선형과 고리 구조 네트워크에서의 웹 프락시의 최적 개수와 위치

정행은 최정임 이상규 문봉희
숙명여자대학교 컴퓨터과학과
{haengeun, jungim, sanglee, moon}@cs.sookmyung.ac.kr

Optimal Number and Location for Web Proxies in Linear and Ring Networks

Haengeun Chung Jungim Choi Sang-Kyu Lee Bong-Hee Moon
Dept. of Computer Science, Sookmyung Women's University

요 약

WWW의 대중화를 통한 인터넷 사용자의 증가로 발생되는 통신지연의 해결책으로 웹 캐싱 방법이 고려되어 왔다. 웹 서버의 용량초과로 서비스시스템을 확장하고자 할 때 서버 운영자의 측면에서는 사용자의 서비스 응답시간을 줄이는 것 못지 않게 서버를 확장하고 유지하는데 드는 비용을 최소화하는 것도 중요한 사항이다. 그러므로 본 논문에서는 인터넷상에서 프락시가 될 수 있는 잠재노드들이 양방향 통신을 지원하는 링크들로 선형(linear)과 고리(ring) 구조를 이룰 때 서비스 노드들 각각이 정해진 임계 응답시간을 보장하기에 필요한 최소한의 프락시의 개수와 그의 위치를 동시에 찾아내는 알고리즘을 제안한다. 여기서 프락시의 위치는 위상의 어느 족에 위치해도 무방하다고 본다.

1. 서론

WWW의 확산을 통한 인터넷 사용자의 증가로 네트워크 상의 통신량은 급격한 증가를 보여왔다[1]. 이러한 현상은 요청되는 통신량이 제한된 네트워크의 한계를 초월함으로 인해 클라이언트로부터 들어오는 데이터 요구에 서버가 제때에 응답을 해주지 못하는 문제를 야기시켰다. 이와 같은 문제는 특히 서버나 네트워크의 혼잡(congestion), 부적절한 대역폭(bandwidth)을 가지는 링크, 그리고 propagation delay등으로 인해 야기되는데 이러한 통신 지체(latency)를 줄이는 가장 효율적인 방법으로 클라이언트와 서버사이에 캐싱을 위해 중간 서버를 따로 두어 서버의 부하를 줄이는 효과를 내는 웹 프락시 방법이 고려되고 있다[2, 3, 4, 5]. 이에 본 논문은 선형구조(linear topology)와 고리구조(ring topology) 네트워크에서 주어진 조건을 만족하도록 하면서 클라이언트의 요구에 응답할 수 있는 범위 내에서 필요한 최소의 웹 프락시 개수를 구하고, 이들의 알맞은 위치를 찾는 방법을 제시한다. 이와 관련된 연구로 [4, 5]에서는 선형구조와 트리구조에서 주어진 M개의 프락시의 최적위치를 찾는 방법을 제안했고 [3]에서는 선형구조에서 본 논문과 같은 문제를 다루었으나 프락시의 위치가 항상 그룹의 가장 작은 인덱스를 갖는 노드로 제약을 주었고, 같은 네트워크 모델을 갖는 [6]에서는 최소개수의 프락시가 아닌 주어진 M개의 프락시의 최적위치를 찾는 문제이므로 본 논문과는 다른 문제들이다.

