인터넷상에서 통신 지연 임계값을 보장하는 웹 프락시의 최적 개수와 위치 : 양방향 트리 구조

박숙영[0], 김영남, 이상규, 문봉희
숙명여자대학교 컴퓨터과학과
{ sookyoung[0], ynkim, sanglee, moon }@cs.sookmyung.ac.kr

Optimal Number and Placement of Web Proxies in the Internet with Threshold Latency: The Bidirectional Tree Topology

Sook-Young Park, Young-Nam Kim, Sang-Kyu Lee and Bong-Hee Moon
Dept. of Computer Science, Sookmyung Women's University

요 약

인터넷 사용자의 증가와 그에 따른 WWW(World Wide Web)상에서 발생하는 통신지연의 해결책의 한 방법으로 웹 캐싱 방법이 고려되어 왔다. 이와 같은 웹 프락시는 인터넷 상에서의 위치에 따라 그 효용이 달라지게 되는데, 이와 관련하여 [5]에서는 인터넷상에서 프락시가 될 수 있는 잠재노드들이 트리 구조를 이룰 때 전체적인 접근 시간을 줄이기 위해 미리 정해진 개수(M)의 프락시의 위치를 구하는 방법에 대해 설명하고 있다. 하지만 서버 관리자의 측면에서는 서버를 구축하고 유지하는데 드는 비용과 직결되는 요소인 프락시의 개수를 정하는 것도 중요한 문제이다. 본 논문에서는 인터넷상에서 프락시가 될 수 있는 잠재노드들이 양방향 트리 구조를 이룰 때 서비스 노드들 각각이 정해진 임계값을 보장하기 위해 필요한 최소한 프락시의 개수와 그의 위치를 동시에 찾아내는 알고리즘을 제안한다. 이 때 프락시의 위치는 서비스 그룹 내에서 아무런 제약 없이 임의의 위치에 올 수 있는 경우를 고려한다.

1. 서 론

인터넷 사용자의 증가로 네트워크 상의 통신양은 급격한 증가를 보여왔다[1]. 이러한 현상은 WWW의 대중화로 더욱 두드러졌는데 클라이언트로부터 요청되는 데이터 양이 네트워크의 통신한계량을 초월함으로 인해 요청에 대한 응답의 지연을 일으켰다. 이는 특히 서버나 네트워크의 혼잡(congestion) 정도, 부적절한 대역폭(bandwidth)을 가지는 링크, 그리고 전달 지연(propagation delay)등으로 인해 야기되는데 이러한 통신지체(latency)을 줄이는 가장 효율적인 방법으로 클라이언트와 서버사이에 캐싱을 위한 웹 프락시 서버를 따로 두어 서버의 부하를 줄이는 효과를 내는 방법이 고려되고 있다[2, 3, 4, 5]. 이에 본 논문은 트리 구조(tree 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 단방향 트리구조에서 본 논문과 같은 문제를 다루었으나 프락시의 위치가 항상 서브트리의 루트 노드라는 제약을 주었으므로 본 논문과는 다르거나 더 단순화된 문제들이다.

2. 네트워크 모델 및 문제 정의

프락시가 될 수 있는 잠재 노드들이 양방향 통신을 지원하는 트리 구조(tree topology)를 이루고 N개의 노드로 이루어진 트리에서 임의의 노드를 i라 하면 ($1 \le i \le N$), 각각의 노드 i는 특정시간 동안에 자신에게서 발생되는 통신 요청양을 나타내는 양의 실수인 가중치 $p(i)$를 갖는다. 각각의 링크(i, j)에도 그 링크를 통해 일정양의 데이터를 전달하는데 드는 통신비용(latency)을 나타내는 양의 실수 $dist(i,j)$가 부여되는데 각 링크는 양방향 통신이 가능하여 $dist(i,j) = dist(j,i)$ 이고 임의의 세 노드 i, k, j에 대해 ($i \le k \le j$), $dist(i,k) + dist(k,j) = dist(i,j)$라고 가정한다. 만약 $i = j$이면 $dist(i,j) = 0$이다. 노드 j를 루트로 하는 서브트리(ST_j)에서 서브트리 ST_j 안의 모든 노드들에게 서비스를 제공하는 프락시 서버의 위치가 노드 r이라 할 때 이 서브트리에서 발생되는 통신비용을 $cost(i, r)$로 표기한다. 이는 노드 i와 노드 i의 자식 노드들이 노드 r로부터 서비스를 받을 때 그 그룹에서 발생되는 통신에 대한 비용으로 노드 r을 제외한 다른 노드들의 각각의 통신양에 노드 r로부터의 거리에 따른 비용을 곱한 값의 합으로 계산될 수 있다.

