# RSS 서비스를 위한 최소 누락 수집 정책
## (A Minimum Missing Aggregation Policy for RSS Services)

한 영 근 † 이 상 호 ††
(Young Geun Han) (Sang Ho Lee)

**요 약** RSS는 웹 콘텐츠 배급을 위한 XML기반 포맷으로, 사용자는 RSS 피드 수집기를 통해 RSS 피드를 수집한다. RSS 피드를 효과적으로 수집하기 위해서는 RSS 피드에 대한 수집 정책이 필요하다. 본 논문은 RSS 피드 수집 시에 누락되는 포스팅을 최소화하기 위한 RSS 피드 수집 정책을 제안하고, 실험을 통해 제안한 수집 정책과 기존 수집 정책을 비교 분석하였다. 본 논문에서 제안한 수집 정책은 기존 수집 정책과 비교하여 6%의 수집 지연 증가로 23%의 수집 누락이 감소됨을 실험을 통하여 알 수 있었다.
**키워드** : RSS 피드, RSS 피드 수집기, 수집 정책

***Abstract*** RSS is the XML-based format for the syndication of web contents, and users aggregate RSS feeds with RSS feed aggregators. In order to effectively aggregate RSS feeds, an RSS aggregation policy is necessary. In this paper, we first propose an aggregation policy to minimize the number of postings being missed within an aggregation. Second, we analyze and compare our aggregation policy with existing aggregation policies. Our analysis reveals that our aggregation policy can reduce approximately 23% of the aggregation missing in comparison with the other aggregation policies while it increases only 6% of the aggregation delay.
**Key words** : RSS feed, RSS feed aggregator, aggregation policy

## 1. 서 론

지난 몇 년 동안 블로그(Blog) 수가 급속도로 증가하면서, RSS 서비스의 사용도 함께 확산되었다[1]. RSS는 Really Simple Syndication, Rich Site Summary, RDF Site Summary의 약칭으로 웹 콘텐츠 배급을 위한 XML기반 포맷(format)이다[1-3]. 일반적으로 블로그 및 뉴스 제공 사이트와 같이 정보 업데이트가 자주 발생하는 웹사이트에서 정보를 사용자에게 쉽게 배포하기 위한 방법으로 RSS를 사용한다. RSS 구조는 그림 1과 같이 RSS 피드 제공 사이트를 나타내는 채널(channel)과 생성되는 정보(포스팅)를 나타내는 아이템(item)으로 구성되고, 세부 항목으로 제목(title), 주소(link), 내용(description), 생성된 시간(pubDate) 등이 제공된다[2].

RSS 서비스 방식은 정보제공자가 사용자에게 정보를 보내는 푸시(Push) 방식(예를 들면, 전자메일 발송)이 아닌, 구독을 원하는 사용자가 정보제공자에 의해 게시된 정보를 가져가는 풀(Pull) 방식으로 이루어진다.

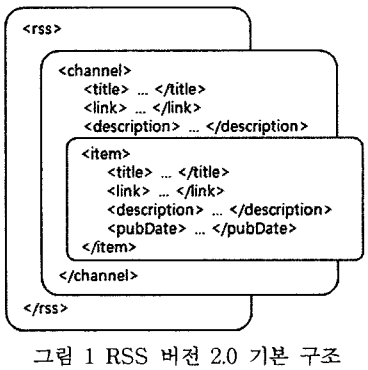

그림 1 RSS 버전 2.0 기본 구조

† 학생회원 : 숭실대학교 컴퓨터학과
younggeun@gmail.com
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
shlee199@gmail.com
논문접수 : 2007년 12월 10일
심사완료 : 2008년 7월 11일


정보과학회논문지: 데이타베이스 제35권 제5호(2008.10)

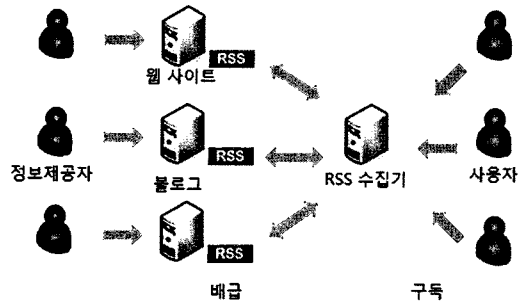


그림 2 RSS 피드 배급 및 구독 프로세스

기본적인 RSS 피드 배급(syndication)과 구독(subscription) 프로세스는 그림 2와 같다. 정보제공자(publisher)가 RSS 피드를 게시하면 사용자는 구독을 원하는 RSS 피드 주소를 수집기에 등록하고, RSS 피드 수집기(RSS feed aggregator 또는 RSS feed reader)는 등록된 RSS 피드를 수집하여 사용자에게 제공한다.

RSS 피드 수집 결과를 향상시키는 방법에는 하드웨어나 네트워크 자원을 증가시키는 방법도 있으나, 동일한 자원을 사용하여도 각 RSS 피드들을 얼마나 자주 수집하고, 어느 시점에 수집할 것인가를 결정하는 수집 정책 수립을 통해서도 RSS 수집 효율성을 향상시킬 수 있다.

효율적으로 RSS 피드를 수집하기 위한 RSS 피드 모니터링 알고리즘에 대한 연구도 있었다. Sia와 Cho[4, 5]는 RSS 피드 수집 정책 수립을 위한 이론적인 프레임워크(framework)를 제시하였고, 포스팅 패턴(posting pattern) 기반 자원 할당 및 스케줄링 방법을 소개하였으며, 사용자의 접근 패턴(user browsing pattern) 정보를 이용하여 RSS 피드를 모니터링하는 방법도 연구하였다.