2. 선형 구조

프락시가 될 수 있는 잠재노드들이 양방향 통신을 지원하는 링크들로 선형 구조 (linear topology)를 이루는 네트워크 상에서의 문제 해결을 위한 기본 모델은 다음과 같다. 편의상 노드의 번호를 웹 서버에서부터 순서대로 $1,2,\cdots,N$ 이라 하자. 이때 큰 숫자일수록 웹에서 먼 노드이다. 각각의 노드 i는 양의 실수인 가중치 $w(i)$를 갖고 $(0\le w(i)\in R)$ 이 가중치는 특정 시간동안에 각 노드에서 발생되는 통신 요청의 량을 나타낸다. 각각의 링크 (i,j)에도 양의 실수인 가중치 $d(i,j)$가 부여되는데 이 값은 링크 (i,j)사이의 데이터를 전달하는데 드는 비용을 나타낸다. $d(i,j)$를 노드 i와 노드 j 사이의 통신 시간(latency)으로 볼 수도 있고 이때 각 링크는 양방향 통신이 가능하며 $d(i,j)=d(j,i)$이고 임의의 세 노드 i,k,j에 대해 $(i\le k\le j)$, $d(i,k)+d(k,j)=d(i,j)$가 된다고 가정한다. 만약 $i=j$이면 $d(i,j)=0$이라 한다. $1\le i\le j\le N$인 노드의 범위 i,j에 대해 $W(i,j)$를 노드 i와 노드 j 사이에 있는 노드들의 가중치의 합인 $\sum_{i\le k\le j} w(k)$로 정의하고 노드 범위 i,j 사이의 노드들의 집합에서 프락시의 위치가 k라 할 때 $(i\le k\le j)$의 통신비용을 $cost(i,k,j)$로 표기한다. 이는 노드 i와 노드 j사이에 있는 노드들이 노드 k를 프락시로 하고 그 노드로부터 서비스를 받을 때 그 그룹에서 발생되는 통신에 대한 비용으로 노드 k를 제외한 노드들의 각각의 가중치에 노드 k로부터의 거리에 따른 비용을 곱한 값으로 계산 계산될 수 있다. 따라서 $cost(i,k,j)$는 노드 k를 중심으로 그 왼쪽에 위치한 노드들이 노드 k로부터 서비스 받는데 필요한 비용과 오른쪽에 위치한 노드들이 노드 k로부터 서비스 받는데 필요한 비용의 합으로 나타낼 수 있다. 따라서 그 값은 $\sum_{i\le i'<k} w(i')d(i',k)+\sum_{k<i'\le j} w(i')d(k,i')$이 된다. 본 논문에서는 N개의 노드들을 중복 없이 분할하고 서버가 속한 그룹을 제외한 각 그룹에서 하나의 프락시 노드의 위치를 구하는데 이때 서버와 프락시 노드들이 자신이 속한 그룹의 나머지 노드들에게 제공해야하는 서비스 비용이 주어진 임계값 (T_s)을 넘지 않게 하면서 프락시의 개수를 최소화하는데 목적을 둔다.

이때 서버가 속한 그룹은 서버가 서비스를 하게되고 나머지

그룹에서는 그룹 안에 있는 노드 중 서비스 비용을 최소화 할 수 있는 노드가 프락시로 선정되어야 하는데 Chen등이 [6]에서 제안한 선형구조의 특정 범위에서의 통신비용에 대한 특성을 응용하여 프락시 선정의 계산 시간을 줄였다. 노드 i에서 노드 j까지의 범위를 갖는 그룹에서 최소의 서비스 비용을 갖는 프락시의 위치를 $opt_loc(i,j)$라하고 그 그룹의 비용변환점을 $\sum_{i\le k'\le k} w(k') \ge \sum_{k+1\le k'\le j} w(k')$을 만족하는 가장 작은 정수 k ($i\le k\le j$)라 한다면 다음과 같은 특성을 관찰할 수 있다.

정리 1. *노드 i에서 노드 j를 범위로 하는 그룹의 비용변환점의 노드가 최소 서비스 비용을 갖는 프락시의 위치인 $opt_loc(i,j)$이 된다.*

증명: 최적 프락시의 위치 $opt_loc(i,j)$를 k라 하자. k가 프락시일 때 비용이 가장 작으므로 $cost(i,\ k,\ j)\le cost(i,\ k-1,\ j)$임을 알 수 있고 따라서,

$$\sum_{i\le k'<k} w(k')d(k',k) + \sum_{k\le k'\le j} w(k')d(k,k') \le \sum_{i\le k'<k-1} w(k')d(k',k-1) + \sum_{k-1\le k'\le j} w(k')d(k-1,k')$$

$$\sum_{i\le k'<k} w(k')(d(k',k)-d(k',k-1)) \le \sum_{k-1\le k'\le j} w(k')(d(k-1,k')-d(k,k'))$$이 되고

$$\sum_{i\le k'<k} w(k') \le \sum_{k\le k'\le j} w(k')$$임을 알 수 있다. 이와 마찬가지로 $cost(i,\ k,\ j)\le cost(i,\ k+1,\ j)$이 되어야 하므로,

