4．ネットワーク技術

電子ニュースシステムにおける記事保有期間の管理

総合情報処理センター　鶴　正人

1．電子ニュースシステムとその管理

近年の計算機ネットワークの進歩・普及により，日常的に，例えば研究室や自宅の個人の計算機からでも，いろいろなネットワークサービスを直接利用することが可能になりつつある．しかし，残念ながら現時点では，ネットワークのサービスは，買ってきた計算機の電源を入れるだけですぐ利用できるというものではなく，インストールや環境設定がかなり面倒なだけでなく，その後の維持管理も発生するのが常である（分散処理の宿命かもしれないが）．

それらの内でも手がかかるものの一つが，電子ニュース（のサーバ）である．ニュースサーバと呼ばれるシステムの働きを簡単に言ってしまうと，そのシステム上から投稿された，あるいは他システムから転送された記事を受付け，蓄積し，必要なら他のシステムへ転送し，さらに，古くなったものは消去する，ということになる．

ここで，記事は，ニュースプールと呼ばれる場所（以下 /usr/spool/newsというディレクトリとする）下のファイルとして蓄積される．

例えば，nagasaki-u.forumというニュースグループの番号23の記事は，

/usr/spool/news/nagasaki-u/forum/23

というファイルに格納される．

このように，ニュースサーバにおいては，世界中で投稿された記事が，ネットワークを通して流れ着くので，多い日には，5000通以上，15メガバイト以上を受信する．そして，それを保有しておくための計算機のディスクスペース（容量＝空きブロック数，または，ファイル数＝空きi-node数）には限りがあるので，古い記事から消していくわけだが，一方，ユーザにとっては，後で読み返せるためにも，できるだけ記事は長く置いていて欲しい．

そこで，記事の受信量の変動を考慮に入れ，ニュースプールの溢れを防ぎ，かつ必要度に応じて最大限の記事保有期間を確保することが求められてくる．

というわけで，本稿では，電子ニュースサーバおける記事の消去（保有期間の調整）の問題を紹介する．

なお，ニュースサーバの維持管理には，他に，エラーの監視，ニュースグループの管理，転送経路や冗長転送の管理等があり，それなりの労力を要する．

2．簡単なモデル

まず，話を単純化する．ニュースサーバに対して，平均速度λで記事が流入（到着）してきて，ニューススプールに格納される．そして，あるタイミングである程度古い（到着してからの時間で計る）記事から消去(expire)する．

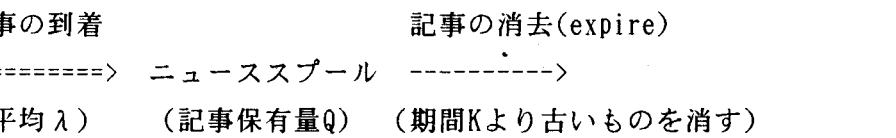
このとき，

% ニューススプールを溢れさせない（記事保有量がある値を超えない），

% できるだけ記事は長く保有したい，

% できるだけ標準のツールを使い，簡単（軽い）な仕組みで実現したい，

を満たすような expire の方針を決めたい．



ここで，流入速度λとは，ニューススプールの空き容量が問題になるならば，単位時間内の到着記事の総容量（バイト数）であり，空きi-node数（ファイル数）が問題ならば，単位時間内の到着記事個数である．

なお，厳密な議論には，記事の到着の確率的振る舞い（分布）を仮定する必要があるので，ここでは大雑把な議論だけを行う．

そこで，以下，[A] ～ [D]の４通りの expire方法を比較，検討する．

なお，実際の記事の保有期間は，ニュースグループの重要度に応じて差別化するのが普通であるが，ここでは簡単のために，そのような差別化は考えない．

Tn，Qn，qnを以下のように決める．

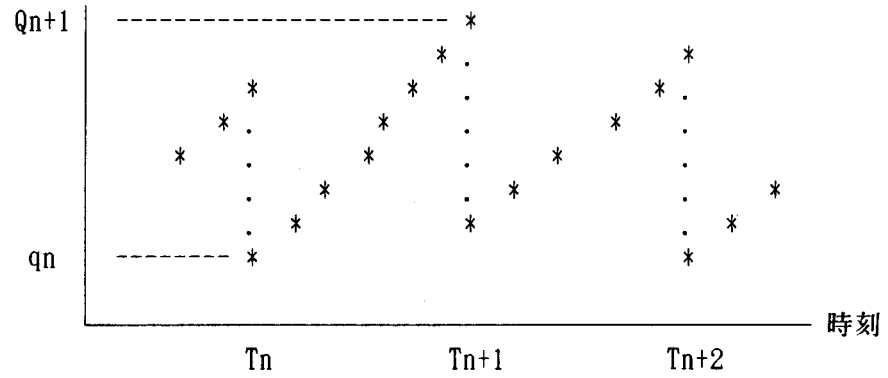
% n番目の expireの起動時刻をTn

% n番目の expireの起動直前の記事保有量（総バイト数または総個数）をQn

% n番目の expireの起動直後の記事保有量をqn

[A] 一定期間d毎に expireし，一定期間Kより古い記事を消去する．

記事保有量



% Tn+1 = Tn + d

% qn = "期間(Tn - K, Tn]の間の到着記事量"