본 논문은 RSS 피드 수집 시 수집에서 누락되는 포스팅 수를 최소화하기 위한 RSS 피드 수집 정책을 제안하고, 실험을 통해 제안한 수집 정책과 기존 수집 정책을 비교 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 RSS 피드 수집 정책을 설명하며, 3장에서는 기존 수집 정책의 문제점과 본 논문에서 제안하는 수집 정책에 대해서 설명한다. 4장에서는 RSS 피드 수집 결과 및 실험을 통해서 기존 수집 정책과 본 논문에서 제안한 수집 정책을 비교 평가한 결과를 기술한다. 마지막으로 5장에서는 비교 평가된 내용에 대한 결론 및 향후 연구 계획을 제시한다.

## 2. 최소 지연 수집 정책

RSS 피드 수집기가 n개의 RSS 피드를 수집한다고 가정할 때, 가장 간단한 수집 정책은 n개의 RSS 피드를 동일한 횟수로 수집하는 정책(이하 동일횟수수집정책)이다. 그러나 이 수집 정책으로 수집할 경우에는 정보가 많이 생성되는 RSS 피드와 정보가 적게 생성되는 RSS 피드가 동일하게 수집되기 때문에 비효율적이다.

### 2.1 수집지연

RSS 피드 수집 정책을 평가하는 방법으로 수집 지연(Delay)이 있다[4]. 수집 지연은 생성된 포스팅이 수집될 때까지 소요된 시간을 말한다. RSS 피드 $F$에서 $t_1,\ldots,t_k$ 시점에 포스팅 $p_1,\ldots,p_k$가 생성되고, RSS 피드 수집기에 의해서 $a_1,\ldots,a_m$ 시점에 수집된다고 가정하자. 포스팅이 수집된 시점 $a_j$는 $a_1,\ldots,a_m$ 중 포스팅이 생성된 시점 $t_i$ 보다 이후($t_i \le a_j$)인 값 중 최소값이고, 포스팅 $p_i$의 수집 지연 $D(p_i)$는 다음과 같다.

$$D(p_i) = a_j - t_i$$

수집 지연은 그림 3과 같이 $a_j$가 $t_i$에서 멀수록 커지고, 가까울수록 줄어든다.

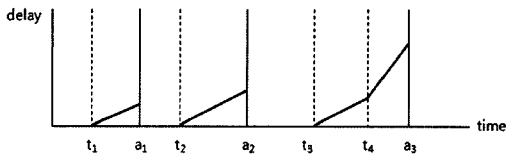


그림 3 수집 지연

RSS 피드 $F$에서 생성되는 모든 포스팅 $p_1,\ldots,p_k$에 대한 전체 수집 지연은 다음과 같다.

$$D(F) = \sum_{i=1}^{k} D(p_i) = \sum_{i=1}^{k} (a_j - t_i),\ where\ t_i \in [a_{j-1}, a_j]$$

수집 지연은 수집 시점 $a_j$과 그 이전 수집 시점 $a_{j-1}$ 사이에 생성된 정보들의 생성 시간을 대상으로 계산하기 때문에 $t_i \in [a_{j-1}, a_j]$ 조건이 필요하다. 1개의 RSS 피드 $F_i$에 대한 전체 수집 지연은 $D(F_i)$이므로 n개의 RSS 피드를 수집할 경우, 모든 RSS 피드를 수집하는 동안 발생하는 전체 수집 지연은 다음과 같다.

$$D(A) = \sum_{i=1}^{n} D(F_i)$$

**예제 1.** 그림 4와 같이 RSS 피드에서 포스팅 $p_1,\ldots,p_5$가 $t_1,\ldots,t_5$ 시점에 생성되고, $a_1$, $a_2$ 시점에 수집될 경우, $D(p_1)$은 $p_1$이 수집된 시간 $a_1$과 생성된 시간 $t_1$의 차이 즉, 3이 된다.

수집 지연은 수집 시점 $a_j$와 이전 수집 시점 $a_{j-1}$ 사이에 생성된 정보들의 생성 시간을 대상으로 계산하므로 $p_1$, $p_2$은 $a_1$를 기준으로, $p_3$, $p_4$, $p_5$은 $a_2$를 기준으로

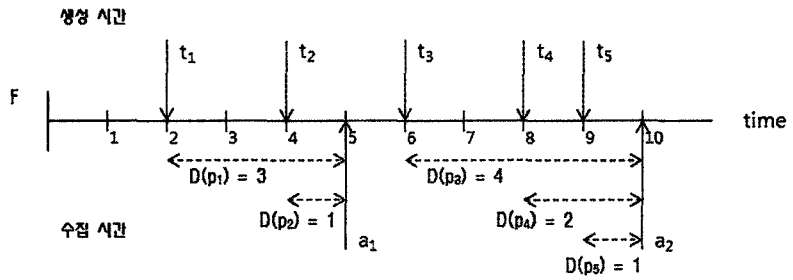


그림 4 수집 지연 예시

수집 지연을 계산한다. 따라서 RSS 피드에서 생성된 모든 포스팅 $p_1,\ldots,p_5$에 대한 수집 지연은 다음과 같다.

$$D(F)=\sum_{i=1}^{5}D(p_i)=\sum_{i=1}^{5}(a_j-t_i),\ where\ t_i\in[a_{j-1},a_j]$$
$$=D(p_1)+D(p_2)+D(p_3)+D(p_4)+D(p_5)$$
$$=(5\text{-}2)+(5\text{-}4)+(10\text{-}6)+(10\text{-}8)$$
$$+(10\text{-}9)=11$$ ■