$$\sum_{i\le k'<k} w(k')d(k',k) + \sum_{k\le k'\le j} w(k')d(k,k') \le \sum_{i\le k'<k+1} w(k')d(k',k+1) + \sum_{k+1\le k'\le j} w(k')d(k+1,k')$$

$$\sum_{k\le k'\le j} w(k')(d(k,k')-d(k+1,k')) \le \sum_{i\le k'<k} w(k')(d(k',k+1)-d(k',k))$$이 되고

$$\sum_{i\le k'<k} w(k') \ge \sum_{k\le k'\le j} w(k')$$이 된다. 그런데 여기서 $k\le z<j$인 어느 정수 z가 프락시 일 때의 서비스 비용을 보면,

$\sum_{i\le k'\le z} w(k') \ge \sum_{z<k'\le j} w(k')$ 이 되고 ($\because z\ge k$) 따라서

$$\sum_{i\le k'\le z} w(k')d(z,z+1) \ge \sum_{z<k'\le j} w(k')d(z,z+1)$$

$$\sum_{i\le k'\le z} w(k')(d(k',z+1)-d(k',z)) \ge \sum_{z<k'\le j} w(k')(d(z,k')-d(z+1,k'))$$ 따라서

$cost(i,\ z+1,\ j)\ge cost(i,\ z,\ j)$, $k\le z<j$ 임을 알 수 있다. 그러므로 k가 노드 i,j상에서 $\sum_{i\le k'\le k} w(k') \ge \sum_{k+1\le k'\le j} w(k')$을 만족하는 가장 인덱스가 작은 노드가 되고 동시에 $opt_loc(i,j)$이 된다. ■

```
Algorithm 1: linear
Ts : 주어진 임계값, last : 주어진 N의 마지막 index 값
/** 웹서버가 서비스하는 부분 설정 **/
 i=0, web_cost=0
 while (web_cost< Ts)
   i ++
   web_cost= web_cost+(w(i)×d(0,i))
 endwhile
 start=i
/** Proxy 위치 찾는 함수 호출 **/
 Proxy (start, last)
```

<Algorithm 1>은 N개의 노드로 구성된 선형구조의 네트워크에서 임계값 Ts를 만족하는 웹 프락시의 위치와 개수를 찾는 알고리즘이다. 서비스 비용이 임계값을 넘지 않게 그룹을 지을 때 미리 지정된 서버가 서비스를 하는 그룹과 임의의 프락시에 의해 서비스를 받게 되는 그룹, 두 종류가 있다. <그림 1> 왼쪽의 점선으로 묶여있는 그룹은 웹서버가 주어진 임계값을 넘지 않으면서 서비스 할 수 있는 곳까지의 범위를 찾아야

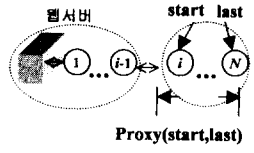

그림 1: Algorithm 1의 구조

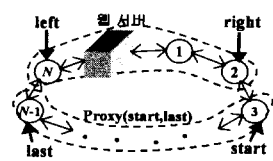

그림 2: Algorithm 2의 구조

되는 부분이고, 그 나머지 범위에 대해서는 $Proxy(start,\ last)$ 함수를 이용하여 proxy의 위치와 개수 및 각 proxy가 서비스하는 영역을 찾는다. 노드 i의 비용은 $w(i)\times d(0,i)$로 나타내며, 누적된 비용의 합이 임계값보다 크면, 웹서버가 서비스할 영역을 설정하게 되고, 그런 다음 나머지 부분에서 프락시의 위치를 찾는 함수를 호출한다. $Proxy(start,\ last)$에서 $W(left, p-1)$는 $left$노드에서 $p-1$노드까지의 가중치의 합을 구하는 함수이다. 즉, $W(i,j)=\sum_{x=i}^{j} w(x)$ (단, $i<j$)로 나타낼 수 있다.

```
Proxy (start, last)
 k=1,  left= right= start
while (left≤ last)
 p=left
 do {
   right ++
   while (W(left, p-1)< W(p+1, right))  p ++  endwhile
   cost_left, p, right = Left_cost(left, p, right)
                       + Right_cost(left, p, right)
   pre_p=p
 }while ((cost_left, p, right< Ts) &&((right+1)≤ last))
 if ((right+1)> last)  then proxy[k ++]=p
 else  proxy[k ++]=pre_p  endif
 left= right+1,  right= left
endwhile
```

