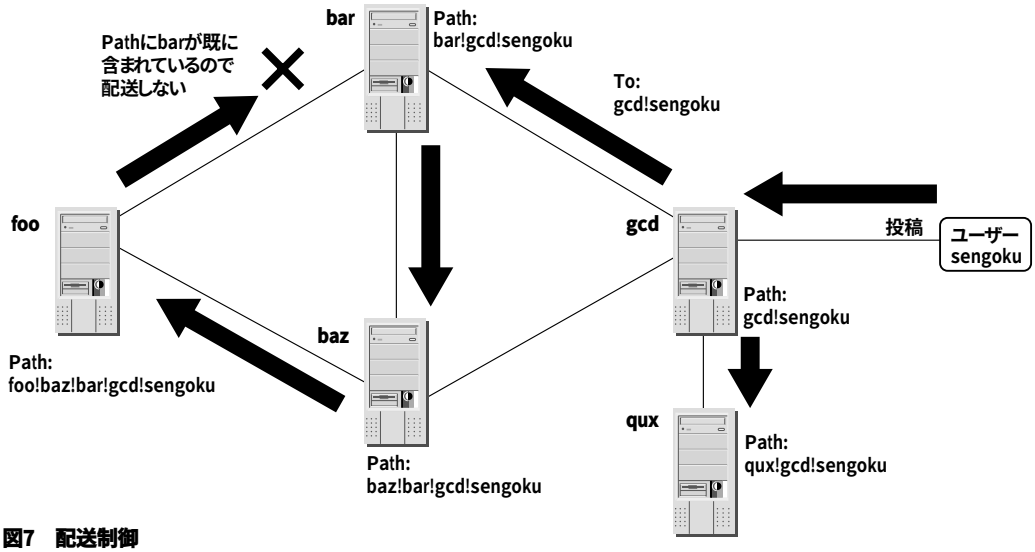

図7　配送制御

・Path: フィールドを使う方法

前述したように，Path: フィールドに配送先のパス・ホスト名が含まれていれば，配送せずに済みますが，この例の場合，Path: フィールドは「gcd!sengoku」で「bar」は含まれていません。従ってこの方法は使えません。

・冗長経路を無くす方法

gcdとbarが接続されていて，barとbazが接続されているのですから，gcdとbazが接続されていなかったとしても，gcdに届いた記事はbar経由で必ずbazに届きます。つまり，gcdとbazを直接結ぶ経路は冗長と言えます。この冗長経路を無くしてしまえば，同じ記事が同じサーバーへ異なる経路から届く無駄を避けることができるでしょう。

しかし，冗長経路を無くしてしまうと，障害に対して弱くなります。例えばgcdとbazを結ぶ経路が無かったとして，もしbarがトラブルで停止してしまったとすると，gcdからbazへの配送が不可能になってしまいます。安定して記事を流通させるには，少なくとも2本以上の配送経路を確保すべきでしょう。

・配送する前に，相手サーバーが記事を持っているか問い合わせる方法

配送相手が同じ記事を持っているか否か確認せずに配送しようとするから無駄が生じるわけで，それならまず確認して相手が持ってないことを確認してから送れば良い，ということになります。

gcdとbazがTCP/IPでつながっている場合は簡単で，TCP/IP上のニューズ配送プロトコルであるNNTP*19にはそのためのコマンドがあります。gcdはこれから配送しようとする記事それぞれについて，**図8**のようなIHAVEコマンドをbazに送信します。「IHAVE」とは，「I have ～」（私は～を持っている）の意味で，「～」の部分は Message-IDで指定します。

bazは，このMessage-IDで指定され

＊19　NNTPについて詳しくはRFC 977を参照してください。

```
IHAVE <9kg0aj$2tn$1@asao.gcd.org>
```

図8　IHAVEコマンド

```
435 article not wanted - do not send it
```

図9　IHAVEに対する応答「その記事は既に見たので不要です」

```
335 send article to be transferred.  End with <CR-LF>.<CR-LF>
```

図10　IHAVEに対する応答「そんな記事は見たこと無い，是非送ってくれ！」

```
Path: gcd.org!not-for-mail
From: news@gcd.org (news)
Newsgroups: to.takatsu.gcd.org
Subject: cmsg ihave gcd.org
Control: ihave gcd.org
Date: 4 Aug 2001 14:14:09 +0900
Message-ID: <9kg0b1$2uc$1@asao.gcd.org>
Lines: 6

<9kfvfu$429$1@nsvn01.zaq.ne.jp>
<9kfvgv$cp$1@nwall2.odn.ne.jp>
<9kg001$hsj$1@news512.nifty.com>
<9kg00b$4fc$1@nwall1.odn.ne.jp>
<9kg09l$cag$1@news511.nifty.com>
<9kg0aj$2tn$1@asao.gcd.org>
```