### 2.2 최소지연수집정책

동일횟수수집정책보다 수집 지연을 줄이기 위한 방법으로 Sia와 Cho[4]는 각 RSS 피드의 포스팅율(posting rate)에 따라 수집횟수를 결정하는 방법(이하 최소지연수집정책)을 제안하였다. 포스팅율은 특정 기간에 생성되는 포스팅 수를 의미하는데, 예를 들어 1일 포스팅율은 1일 동안 생성된 포스팅 수를 나타낸다. 최소지연수집정책은 RSS 피드 수집기가 수행 가능한 수집횟수가 한정되어 있다고 가정하고, 포스팅을 많이 생성하는 RSS 피드일수록 수집횟수를 많이 할당하여 수집 지연을 줄인다.

RSS 피드 $F_i$에 대한 수집 횟수 $\propto$ RSS 피드 $F_i$의 포스팅율

RSS 피드 $F_1,\ldots,F_n$이 각 $F_i$의 포스팅율 $\lambda_i$와 중요도에 따른 가중치 $w_i$를 가지고 있고, RSS 피드 수집기가 기간 $I$ 동안에 최대로 수집 가능한 횟수가 $M$일 경우, $F_i$의 수집 횟수 $m_i$는 아래와 같이 계산한다.

$$m_i=k\sqrt{w_i\lambda_i}$$

이때 $k$는 $\sum_{i=1}^{n}k\sqrt{w_i\lambda_i}=M$을 만족시키는 상수이다.

**예제 2.** 그림 5와 같이 수집 대상 RSS 피드 $F_1$, $F_2$, $F_3$, $F_4$가 존재하고, $\lambda_i$가 30개, 30개, 10개, 10개, 모든 RSS 피드의 중요도는 동일($w_i$=1)하며, $M$은 8회라고 가정하자.

최소지연수집정책으로 수집횟수(표 1 참조)를 할당한다면, $k$는 $\sum_{i=1}^{n}k\sqrt{w_i\lambda_i}=M$을 만족시키는 상수이므로,

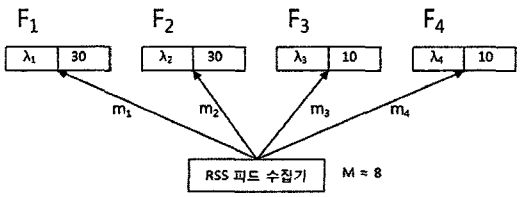


그림 5 수집 대상 RSS 피드 예시

$\sum_{i=1}^{4}k\sqrt{\lambda_i}=8$, $k$는 약 0.46이 되고, $m_1$, $m_2$, $m_3$, $m_4$는 각각 3회, 3회, 1회, 1회가 된다.

동일횟수수집정책으로 수집한다면, $M$이 8회이므로 수집횟수를 각 RSS 피드별로 동일하게 나누어서 각각 2회씩 수집하게 된다. ■

## 3. 최소 누락 수집 정책

### 3.1 수집누락

RSS 피드는 생성된 포스팅 전체를 저장하지 않고,

표 1 최소지연수집정책에서의 수집횟수 할당 예시

| RSS 피드($F_i$) | 포스팅율($\lambda_i$) | 중요도($w_i$) | $k\sqrt{w_i\lambda_i}$ | 수집횟수($m_i$) |
|---|---|---|---|---|
| $F_1$ | 30개 | 1 | 2.535898385 | 3회 |
| $F_2$ | 30개 | 1 | 2.535898385 | 3회 |
| $F_3$ | 10개 | 1 | 1.464101615 | 1회 |
| $F_4$ | 10개 | 1 | 1.464101615 | 1회 |

최근에 생성된 포스팅 일부만 저장한다. 본 논문에서는 i번째 피드에서 저장되는 포스팅의 개수를 $S_i$로 표시한다. RSS 피드에 저장되는 포스팅 수가 한정되어 있으므로, RSS 피드 수집기에 의해 수집되기 전에 RSS 피드에서 삭제되는 포스팅이 발생될 수 있다.

**예제 3.** 예제 2에서 최소지연수집정책에 의해서 수집 횟수를 각각 1회씩 할당받은 $F_3$, $F_4$가 그림 6과 같이 $S_3$은 10, $S_4$는 5라고 가정하자.

$F_3$은 $S_3$가 10이므로 10개의 포스팅이 모두 저장되어 있고, 1회 수집으로 10개 모두 수집이 가능하다. 그러나 $F_4$는 $S_4$가 5이므로 최근 포스팅 $p_6$, ..., $p_{10}$이 생성되는 동안 이전 포스팅 $p_1$, ..., $p_5$은 삭제되기 때문에 1회 수집으로는 최근 5개의 포스팅만 수집할 수 있고, 이전에 생성된 5개의 포스팅은 수집할 수 없다. ■

RSS 피드에서 생성된 포스팅 중에서 RSS 피드 수집기가 수집하지 못한 포스팅을 본 논문에서는 누락 포스팅(Missing posting)이라고 정의하고, i번째 RSS 피드에서 발생하는 누락 포스팅 수를 $MP(F_i)$로 나타낸다. RSS 피드 수집기가 n개의 RSS 피드를 수집할 때 발생하는 전체 누락 포스팅 수는 다음과 같다.