% Qn+1 = "期間(Tn - K, Tn+1]の間の到着記事量"

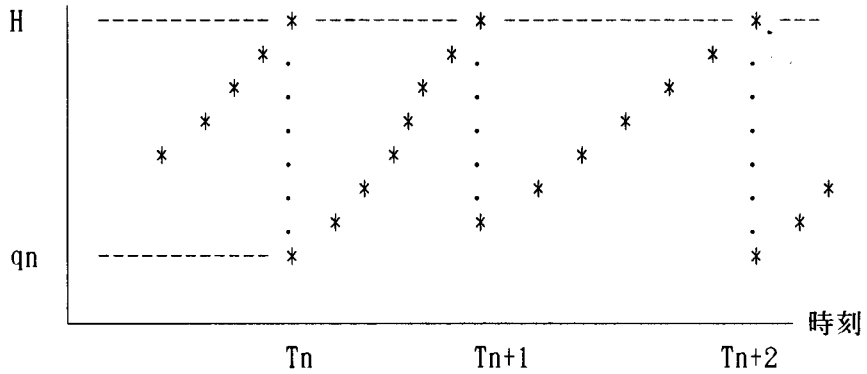
なので，

1)保有量の極大値は，平均～ λ(K + d)

2)保有期間は，　　最小＝ K　　平均～ K + d/2

[B] 記事保有量が一定値Hを超えたら，expireし，一定期間Kより古い記事を消去する．（実際には，短か目の一定周期で保有量をcheckする）

記事保有量



ただし，λK < H とする．

% qn = "期間(Tn - K, Tn]の間の到着記事量"

% Qn+1 = "期間(Tn - K, Tn+1]の間の到着記事量" = H

(∴ λ(K + Tn+1 - Tn) ～ H)

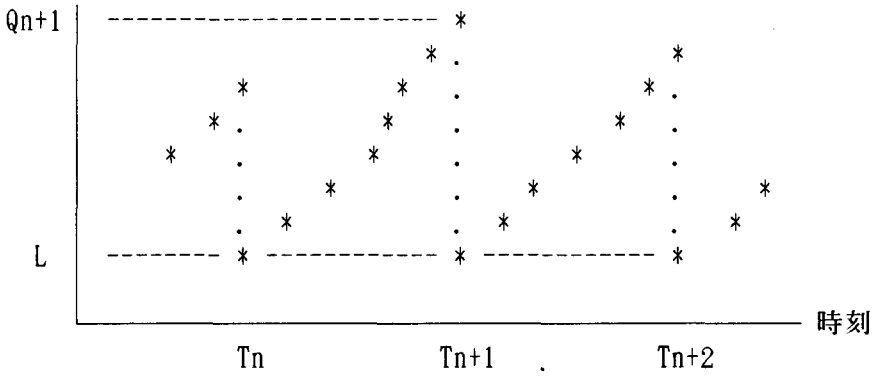
なので，

1)保有量の極大値は，常に H

2)保有期間は，　　最小＝ K　　平均～ K + (H/λ - K)/2

[C] 一定期間d毎に expireし，消去後の記事保有量が一定値Lになるまで，古い記事から順に消去する．

記事保有量



ここで，Tnで，Tn-Kn より古い記事を消去したとすると，

% Tn+1 = Tn + d

% qn = "期間(Tn - Kn, Tn]の間の到着記事量" = L　(∴ λKn ～ L)

% Qn+1 = "期間(Tn - Kn, Tn+1]の間の到着記事量"

なので，

1)保有量の極大値は，平均～ L + λd

2)保有期間は，　　平均～ L/λ + d/2

ここで，[A]-[C]を比較すると，

(1) 保有期間の最低保証値がはっきりしている点では，[A]，[B]が[C]より望ましい．

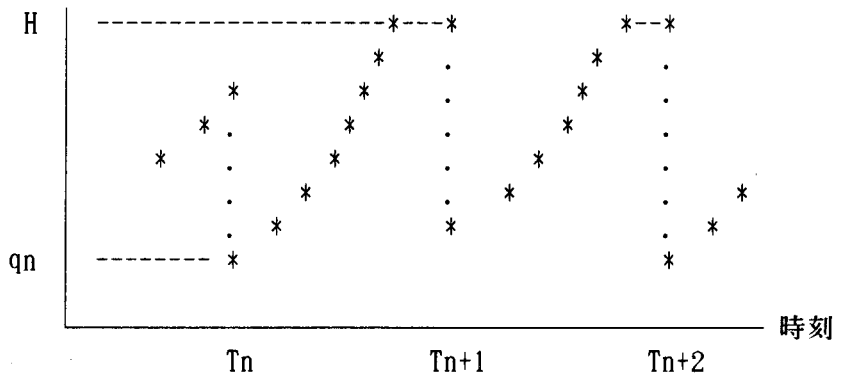
(2) 保有量の上限に関して，[B]は固定だし（実際にはチェック周期分の検出遅れはある），[C]も制御しやすいが，[A]の場合，このままでは流入速度の変動の影響をもろに受け，危険度が高い．