정리 1의 특성을 이용하여 가중치의 합에 대한 크기가 바뀌는 위치에 있는 노드 p를 찾는다. 그런 후, $left$와 $right$사이에서 p가 프락시일 때의 전체 비용을 임계값과 비교하기 위해 함수 $Left_cost(left, p, right)$와 함수 $Right_cost(left, p, right)$의 합을 변수 $cost_{left, p, right}$에 저장한다. 함수 $Left_cost(left, p, right)$는 $left$노드와 $right$노드 사이에서 p가 프락시일 때, p에게 서비스 받는 왼쪽 노드들의 비용의 합을, $Right_cost(left, p, right)$는 p에게 서비스 받는 오른쪽 노드들의 비용의 합을 말한다. 즉,

$Left_cost(left, p, right) = \sum_{x=left}^{p-1}(w(x)\times d(x,p))$와

$Right\_cost(left, p, right) = \sum_{x=p+1}^{right}(w(x)\times d(p,x))$ (단, $left\le p\le$

right)으로 계산된다. 이렇게 계산된 변수 $cost_{left,\ p,\ right}$의 값이 임계값보다 큰 경우, 프락시의 위치는 이전에 고려되어졌던 프락시의 위치가 저장된 *pre_p* 번째 노드가 된다. 그리고, 맨 마지막 그룹에서는 프락시가 서비스하는 노드들의 비용이 임계값보다 작을지라도 고려할 노드가 주어진 노드 중에 가장 마지막 인덱스를 갖는다면 현재 *p*의 위치에 프락시를 배치해야 한다. 프락시의 위치를 찾으면 변수 *left*와 *right*를 업데이트하고, 이와 같은 과정을 *left*가 함수 *Proxy*(*start, last*)의 매개변수 *last* 보다 작을 때까지 반복하면 프락시가 배치된다.

3. 고리 구조 (ring topology)

고리 구조의 네트워크 모델은 웹서버와 주어진 *N*개의 노드들이 원형으로 연결되어 있는 것을 제외하고 선형구조와 동일하다. 선형구조에서와 마찬가지로 먼저 웹서버가 서비스하는 부분(*left*노드부터 *right*노드까지)을 정하고 이 영역의 바깥노드인 *start* 노드부터 *last* 노드까지는 선형구조에서 사용한 *Proxy*(*start, last*)를 호출해서 프락시의 위치를 구한다(<그림 2> 참조). 웹서버가 서비스하는 부분을 정하는 과정에서 그 부분의 비용이 주어진 임계값보다 적은 비용을 가져야함과 동시에 그 부분에 포함되지 않은 나머지 부분의 노드들의 가중치의 합인 *W*(*start, last*)가 최소값을 가져야 필요한 프락시의 개수를 최소화할 수 있게 된다.

<Algorithm 2>는 고리 구조에서 웹 프락시의 위치와 개수를 찾는 알고리즘으로 크게 웹서버가 서비스하는 범위를 설정하는 부분[module 1, 2, 3, 4]과 나머지 프락시의 위치를 찾는 부분으로 이루어진다. 먼저 [module1]에서는 서버를 기준으로 오른쪽에 있는 노드들의 비용의 합(*Right_cost*(*left*, 0, *right*))이 주어진 임계값을 넘지 않으면서 서비스할 수 있는 곳까지의 범위를 찾아서 변수 *right*의 최대값을 구한다. 그 결과 변수 *start*의 값을 얻게 된다. [module2]에서 왼쪽으로 *left* 인덱스를 감소시켜가면서 웹서버를 중심으로 *cost* (*cost* = *Right_cost*(*left*, 0, *right*) + *Left_cost*(*left*, 0, *right*))가 임계값을 넘지 않으면서 서비스할 수 있는 곳까지의 범위를 찾고, 이 결과로 변수 *last*의 값을 구할 수 있다. 이렇게 웹서버가 서비스하는 부분을 설정한 후, 나머지 노드들의 가중치의 합은 [module3]에서 함수 *W*(*right*+1, *left*−1)를 이용해서 구하고 현재 최소 가중치의 합을 저장해 둔다. [module4]에서는 변수 *right*값을 감소시키면서 임계값 한도 내에서 왼쪽노드를 최대한 포함하는 새로운 그룹을 만든다. 새로운 그룹을 만들 때마다 그 그룹의 가중치들의 합을 구하여 저장된 최소 가중치들의 합과 비교하는 과정을 *right*가 웹서버가 될 때까지 반복해준다. [module 4]까지를 마치면 웹서버가 서비스하는 부분이 주어진 임계값을 만족하면서 바깥부분의 노드들의 가중치의 합이 최소값을 가지는 범위의 인덱스 *start, last*를 얻는다. 이를 선형구조에서 사용한 함수 *Proxy*(*start, last*)에 적용하여 나머지 프락시들의 위치를 구하게 된다.