$$MP(A) = \sum_{i=1}^{n} MP(F_i)$$

본 논문에서는 i번째 RSS 피드에서 j번째 수집 후에 수집할 수 있는 총 포스팅 수를 $TP_{i,(j+1)}$(target postings), i번째 RSS 피드를 j번째 수집할 때 수집하는 포스팅 수를 $PPA_{i,j}$(postings per aggregation)로 나타낸다.

$$TP_{i,(j+1)} = TP_{i,j} - PPA_{i,j}$$

RSS 피드 $F_i$에서 생성된 모든 포스팅 수가 $\lambda_i$이므로 수집 전(j=0)에 수집할 수 있는 총 포스팅 수 $TP_{i,1} = \lambda_i$이고, $F_i$를 $m_i$회 수집 후(j=$m_i$)에 수집할 수 있는 총 포스팅 수 $TP_{i,(m_i+1)}$는 $MP(F_i)$가 된다.

$$MP(F_i) = \lambda_i - \sum_{j=1}^{m_i} PPA_{i,j}$$

### 3.2 최소누락수집정책

본 논문에서 제안하는 수집 정책(이하 최소누락수집정책)은 누락 포스팅 수를 최소화하기 위해 그림 7과 같이 $PPA_{i,j}$의 값이 최대인 $F_i$에 수집횟수 1회를 할당하며, 이를 $M$회 반복한다.

RSS 피드 수집기가 $F_i$를 1회 수집할 때 수집하는 포스팅 수(즉 $PPA_{i,j}$)는 $TP_{i,j}$가 $S_i$보다 클 경우 $S_i$가 되고, $TP_{i,j}$가 $S_i$보다 작을 경우 $TP_{i,j}$가 된다. 그러나 $TP_{i,j}$가 $S_i$보다 클 경우 $PPA_{i,j}$ 값이 $S_i$가 되는 것은 RSS 피드 수집기가 $F_i$를 수집할 시점에 $S_i$만큼의 신규 포스팅이 생성되었다는 가정이 필요하다.

본 논문에서는 RSS 피드 수집기가 $F_i$를 수집할 시점에 $S_i$만큼의 신규 포스팅이 생성되지 않았을 경우를 고려하여 $TP_{i,(j+1)}$의 합이 0이 된 후에도 총 수집횟수가 $M$보다 작을 경우, 모든 $TP_{i,(j+1)}$를 $\lambda_i$로 설정하고, 총 수집횟수가 $M$이 될 때까지 수집횟수 할당을 반복한다.

**예제 4.** 그림 8과 같이 수집할 RSS 피드 $F_1$, $F_2$, $F_3$, $F_4$가 1일 포스팅을 30개, 30개, 10개, 10개 생성하고, $S_i$가 각각 15개, 10개, 10개, 5개이며, RSS 피드의 중요도는 동일($w_i$=1)하다고 가정하자. 또한 RSS 수집기가 수집 가능한 수집횟수 $M$을 8이라고 하자.

최소누락수집정책으로 수집횟수를 할당한다면, $M$이 8이므로 $PPA_{i,j}$를 계산(표 2 참조)하여 $PPA_{i,j}$가 가장 큰 RSS 피드에 수집횟수 1회를 할당하는 과정을 8회 실시한다.

표 3은 세 가지 수집 정책에 의해 할당된 RSS 피드별 수집횟수와 누락 포스팅 수를 비교하여 나타낸다.

동일횟수수집정책(1), 최소지연수집정책(2), 최소누락수집정책(3)으로 수집할 경우 $MP(A)$는 다음과 같다.

$$(1)\ MP(A) = \sum_{i=1}^{4} MP(F_i) = \sum_{i=1}^{4} (\lambda_i - \sum_{j=1}^{m_i} PPA_{i,j})$$