(3) 運用面から見て，[B]はexpireの起動時期が不確定で，他の処理も含めた管理がやりにくい．また，[C]は，消去後の保有量が一定になるような保有期間の算出を動的に行うオーバヘッドがかかる．

そこで，次の[D]のように，[A]の欠点を補うために，受信フロー制御の併用を行うのが普通である．

[D] 一定期間d毎にexpireし，一定期間Kより古い記事を消去する．
ただし，記事保有量が一定値Hを超えたら，受信を一時的に拒否する．

記事保有量



ただし，λK < H とする．

% Tn+1 = Tn + d

% qn = "期間(Tn - K, Tn]の間の到着記事量"

% Qn+1 = "期間(Tn - K, Tn+1]の間の到着記事量" <= H

なので，

1)保有量の極大値は，最大＝ H

2)保有期間は，　　最小＝ K　　平均～ K + d/2

となって，一応はうまくいく．

しかし，受信を拒否しても，大元の記事の発生が止まるわけではないので，単に途中に滞留しているだけで，受信再開時には，その分どっと到着する．
また，この滞留が多くなると，重要な記事が延着したり，途中で記事が欠落したり，そして，実際には滞留（中継）地点でのexpireも起こるから，記事が来る前に消される可能性も出てくる．よって，この滞留の様子を確認する必要がある．

記事の発生		記事の到着		記事の消去
==========>	外部の	==========>	ニュース	---------->
μn	中継点	λn	スプール	
	(滞留量Rn)		(保有量Qn)	

また話を単純化して，以下では，

% n番目のexpireの起動時刻をTn(Tn+1 = Tn + d)

% 期間[Tn-1, Tn)における，記事の発生量をμn ，

% 期間[Tn-1, Tn)における，記事の到着量をλn ，

% 時刻Tnでの記事滞留量をRn，

% 時刻Tn（のexpire直後）での記事保有量をQn，

とし，個々の記事の転送には全く遅延がないものとする．

さらに，保有期間Kは，expire間隔dの倍数（k倍），つまり，k * dとする．

この時，H，kを決め，隣接するexpireの間の記事の発生量 {μn} (n=1,2,...)を与えると，Rn，Qn の変化は以下のように近似できる．

```
if( Rn-1 + μn <= H - Qn-1 ) {
  λn = Rn-1 + μn ;
  Rn = 0 ;
}
else {
  λn = H - Qn-1 ;
  Rn = Rn-1 + μn - λn ;
}
Qn = Qn-1 - λn-k + λn ;
```

解析的に考えずに，安易にシミュレートしてみると，次のような例が得られた．

% H = 135,000 (Kバイト)

% 一時間の記事到着量 {In} (Kバイト)

% d = 12, 24 (時間)

% K = 10*24, 12*24, 14*24 (時間) - つまり，10〜14日

を与え，

$$\% \ \mu n = \sum_{d*(n-1)+1}^{d*n} Ii \qquad \% \ k = K / d$$

から，約58日間分シミュレートした．なお，与えた {In} は，本学での実際の受信データから生成したものである．平均 408 (Kバイト)，標準偏差 284.9.