図11　IHAVEコントロール・メッセージ

る記事を自分が既に持っているかを検索し，もし持っていれば，**図9**のような応答＊20を返します。持っていなければ**図10**のような応答＊20を返すので，gcdはこれを受けて記事を送信します。

gcdとbazがUUCPで接続されている場合は少々面倒です。UUCPは基本的にはファイルやメール，記事などを送ることができるだけなので，IHAVEコマンドに相当することも「記事」として送信しなければなりません。

コントロール・メッセージ

人間が読むための「記事」ではなく，ニューズ・サーバーが読む記事を「コントロール・メッセージ」と呼びます。実際に使われている＊21コントロール・メッセージは，次の6種類です。

(1)ニューズ・グループの作成
(2)ニューズ・グループの削除
(3)ニューズ・グループに過不足が無いかを確認
(4)記事のキャンセル
(5)IHAVE
(6)SENDME

(5)のIHAVEと(6)のSENDMEが，前述のIHAVEコマンドと同等のことをUUCP経由で行うためのコントロール・メッセージです。(1)～(4)については次号で解説します。

図11にIHAVEコントロール・メッセージの例を示します。ニューズ・サーバーgcd.orgから，takatsu.gcd.orgに対して，gcd.orgが現在持っている記事のMessage-IDのリストを知らせるためのIHAVEコントロール・メッセージです。

「Control: 」フィールドがあるほかは，通常の記事と同じです。IHAVEコントロール・メッセージの場合，Control: フィールドに「ihave 自分のパス・ホスト名」を指定し，さらにNewsgroups: フィールドに「to.相手のパス・ホスト名」を指定します。「Subject: 」フィールドが「cmsg」に続いてControl: フィールドと同じ内容が入っていますが，これはネット・ニューズ

＊20　もちろん，最初の数字のみが意味を持ちます。数字に続く文字列は（デバッグ目的など）人間が見たときの便宜を図っているだけで，何が書いてあっても無視されます。
＊21　このほかにSENDSYS，SENDUUNAME，VERSIONがありますが，ネット・ニューズの初期を除けば日常的に使われることはまず無いのではないでしょうか。

の初期にはControl: フィールドがなく，Subject: フィールドで代用していたころの名残りです。

本文は，ニューズ・サーバーgcd.orgがこれからtakatsu.gcd.orgに送ろうとしている記事のMessage-IDを並べたものです。takatsu.gcd.orgがこのIHAVEコントロール・メッセージを受け取ると，本文に並んでいるMessage-IDの中から，takatsu.gcd.orgが持っていない記事を抜き出してSENDMEコントロール・メッセージを作成し，gcd.orgへ送り返します。SENDMEとは，「send me」（私に送って）の意味で，送ってほしい記事のMessage-IDを本文に並べたコントロール・メッセージです*22。SENDMEコントロール・メッセージの一例を**図12**に示します。

```
Path: takatsu.gcd.org!not-for-mail
From: news@takatsu.gcd.org (news)
Newsgroups: to.gcd.org
Subject: cmsg sendme takatsu.gcd.org
Control: sendme takatsu.gcd.org
Date: 4 Aug 2001 14:14:24 +0900
Message-ID: <9kg0bg$2um$1@takatsu.gcd.org>
Lines: 2

<9kg001$hsj$1@news512.nifty.com>
<9kg0aj$2tn$1@asao.gcd.org>
```

図12　SENDMEコントロール・メッセージ

SENDMEコントロール・メッセージは，IHAVEコントロール・メッセージの「ihave」の部分が「sendme」になっているほかは，IHAVEコントロール・メッセージと同じ形式です。Control: フィールドのsendmeに続けて自分のパス・ホスト名を指定し，Newsgroup: フィールドの「to.」に続けて相手のパス・ホスト名を指定します。

通常，IHAVEコントロール・メッセージは，数10分～数時間ごとに送信するので，その時間間隔を*n*分とし，IHAVEコントロール・メッセージの作成時刻を*t*とすれば，時刻*t*-*n*から時刻*t*までにgcd.orgに届いた記事の中で，takatsu.gcd.orgに送るべき記事が対象となります。

ただし，takatsu.gcd.orgは，どのニューズ・サーバーから主に記事を受け取るのかを決める必要があります。もし，どこを主にするか決めず，接続している全サーバーから同じようにIHAVEコントロール・メッセージを受け取ると，その時点では記事をまだ受け取っていないので，すべてのサーバーに対して同じ記事を要求するSENDMEコントロール・メッセージを送信してしまい，結果として同じ記事を何度も受け取ることになってしまいます。