$$= (30-(15+15)) + (30-(10+10)) + (10-(10+0)) + (10-(5+5)) = 10$$

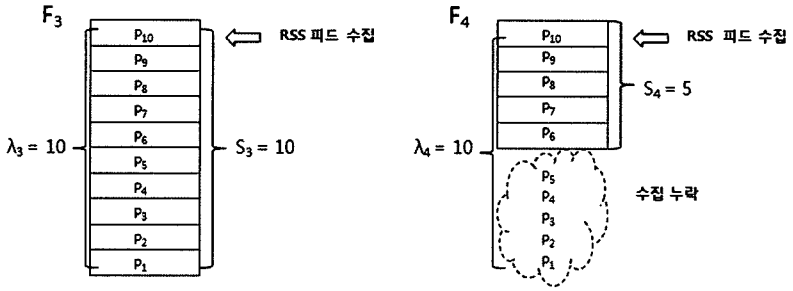


그림 6 수집 누락 예시

```
알고리즘   최소누락수집정책 알고리즘
Input : M,                                  // M : RSS 수집기가 수행 가능한 수집 횟수
        n,                                  // n : 수집대상 RSS 피드 개수
        λi = {λ1, ...,λn},                  // λi : i번째 RSS 피드의 포스팅율
        Si = {S1, ..., Sn}                  // Si : i번째 RSS 피드에 저장되는 포스팅 수
프로시저
 [1]   for i ← 1 to n
 [2]       mi = 0, TPi,1 = λi               // mi : i번째 RSS 피드의 수집횟수
 [3]   end for
 [4]   total_TP = Σ_{i=1}^{n} TPi,1         // total_TP : TPi,(j+1)의 합
 [5]   for total_m ← 1 to M
 [6]       if ( total_TP = 0 ) then         // TPi,(j+1)이 모두 0일 때, λi로 설정
 [7]           for i ← 1 to n
 [8]               j = mi + 1
 [9]               TPi,j = λi
 [10]          end for
 [11]      for i ← 1 to n
 [12]          j = mi + 1
 [13]          if ( TPi,j ≥ Si) then        // PPAi,j 계산
 [14]              PPAi,j  ≈ Si
 [15]          else
 [16]              PPAi,j = TPi,j
 [17]      end for
 [18]      Find the maximum value of PPAi,j (where 1≤i≤n, j=mi+1)
 [19]      Let indexes i and j of maximum PPAi,j be maxI, maxJ, respectively
 [20]      mmaxI = mmaxI + 1                // PPAi,j가 최대인 RSS 피드에 수집횟수 할당
 [21]      TPmaxI,(maxJ+1) = TPmaxI,maxJ - PPAmaxI,maxJ
 [22]      total_TP = Σ_{i=1}^{n} TPi,(mi+1)
 [23]  end for

TPi,(j+1)(target postings) : i번째 RSS 피드에서 j번째 수집 후에 수집할 수 있는 총 포스팅 수
PPAi,j(postings per aggregation) : i번째 RSS 피드에서 j번째 수집할 때, 수집하는 포스팅 수
```

그림 7 최소누락수집정책 알고리즘

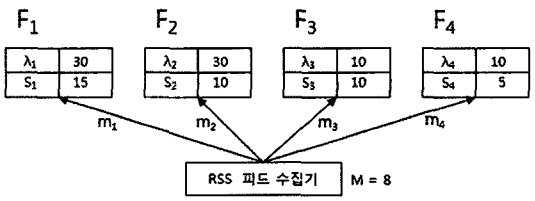


그림 8 $S_i$가 추가된 수집 대상 RSS 피드 예시

(2) $MP(A)$ = (30-(15+15+0)) + (30-(10+10+10)) + (10-10) + (10-5) = 5

(3) $MP(A)$ = (30-(15+15)) + (30-(10+10+10)) + (10-10) + (10-(5+5)) = 0 ■

## 4. 실험

### 4.1 수집 데이타

본 논문에서 실험을 위하여 구현한 RSS 피드 수집기의 전체적인 구조는 그림 9와 같다. 수집할 RSS 피드의 URL을 입력기(Importer)를 통해 RSS 피드 주소 DB(Feed URL)에 입력한다. DB에 저장된 RSS 피드 주소는 피드 적재기(Feed Fetcher)에 의해서 큐(queue)에 적재된다. 피드 리더기(Feed Reader)는 큐에서 RSS 피드 URL을 가져오고, RSS 피드를 읽어서 필요한 요소(element)들을 파싱(parsing)하고 DB의 포스팅 저장소(Posts repository)에 저장한다. 피드 모니터기(Feed Monitor)는 포스팅 저장소에서 각 RSS 피드별 $\lambda_i$와 $S_i$를 계

표 2 $PPA_{i,j}$ 계산을 통한 수집횟수 할당 예시

| $M$ | 최대 $PPA_{i,j}$ 검색을 위한 계산 | | | | | | | | | | | | | | | | $m_i$ 할당 결과 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_1$ | | | | $F_2$ | | | | $F_3$ | | | | $F_4$ | | | | $m_1$ | $m_2$ | $m_3$ | $m_4$ |
| 1 | $TP_{1,1}$ | 30 | $PPA_{1,1}$ | 15 | $TP_{2,1}$ | 30 | $PPA_{2,1}$ | 10 | $TP_{3,1}$ | 10 | $PPA_{3,1}$ | 10 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 1 | 0 | 0 | 0 |
| 2 | $TP_{1,2}$ | 15 | $PPA_{1,2}$ | 15 | $TP_{2,1}$ | 30 | $PPA_{2,1}$ | 10 | $TP_{3,1}$ | 10 | $PPA_{3,1}$ | 10 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 2 | 0 | 0 | 0 |
| 3 | $TP_{1,3}$ | 0 | $PPA_{1,3}$ | 0 | $TP_{2,1}$ | 30 | $PPA_{2,1}$ | 10 | $TP_{3,1}$ | 10 | $PPA_{3,1}$ | 10 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 2 | 1 | 0 | 0 |
| 4 | $TP_{1,3}$ | 0 | $PPA_{1,3}$ | 0 | $TP_{2,2}$ | 20 | $PPA_{2,2}$ | 10 | $TP_{3,1}$ | 10 | $PPA_{3,1}$ | 10 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 2 | 2 | 0 | 0 |
| 5 | $TP_{1,3}$ | 0 | $PPA_{1,3}$ | 0 | $TP_{2,3}$ | 10 | $PPA_{2,3}$ | 10 | $TP_{3,1}$ | 10 | $PPA_{3,1}$ | 10 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 2 | 3 | 0 | 0 |
| 6 | $TP_{1,3}$ | 0 | $PPA_{1,3}$ | 0 | $TP_{2,4}$ | 0 | $PPA_{2,4}$ | 0 | $TP_{3,1}$ | 10 | $PPA_{3,1}$ | 10 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 2 | 3 | 1 | 0 |
| 7 | $TP_{1,3}$ | 0 | $PPA_{1,3}$ | 0 | $TP_{2,4}$ | 0 | $PPA_{2,4}$ | 0 | $TP_{3,2}$ | 0 | $PPA_{3,2}$ | 0 | $TP_{4,1}$ | 10 | $PPA_{4,1}$ | 5 | 2 | 3 | 1 | 1 |
| 8 | $TP_{1,3}$ | 0 | $PPA_{1,3}$ | 0 | $TP_{2,4}$ | 0 | $PPA_{2,4}$ | 0 | $TP_{3,2}$ | 0 | $PPA_{3,2}$ | 0 | $TP_{4,2}$ | 5 | $PPA_{4,2}$ | 5 | 2 | 3 | 1 | 2 |