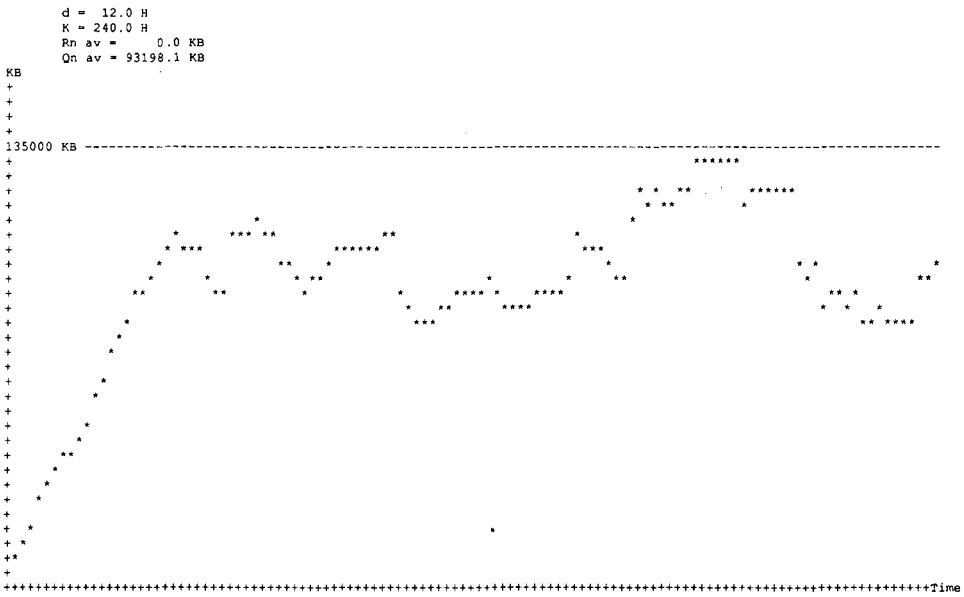
以下のグラフでは，

* 印がQn（expire直前の保有記事容量）の時間変化，

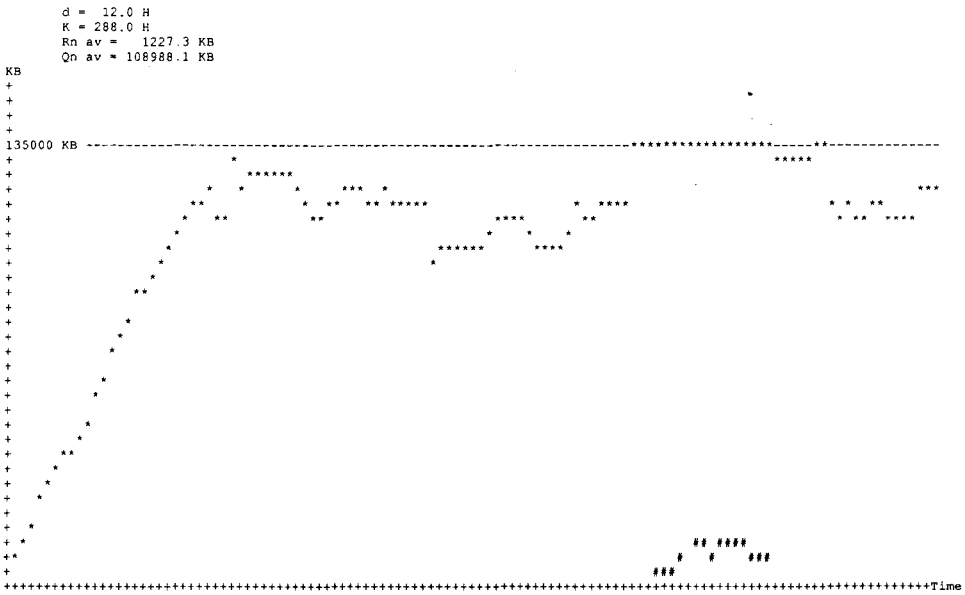
印がRn（外部滞留記事容量）の時間変化，である．

〈 図2.1 保有記事量及び滞留記事量のシミュレーション 〉

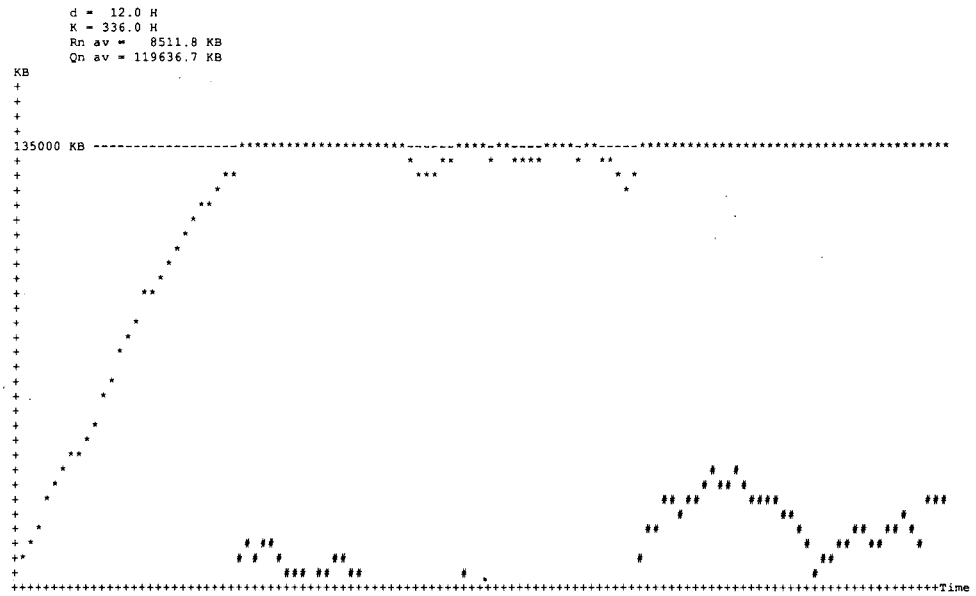
(1) expire間隔(d) = 12時間, 保有期間(K) = 10日



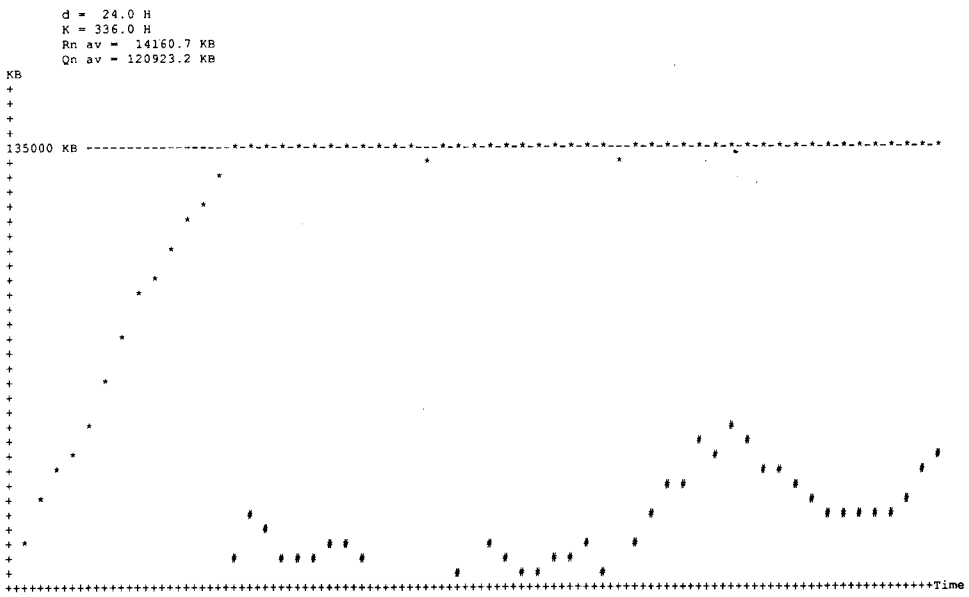
(2) expire間隔(d) = 12時間, 保有期間(K) = 12日



(3) expire間隔(d) = 12時間, 保有期間(K) = 14日



(4) expire間隔(d) = 24時間, 保有期間(K) = 14日



2.で考えた，保有量の極大値の平均～ λ(K + d) というような大雑把な推論からは，(K + d) < H/λ ～ 135000/408 = 330時間 = 13日となって，上の(3)か，または(4)あたりが，外部滞留が定常的には発生しない限界だろうと予想されるが，それはだいたい当たっている．

しかし，(3)と(4)の比較から判るように，K=14日の場合，expire間隔d が12時間と24時間とで，滞留量がかなり違う（平均量でも倍近くなる）．

3．長崎大学における実際

現在，総合情報処理センター（以下，センターと呼ぶ）の計算機（sieboldという名のUNIXワークステーション）を大元のニュースサーバとして，そこで学外からの記事を受信し，それを学内の他のニュースサーバ計算機に転送している．

そして，そのsieboldでは，2.の[D]に相当する方法で，expireを運用している．

しかし，2.ほど単純ではなく，例えば，以下のような点を考慮する必要がある．

(1) 現在，本学で講読しているニュースグループは，700近くあるが，記事保有期間に関して，よく読まれる（または重要な）グループは長く保有したい．

よって，ニュースグループ（群）を意識したexpireの期間設定が必要になる．

(2) 到着記事量は，かなり変動が大きく，かつ，分布自体が変化するようである．