4. 결론

본 논문에서는 양방향 통신이 가능한 인터넷에서 선형 구조와 고리 구조를 갖는 웹서버 시스템의 모든 프락시들이 주어진 임계시간을 초과하지 않으면서 클라이언트의 요구를 처리하는데 필요한 프락시의 최적 개수와 위치를 구하는 방법에 대한 알고리즘을 제안했다. 이는 시스템이 웹서버의 최적 개수를 구하고 그 프락시들을 적절한 곳에 배치함으로서 제한된 시간 안에 클라이언트의 요구를 보장해 주고, 시스템을 구축하고 유지하는 비용을 최소화하는데 그 의의가 있다.

```
Algorithm 2 : ring
/** 웹서버가 서비스하는 부분 설정 **/
/* module 1 : 오른쪽으로 그룹짓기 */
right = 1, left = 0
while (Right_cost (left,0, right)< Ts)  right ++   endwhile
right ++, start = right + 1
/* module 2 : 왼쪽으로 그룹 짓기 */
left = N
cost = Right_cost(left,0, right) + Left_cost(left,0, right)
while ( cost < Ts )
  left ++
  cost = Right_cost(left,0, right) + Left_cost(left,0, right)
endwhile
left = (left+1) mod (N+1), last = left − 1
/* module 3 : 그룹 바깥의 가중치들의 합 */
wgt = W(right+1, left−1), min_weight = wgt
/* module 4 : 새로운 그룹 만들기 */
while (right >= 1)
  right ++
  cost -- w(right+1)×d(0, right+1), wgt -- w(right+1)
  while (cost< Ts)
    left --,   cost += w(left)×d(left,0), wgt += w(left)
  endwhile
  left ++,   wgt = wgt − w(left−1)
 if (min_weight > wgt) then
   start = right + 1, last = left − 1, min_weight = wgt
 endif
endwhile
/** 나머지 proxy위치 찾는 함수 호출**/
Proxy(start, last)
```

5. 참고 문헌

[1] N. Yeager and R. McGrath, Web Server Technology, Morgan Kaufman, 1996

[2] M. Baentsch, L. Baum, G. Molters. S. Rothkugel and P. Sturm, "World Wide Web Caching: The Application-Level View of the Internet." IEEE Communications Magazine, Vol.35, No.6, June 1997.

[3] J. Choi, H. Chung, S. Lee, and B. Moon, "Optimal Number and Placement of Web Proxies in the Internet : The Linear Topology", 1999 컴퓨터시스템연구회 추계 학술발표회, page 51~59, September 1999.

[4] B. Li, M. J. Golin and G. F. Italiano and X. Deng, "On the Optimal Placement of Web Proxies in the Internet: Linear Topology," In the 8th IFIP Conference on the High Performance Networking (HPN'98), Vienna, Austria, September 1998.

[5] B. Li, M. J. Golin and G. F. Italiano and X. Deng and K. Sohraby, "On the Optimal Placement of Web Proxies in the Internet," In IEEE InfoComm 1999, pages 1282-1290, 1999.

[6] L. Chen, A, Arora, and H. Choi, "Efficient Optimal Algorithms for Locating web Proxies in Linear and Ring Networks", submit to journal publication.