표 3 수집 정책별 수집횟수, 누락 포스팅 수 비교 예시

| $F_i$ | $\lambda_i$ | $S_i$ | 동일횟수수집정책 | | 최소지연수집정책 | | 최소누락수집정책 | |
|---|---|---|---|---|---|---|---|---|
| | | | $m_i$ | $MP(F_i)$ | $m_i$ | $MP(F_i)$ | $m_i$ | $MP(F_i)$ |
| $F_1$ | 30 | 15 | 2 | 0 | 3 | 0 | 2 | 0 |
| $F_2$ | 30 | 10 | 2 | 10 | 3 | 0 | 3 | 0 |
| $F_3$ | 10 | 10 | 2 | 0 | 1 | 0 | 1 | 0 |
| $F_4$ | 10 | 5 | 2 | 0 | 1 | 5 | 2 | 0 |

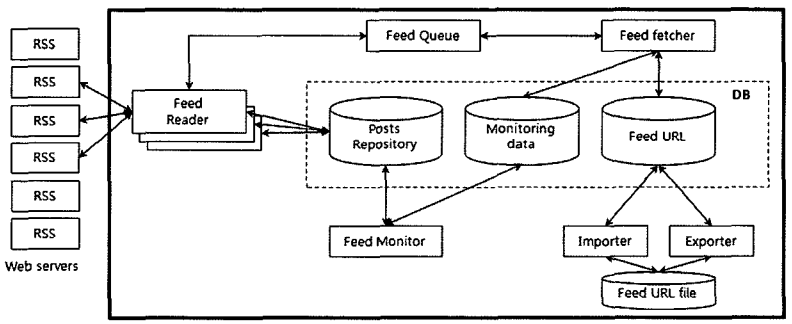


그림 9 RSS 피드 수집기 구조

산하여 모니터링 DB(Monitoring data)에 저장한다.

본 논문에서는 실험을 위하여 블로그 허브 사이트 올블로그(http://www.allblog.net)[6]에서 무작위로 1,000개의 RSS 피드를 수집 대상으로 선정하고, 6주간 2시간 간격으로 총 45,513개의 포스팅을 수집하였다. 2시간 간격으로 수집할 경우 누락 포스팅 발생 여부를 확인하기 위해서 이전 수집된 포스팅과 2시간 후 수집된 포스팅을 비교하였다. 만약 중복된 포스팅이 있다면 아직 해당 RSS 피드의 $S_i$만큼 신규 포스팅이 생성되지 않으며, 이는 누락 포스팅이 발생되지 않았다는 것을 의미한다. 확인 결과 1,000개의 피드 중에서 3개를 제외한 나머지 피드는 중복된 포스팅이 존재하였고, 본 논문에서는 2시간 간격으로 수집된 포스팅이 RSS 피드에서 생성된 모든 포스팅으로 가정하였다.

수집 대상 RSS 피드 주소의 도메인 분포는 표 4와 같다. 상위 5개 도메인의 개수가 80% 이상을 차지했다.

표 4 RSS 피드 주소의 도메인 분포 (상위 5개)

| RSS 피드 수 | 도메인 |
|---|---|
| 302 | daum.net |
| 283 | egloos.com |
| 112 | tistory.com |
| 97 | naver.com |
| 25 | feedburner.com |

전체 RSS 피드에서 수집한 일일 포스팅 수는 그림 10과 같다. 일일 평균 1,083개의 포스팅이 수집되었으며, 주말에는 평균 892개로 주중 평균 1,160개 보다는 포스팅 수가 적었다.

RSS 피드의 1일 포스팅율 분포는 그림 11과 같다. 1일 10개 이하의 포스팅을 생성하는 RSS 피드의 수가 전체 RSS 피드 수의 94%를 차지하며, 1일 10개 이상의

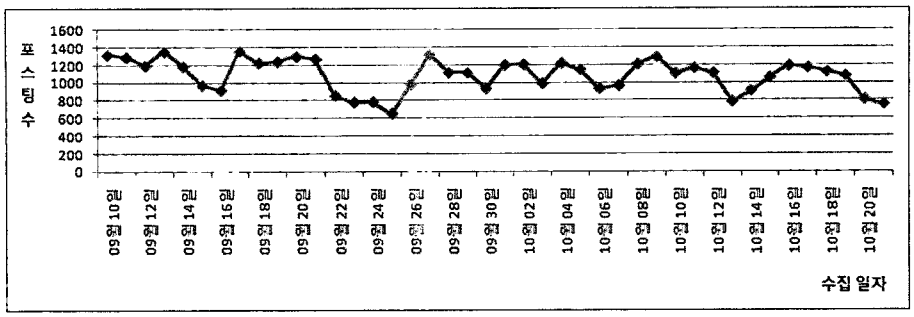


그림 10 전체 RSS 피드에서 수집한 일일 포스팅 수

표 5 $S_i$에 따른 RSS 피드 분포

| $S_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 20 | 25 | 30 | 37 | 60 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 피드 수 | 16 | 3 | 2 | 2 | 48 | 2 | 6 | 3 | 5 | 436 | 1 | 1 | 3 | 1 | 374 | 2 | 1 | 1 | 7 | 6 | 63 | 1 | 1 |

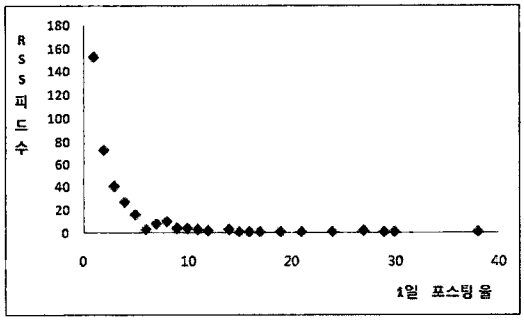


그림 11 1일 포스팅율 분포

포스팅을 생성하는 RSS 피드의 수는 적게 분포하였다.