これは，ネットワーク自体が日々変化していくし，また，ニュースグループも，新設，改廃，内容の変化等が起こるからであろう．単純に，平均速度λのポアソン到着，などと仮定した計算はできないかも知れない．

(3) (2)の変動の別の理由に，ネットワークの停止や上流サーバの停止に伴って，一日以上，ニュース転送がストップするケースがある．

この場合，転送が再開された時点で，怒濤のような速度で記事が到着する．

ただし，（統計的に調べたわけではないが）経験上の感覚からいうと，受信処理またはネットワークの能力（上流サーバとは，48Kbpsの学術情報網でつながっている）からか，最大でも，30通/分ぐらいのようである．

そこで，以下に，sieboldでの状況を紹介するが，実際のexpire関連のソフトウェアの設定方法も簡単に触れる．なお，UNIXマシンなので，UNIX用語は断りなしに使う．

一般に電子ニュースシステムには，bnewsとcnewsの2種類が普及しているが，世界的に，bnewsからcnewsへの移行が進んでおり，sieboldでもcnewsを使っている．つまり，以下に出てくる具体的設定方法はcnewsでの話である．

[1] ニューススプールの環境

/var/spool/newsというディレクトリをニューススプールとし，単独に1つのパーティションを割り当てている．

ただし，ニューススプールの中に，out.goingというディレクトリがあるが，そこだけは，symbolic-linkを用いて，他のパーティションに追い出している．

なお，ニューススプールとニュース管理用ディレクトリ（通常，/usr/lib/news）とは，通常並行してアクセスされるので，物理的なIOの待ちを減らすため，別ディスクに置いている．

```
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/sd0g             151399  133754    2505    98%    /var/spool/news
Filesystem             iused   ifree  %iused  Mounted on
/dev/sd0g              30757   45851    40%   /var/spool/news
```