ここではsengoku.orgから主に記事を受け取ることにしたとします。この場合，sengoku.orgからはIHAVEコントロール・メッセージではなく，最初からすべての記事を送ってもらうよう設定します。そして，その他のサーバーからはIHAVEコントロール・メッセージを送ってもらって，sengoku.orgから送ってもらえなかった記事を，他のサーバーから送ってもらって補うようにすると良いでしょう。

その他のサーバーとして，ここではgcd.orgからIHAVEコントロール・メッセージを*n*分間隔で送ってもらうことにしたとします。そしてgcd.orgは，時刻*t*-*n*から時刻*t*までにgcd.orgに届いた記事を対象として，IHAVEコントロール・メッセージを時刻tに送信する設定にしたとします。

図13に示すように，ある記事Aが，

*22　必ずしもIHAVEコントロール・メッセージに対する返答として送る必要はなく，相手が持ってそうな記事を送ってもらいたい時にも使うことができます。ただし，特定のサーバー以外からのSENDMEコントロール・メッセージは無視するサーバーがほとんどなので，あらかじめSENDMEコントロール・メッセージを受け付ける設定にしておいてもらう必要があるでしょう。

gcd.orgに時刻*t-a*に届き，sengoku.orgには時刻$t + a'$に届いたとすると，takatsu.gcd.orgがIHAVEコントロール・メッセージを受け取った時刻*t*の時点では，まだ記事Aはtakatsu.gcd.orgには届いていませんから，記事Aを求めるSENDMEコントロール・メッセージをgcd.orgに送ってしまいます。

その結果，gcd.orgは記事Aをtakatsu.gcd.orgへ送りますし，記事Aがsengoku.orgに届き次第，sengoku.orgも記事Aをtakatsu.gcd.orgへ送ることになってしまいます。

この無駄を避けるには，gcd.orgは，時刻*t-n*から時刻*t*までにgcd.orgに届いた記事を対象としたIHAVEコントロール・メッセージを，時刻*t*に送るのではなく，sengoku.orgから記事Aから送られてこないことがほぼ確実であることが判明する時刻*t+d*に送るようにすると良いでしょう。例えば*d*として1日を設定します。**図14**に示すように，時刻*t+d*までに記事Aがsengoku.org経由でtakatsu.gcd.orgに届けば，記事Aを求めるSENDMEコントロール・メッセージをgcd.orgへ送らずに済みます。

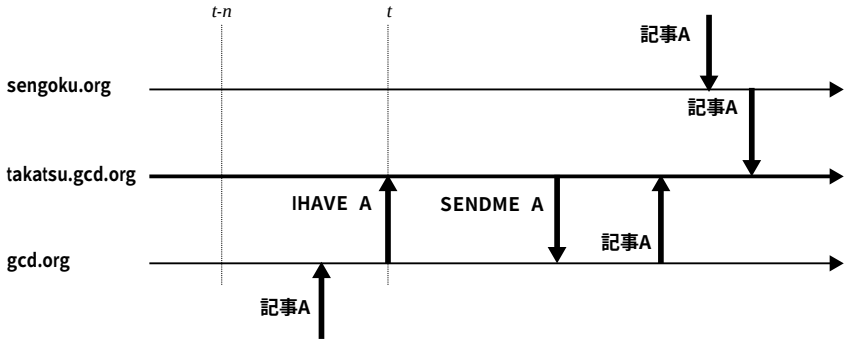

図13　IHAVEコントロール・メッセージの送信タイミング

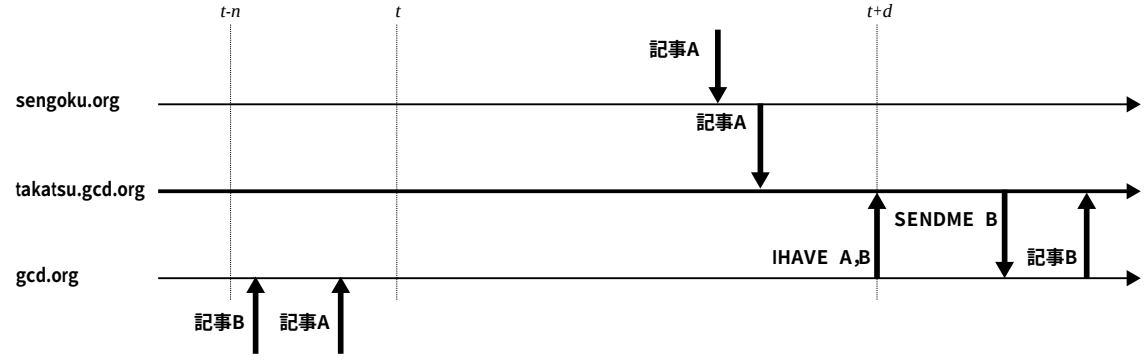

図14　遅延付きIHAVEコントロール・メッセージ