표 5는 RSS 피드들의 $S_i$ 분포를 나타낸다. 약 83%의 피드가 10개에서 15개의 포스팅을 저장하고, 나머지 피드는 1개에서부터 60개까지 분포한다.

### 4.2 수집 정책 비교 평가

RSS 수집 정책의 효율성 평가는 동일횟수수집정책(1), 최소지연수집정책(2), 그리고 본 논문에서 제안하는 최소누락수집정책(3)을 수집 지연 정도와 누락 포스팅 수를 기준으로 비교 평가하였다. RSS 피드별 중요도 $w_i$는 모두 동일하다고 가정하였으며, RSS 피드 수집기의 최대 수집 횟수 $M$을 최소 1,000회에서 최대 20,000회로 설정을 변경하면서 각 수집 정책별 수집 지연 정도와 누락 포스팅 수 변화를 확인하였다.

RSS 수집 정책의 효율성 평가를 위해서는 그림 12와 같은 순서의 계산이 필요하다. 6주간의 수집 데이타 중, 처음 3주간의 $\lambda_i$와 $S_i$를 계산하고, 3가지 수집 정책으로 각 RSS 피드별 수집횟수를 계산한다. 수집횟수가 결정되면 각 수집이 동일한 간격으로 수행되도록 수집시점을 계산한다. 최종적으로 각 RSS 피드별 수집시점과 다음 3주간 수집된 포스팅의 생성 시간과 생성된 수를 이

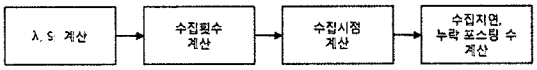


그림 12 RSS 수집 정책 비교 평가를 위한 계산 순서

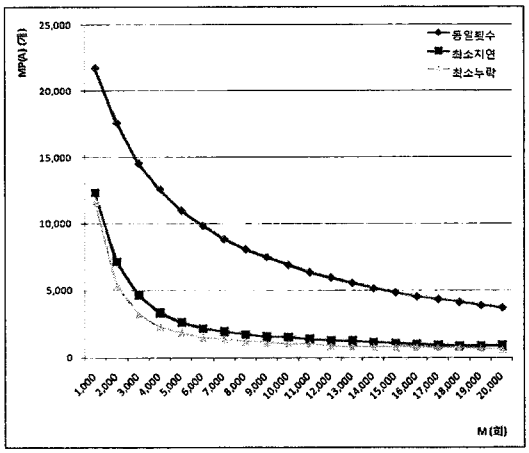


그림 13 수집 정책별 M에 따른 누락 포스팅 수

용하여 수집 지연과 누락 포스팅 수를 계산한다.

수집 정책 비교 평가 결과는 그림 13과 같다. $M$을 증가시킬수록 모든 수집 정책에서 발생하는 누락 포스팅 수는 감소하지만, 최소누락수집정책이 모든 수집에서 가장 적은 수의 누락 포스팅이 발생했다. 3주간의 전체 포스팅 수(22,399개)를 기준으로 했을 때, 동일횟수수집정책은 평균 37%, 최소지연수집정책은 평균 11%, 최소누락수집정책은 평균 8%의 누락 포스팅이 발생하였다. 최소누락수집정책으로 수집할 때, 동일횟수수집정책 대비 77%, 최소지연수집정책 대비 23%의 누락 포스팅 수가 감소되었다.

수집된 포스팅 수 대비 $D(A)$를 비교한 결과는 그림 14

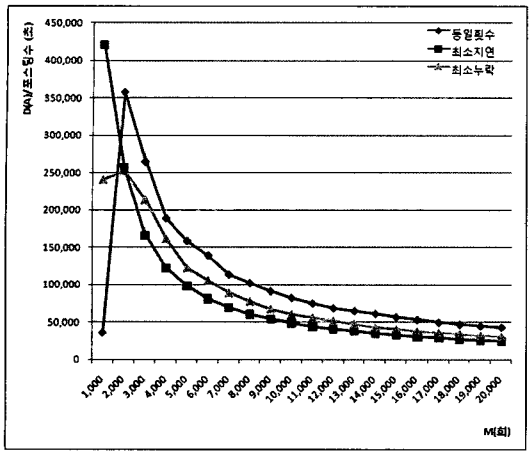


그림 14 수집 정책별 M에 따른 수집 지연

표 6 수집정책별 비교 분석

| | 최소누락수집정책 | |
|---|---|---|
| | 누락 포스팅 관점 | $D(A)$/포스팅수 관점 |
| 동일횟수수집정책 | 0.23 | 0.86 |
| 최소지연수집정책 | 0.77 | 1.06 |

와 같다. 대부분의 경우(본 실험에서는 $M$이 2,000이상)에서는 최소지연수집정책이 가장 적은 수집 지연을 발생하였으며, 최소누락수집정책은 최소지연수집정책보다는 많은 수집 지연을 발생하나 동일횟수수집정책보다는 적은 수집 지연을 발생하였다. 평균 수집 지연 정도를 비교하였을 때, 표 6과 같이 최소누락수집정책은 동일횟수수집정책 대비 평균 14%의 감소율을 보였으나, 최소지연수집정책 대비 평균 6% 증가하였다.