つまり，実効容量：約136MB，実効i-node数：約76,000個であり，ご覧にように，容量に関して，かなり自転車操業である．

一般にはニューススプールを溢れさせないためには，容量とi-node数の両方の制約を満たす必要がある．しかし，sieboldの場合は，容量に比べてi-node数が十分大きい（保有記事の平均サイズを小さめにみて2Kbytesとしても）ので，容量の制約だけを考えておけばよかった．

[2] expireの起動

expireというコマンドを，6時間毎に日に4回も起動している．決して，お勧めではないが，余裕のない容量で運用する場合，expire間隔を小さくするのも止むを得ないということは，2.の[D]のシミュレーションからもわかると思う．

具体的には，安全にexpireを起動するためのシェルスクリプトdoexpireを，cron機能で，定時に起動する．

< 図3.1 crontab >

/var/spool/cron/crontabs/root:

```
.....
50 4,10,16,22 * * * su news -c '/usr/lib/newsbin/expire/doexpire -v'
07 13         * * * su news -c '/usr/lib/newsbin/maint/newsdaily'
.....
```

expireは，一～数分かかる処理であり（bnewsのexpireに比べると格段に速くなったが..），その途中で，何かの理由で中止させられると，ニュースの管理情報ファイル（history,active）の一貫性がなくなったり，最悪は壊れたりする．つま

り，あまり頻繁にexpireするのは，それだけ危険を伴う．

誰かが読んでいる途中の記事をexpireしたらどうなるか？ これは，news reader側の処理しだいであるが，あまり気にする必要はない．

しかし，二つのexpireがぶつかるのは危険である．doexpireには，これに対する予防が組み込まれている．

[3] 保有期間の設定(explist)

どのニュースグループの記事は何日経過したら消去する，という情報は，通常，explist というファイルに記述しておく．（図3.2）

起動されたexpireは，このファイルに基づき，historyの情報と比較しながら，各記事に対して，消すか消さないかを決定する．

sieboldでは，ニューススプールの余裕がないため，

(1) 本学において必ずしも必要でないと思われるものは，すぐ消す．

(2) 重要なものは長く残す．

また，当面は，学内ニュースグループは消さない．

(3) 一時的に(?)，非常に活発なニュースグループがスプールを圧迫し始めたら，しばらく，そのグループは保有を短くする．

(2) との兼ね合いが難しいが...

という方針に基づき，細かすぎるexplistを設定している．

(1)，(2)に関しては，理想は学内利用者のニュースグループ利用頻度の統計を取り，それも考慮すべきであるが，いまは一般的判断でやっている．

(3)に関しては，受信一時停止がよく起こる状況になった時点で，人手で調整している．安易な自動化は危険と思われる．

< 図3.2 あるexplist の一部 >

/usr/lib/news/explist:

```
/expired/                       x       28      -
/bounds/                        x       0-1-150 -
# Special NGs and categories:
control                         x       1       -
all.test,all.all.test           x       1       -
fj.general                      x       9999    -
all.general                     x       14      -
local                           x       7       -
.....
# Special groups: too active or noisy now
comp.sys.amiga                  x       1.5     -
```

```
comp.mail.maps                  x       6       -
bionet.molbio.genbank           x       1.5     -
fj.jokes                        x       1       -
fj.rec                          x       2       -
news.groups                     x       2       -
.....
# comp, gnu, bionet, news
comp.bugs,comp.doc              x       28      -
comp.binaries                   x       14      -
comp                            x       8       -
.....
# fj, jp
fj.guide,fj.lectures            x       9999    -
fj.announce                     x       9999    -
fj.questions                    x       49      -
fj                              x       28      -
jp                              x       9999    -
.....
```

書き方を簡単に説明すると，

(1) 初めの二つは，

/expired/は，記事そのものが消去された後もhistoryにその履歴を残すために，historyから消されない保証期間を指定する．記事自体の保有期間が短い場合に，一度到着し，その後消去した記事が，何かの理由で再度到着した場合に対する防御である．

/bounded/は，Expiresヘッダで記事自体がexpire期日を指定している場合に，それを許す限度のデフォルト値を指定する．各ニュースグループ毎に，下の行で変更することも可能である．

(2) # で始まる行はコメント．

(3) そして，各行にニュースグループ（群）に対する，保有期間の指定を書く．

あるニュースグループが複数の行にマッチしても，最初の行のみが有効であるので，行の順番は重要である．期間には，10進整数か小数（atof関数が理解できる形）が書ける．例えば，

```
control         x       1       -
```

と書くと，ニュースグループ名がcontrolで始まる全てのグループの記事を，到着後，一日経ったら消去する．

xの所は，他に，m(moderate groups only)，u(unmoderate groups only)のような指定ができる．- の所は，必要なら消去時に別の場所にアーカイブ（退避）することを指定できる．

詳しくは，cnewsのソースに含まれる，"expire"オンラインマニュアル参照.

[4] その他の注意点

思いつくままに上げると，

(1) そもそも自分の意図した通りの保有期間で動いていることの確認が必要である. explistの記述のちょっとした間違いは，見ただけでは気づかない.

(2) ニュースサーバの状況の把握において，記事保有量は任意の時点で静的に調べられるが，記事到着量はニュースシステムと連動して，日々情報を取っておく必がある. 基本的には，logファイル(/usr/lib/news/log)から必要な情報を収集する. log は，一日一回の管理用シェルスクリプトnewsdaily（図3.1参照）でローテートされるので，そこに収集処理を組み込めばよい. expire前後で取れた情報は，doexpireの中で取ればよい. また，NNTP手順を用いて他サーバから転送された記事の個数は，nntpdというデーモンプロセスのログ(/usr/lib/news/nntplog) からもわかる.

(3) 記事を消さないニュースグループに注意する. 塵も積もって山になっている場合がある.

(4) 複数のニュースグループにクロスポストされた記事は，実体（i-nodeも含め）は一つで，それらのニュースグループのうちの最も長い保有期間後に，消去される.

(5) 記事のヘッダの中に，

Expires: 10 Feb 1992 18:19:00 GMT

のような記述があると，通常はそっちが優先され，思いどおり消去されない.

(6) expireがcoreを吐いて異常終了したり，ニュース受信やexpire中にシステムダウンしたりして，historyやactiveといった管理ファイルがおかしくなった疑いある場合，それらの再構成が必要である. このために，mkhistory，recovact といったコマンドがある.

詳しくは，cnewsのソースに含まれる，"expire"オンラインマニュアル参照.

[5] 実際の記事の到着と保有

sieboldには，平均して，一日約3200通，約10Mバイトの記事（平均サイズが約3Kバイト）が到着している.

また，ニュースグループ名の第一要素別の一日平均到着記事数（100通以上）は，一般に，

comp　約2600通
fj　約 350通
news　約 100通
bionet 約 100通

のようになり，comp系が，7〜8割りを占める．

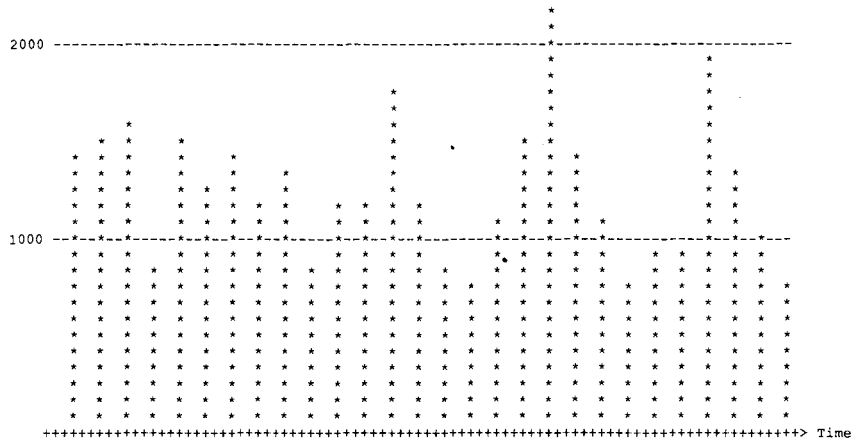
以下に，ある7日間の記事の到着の例を示す（図3.3，3.4）．この時は，6時間毎に調べた到着速度は，平均で，