본 논문에서 고려하는 세 가지 정책을 비교하면 표 6과 같다. 누락 포스팅 관점에서는 최소지연수집정책에서의 77%만이 최소누락수집정책에서 누락되었으며, 지연시간 관점에서는 최소지연수집정책보다 최소누락수집정책이 1.06배 증가하였다. 즉, 최소누락수집정책은 최소지연수집정책과 비교하여 6%의 수집 지연으로 23%의 포스팅 누락을 해결하였다. 최소누락수집정책이 효과적으로 누락 포스팅 문제를 해결한다고 생각한다.

## 5. 결 론

기존 최소지연수집정책이 수집의 신속성을 우선시하는 정책이었다면, 본 논문에서 제안하는 최소누락수집정책은 수집량을 우선시하는 정책이다. 본 논문에서는 RSS 수집 시 발생하는 수집 누락 문제점을 제시하고, 누락 포스팅 수를 최소화하기 위한 수집 정책을 제안하였다. 본 논문에서는 RSS 피드 1,000개를 6주간 수집하고, 수집된 데이타를 이용하여 동일횟수수집정책, 최소지연수집정책, 최소누락수집정책으로 수집할 경우의 수집 지연과 누락 포스팅 수를 비교 평가하였다. 실험 결과, 최소누락수집정책은 기존 최소지연수집정책보다 수집 지연이 증가한 반면, 수집 누락은 감소하는 것을 알 수 있었다. 이러한 정보는 RSS 서비스 분야에서 수집의 신속성과 수집량의 우선순위에 따라 어떤 수집 정책을 선택할 것인가에 대한 지침이 될 수 있다.

본 연구는 RSS 피드를 수집할 때마다 RSS 피드에 저장되는 포스팅 수 만큼 신규 포스팅이 생성된다는 가정을 기반으로 수집 정책을 제안하였다. 그러나 실제 RSS 피드에서는 포스팅이 다양한 패턴으로 생성되기 때문에, 실제적으로 누락 포스팅을 최소화하기 위한 수집횟수는 가정 하에서의 수집횟수보다 많이 필요했다.

향후 다양한 주기에서의 RSS 피드별 포스팅 패턴을 분석하여 포스팅 시간을 예측하고 수집하는 연구가 필요하다. 또한 수집 지연 및 누락 포스팅을 줄이기 위해 포스팅이 되는 시점을 예측하여 수집할 경우, 잘못된 예측으로 인한 수집 지연이 증가하거나, 누락 포스팅이 발생할 수 있으므로, 포스팅 예측의 정확성을 높이기 위한 연구도 필요하다.

## 참 고 문 헌

[1] K. E. Gill, "Blogging, RSS and the Information Landscape: A Look At Online News," WWW 2005 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, 2005.

[2] RSS 2.0 Specification, http://blogs.law.harvard.edu/tech/rss.

[3] What is RSS?, http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html.

[4] K. C. Sia, J. Cho and H. K. Cho, "Efficient Monitoring Algorithm for Fast News Alert," IEEE Transaction on Knowledge and Data Engineering, Vol.19, No.7, pp. 950-961, 2007.

[5] K. C. Sia, J. Cho, K. Hino, Y. Chi, S. Zhu and B. L. Tseng, "Monitoring RSS Feeds Based on User Browsing Pattern," In Proceedings of the International Conference on Weblogs and Social Media, 2007.

[6] Allblog, http://www.allblog.net.

[7] B. E. Brewington and G. Cybenko, "How Dynamic is the Web?" In Proceedings of the 9th International World Wide Web Conference, pp. 257-276, 2000.

[8] J. Cho and H. Garcia-Molina, "Synchronizing a Database to Improve Freshness," In Proceedings the 26th ACM SIGMOD International Conference on Management of Data, pp. 117-128, 2000.

[9] J. Cho and H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental

Crawler," In Proceedings of the 26th International Conference on Very Large Data Bases, pp. 200-209, 2000.

[10] S. J. Kim and S. H. Lee, "An Empirical Study on the Change of Web Pages," In Proceedings of the 7th Asia-Pacific Web Conference, pp. 632-642, 2005.

[11] S. J. Kim and S. H. Lee, "Estimating the Change of Web Pages," In Proceedings of the International Conference on Computational Science 2007, pp. 798-805, 2007.

[12] A. Ntoulas, J. Cho, and C. Olston, "What's New on the Web? The Evolution of the Web from a Search Engine Perspective," In Proceedings of the 13th International World Wide Web Conference, pp. 1-12, 2004.

한 영 근

2003년 한림대학교 컴퓨터공학과(학사) 2008년 숭실대학교 대학원 컴퓨터학과(석사). 관심분야는 인터넷 데이타베이스, 데이타베이스

이 상 호

1984년 서울대학교 컴퓨터공학과(학사) 1986년 미국 노스웨스턴 대학교 전산학과(석사). 1989년 미국 노스웨스턴 대학교 전산학과(박사). 1990년~1992년 한국전자통신연구원 선임연구원. 1999년~2000년 미국 조지메이슨 대학교 소프트웨어정보공학과 교환 교수. 1992년~현재 숭실대학교 컴퓨터학과 교수. 관심분야는 인터넷 데이타베이스, 데이타베이스 튜닝 및 성능 평가