記事個数／時間＝ 211.1

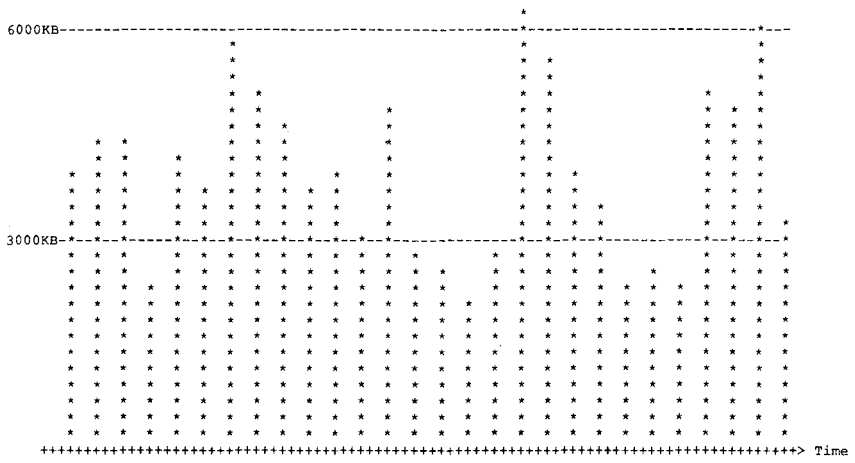
記事容量(KBytes)／時間＝ 660.2

であった．

〈 図3.3 6時間毎の到着記事個数の変動 〉



〈 図3.4 6時間毎の到着記事容量の変動 〉



その結果，保有記事量とその内訳は，表3.1，3.2 のようになった．なお，到着記事の平均サイズより保有記事の平均サイズの方が，一般に大きい．

< 表3.1 ニュースグループ名の先頭要素別保有記事の個数と容量 >

	個数	容量[bytes]	平均サイズ[bytes]
bionet	499(1.5)	2222718(1.7)	4454
comp	24186(73.2)	94402248(72.3)	3903
control	35(0.1)	172958(0.1)	4942
fj	4818(14.6)	21159274(16.2)	4392
gnu	1762(5.3)	4328906(3.3)	2457
kyushu	31(0.1)	42782(0.0)	1380
kyushu-u	34(0.1)	221415(0.1)	6512
nagasaki-u	348(1.1)	1530579(1.2)	4398
news	1326(4.0)	6546959(5.0)	4937
total	33039(100.)	130627839(100.)	3954

< 表3.2 comp下のニュースグループの第二要素別保有記事の個数と容量 >

	個数	容量[bytes]	平均サイズ[bytes]
容量順(トップ10):			
comp.sources	1035	36609478	35372
comp.sys	8462	14976661	1770
comp.mail	656	6929078	10563
comp.os	2591	5814025	2244
comp.binaries	425	5518109	12984
comp.lang	2045	4810955	2353
comp.unix	1945	4080436	2098
comp.windows	1698	3312371	1951
comp.dcom	857	1646384	1921
comp.ai	451	1341457	2974
個数順(トップ10):			
comp.sys	8462	14976661	1770
comp.os	2591	5814025	2244
comp.lang	2045	4810955	2353
comp.unix	1945	4080436	2098
comp.windows	1698	3312371	1951
comp.sources	1035	36609478	35371
comp.dcom	857	1646384	1921
comp.mail	656	6929078	10563
comp.text	555	1043199	1880
comp.protocols	541	1180442	2182

また，細かいニュースグループ別の活発度をみると，個数の上で到着量が多いの

は，表3.3 のようになり，個数の多いグループは記事サイズが小さめである．

< 表3.3 ある4週間のニュースグループ別到着記事個数のトップ15 >

	1st	2nd	3rd	4th	total	num	quan	size
comp.sys.amiga	1668	2760	2033	1903	8364	369	669.0	1813
comp.sys.mac	1430	1834	1672	1549	6485	1795	3217.5	1792
comp.sys.next	574	572	917	708	2771	718	1330.1	1852
comp.sys.ibm	451	713	665	683	2512	127	216.1	1701
bionet.molbio.genbank	418	281	666	963	2328	147	598.6	4072
comp.windows.ms	479	671	524	449	2123	209	375.0	1794
comp.sys.atari	312	624	520	434	1890	75	245.4	3271
comp.os.os2	325	581	427	399	1732	184	413.3	2246
comp.sys.sun	256	496	502	351	1605	547	970.3	1773
comp.windows.x	320	443	445	390	1598	579	1356.3	2342
comp.sys.apple2	300	537	343	371	1551	226	334.5	1480
comp.os.msdos	283	396	318	404	1401	599	1299.5	2169
news.groups	277	314	302	367	1260	283	635.1	2244
fj.mail-lists.x-window	327	373	263	273	1236	204	379.7	1861
comp.binaries.ibm	323	377	234	229	1163	228	821.3	3602

1st ～ 4th: 第一～四週目の各到着個数　total: 4週間の合計数

num, quan, size: 四週目が終わった時点での保有記事数(個)，保有記事容量(Kbytes)，保有記事の平均サイズ(bytes)

ただし，ニュースグループは第3要素以下をまとめて考えている．

例えば，comp.sys.amiga は，

comp.sys.amiga.marketplace

comp.sys.amiga.games

.....

等の15個のニュースグループをまとめた全体である．

4．まとめ

以上，電子ニュースサーバでの記事の保有期間の調整(expier)問題の概要と，本学での実際の例を，簡単に述べた．

既に，電気情報工学科でもニュースサーバを運用しており，今後，各学部や学科単位でニュースサーバを立ち上げ，運用を行うケースが増えてくると思われるが，そのような場合に，本稿が多少なりとも参考になれば幸いである．

基本的には，あまり細かい調整をやるよりも，ニュースプールに使うディスクを増設するなどして容量を増やすことの方がずっと合理的な解であり，センターでもできれば（予算が許せば:-）そのように対処していきたい．

ただし，ディスク容量を増やすにしても，今後は，もう少し突っ込んだ確率的解析に基づいて実施していきたい．

なお，自組織でニュースサーバ(cnews) を立ちあげてみようと計画している方は，センターに御連絡（相談）下されば，センターで作成した資料（インストール／運用の手引き，学内ルール）を送付します．この件に関する連絡は，メールアドレス

f1234@cc.nagasaki-u.ac.jp （UTS上からは，f1234のみでよい）

宛に電子メールでお願いします．