<식1> $$cost(i, r) = \sum_{j \in ST_i} (dist(r, j) \cdot p(j))$$

주어진 트리에서 최소의 프락시 개수로 N개의 모든 노드에 서비스를 제공하려면, 트리를 필요한 최소개수의 서브트리로 분할하고 각 서브트리 그룹에 있어서 통신 임계값(Ts)을 넘지 않으면서 최소 서비스 비용을 갖는 노드가 프락시가 되어야 한다. 루트노드가 s인 서브트리에서 이러한 조건을 만족하면서 프락시 노드가 될 수 있

는 최적의 위치를 $opt_loc(s)$이라 표현하자. 이는 서브트리안의 모든 노드 중에서 가장 적은 통신비용을 갖는 노드를 일컫는다. 이때 각각의 노드에 대한 서비스 비용을 모두 계산해보면 $opt_loc(s)$의 위치를 찾을 수는 있으나, 시간복잡도(time complexity)가 너무 크므로 본 논문에서는 시간복잡도를 줄이면서 $opt_loc(s)$을 찾기 위해 노드의 가중치만을 이용하여 $opt_loc(s)$의 위치 계산이 가능한 방법을 소개한다. 이를 위해 비용변환점이라는 것을 사용하는데 그 정의는 다음과 같다. 서브트리 ST_s의 임의의 노드 r의 m개의 이웃노드들을 $r^1, \cdots, r^k, \cdots, r^m$ ($1 \le m \le N$)이라 할 때, 링크(r, r^k)를 삭제하면 서브트리 ST_s는 두 개의 컴포넌트로 분리되어지는데, 이때 노드 r이 포함되어 있는 컴포넌트를 $C_s(r)$, r^k가 포함되어 있는 컴포넌트를 $C_s(r^k)$라고 표현하자. $C_s(r)$에 포함되는 노드들의 통신 가중치의 합을 $p(C_s(r)) = \sum_{x \in C_s(r)} p(x)$라 하고 $C_s(r^k)$에 속한 노드들의 통신 가중치의 합을 $p(C_s(r^k)) = \sum_{y \in C_s(r^k)} p(y)$라고 표현하면, 그 그룹의 비용변환점을 자신의 모든 이웃노드 r^k에 대해 $p(C_s(r)) \ge p(C_s(r^k))$가 성립하는 노드 r이라 하자[그림 1].

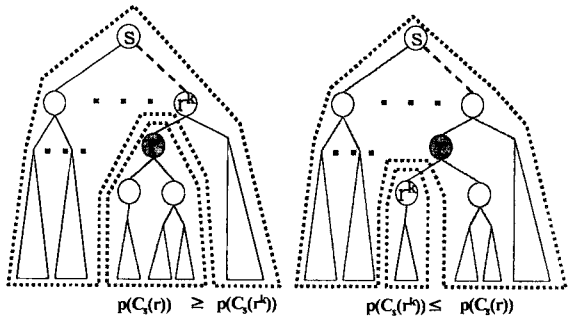

그림 1. 노드 r이 비용변환점이 되는 경우

정리1: 루트 노드가 s이고, 노드 $1, \cdots, i$ (단, $1 \le i \le N$)로 이루어진 서브트리에서 비용변환점이 최소 서비스 비용을 갖는 프락시의 위치인 $opt_loc(s)$이 된다.

증명:
노드 r이 비용변환점일 때, $opt_loc(s)$이 노드 r이 아닌 어떤 노드 j라고 가정해보자.

노드 r이 비용변환점이므로 모든 이웃노드들에 대해 $p(C_s(r)) \ge p(C_s(r^k))$이 성립하고, 반면 $opt_loc(s)$인 노드 j가 프락시일 때 서비스 비용이 가장 작으므로 $cost(s, r) \ge cost(s, j)$ 임을 알 수 있다. $dist$함수를 확장하여 $dist(i, j)$를 노드 i와 노드 j사이의 최단경로 상에 있는 링크 통신비용의 합이라 하면 노드 r과 노드 j의 컴포넌트들 사이의 관계는 다음의 세 가지 경우가 나타날 수 있다[그림 2].

① $C_s(r) \not\subset C_s(j)$ and $C_s(j) \not\subset C_s(r)$
② $C_s(r) \subset C_s(j)$
③ $C_s(j) \subset C_s(r)$

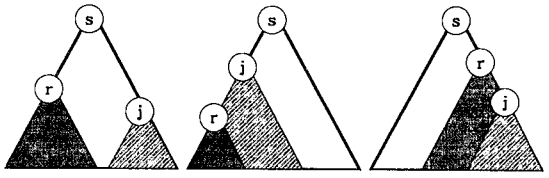

그림 2. 노드 r과 노드 j의 컴포넌트들 사이의 관계

이때 노드 r과 노드 j를 포함하는 컴포넌트들의 $cost$값은 다음과 같고,

$$\sum_{i \in C_s(j), i \notin C_s(r)} p(i) \cdot dist(j,i) + \sum_{i \in ST_r, i \notin C_s(j), i \notin C_s(r)} p(i) \cdot dist(j,i) + \sum_{i \in C_s(j), i \notin C_s(j)} p(i) \cdot dist(j,i) + \sum_{i \in C_s(j), i \in C_s(r)} p(i) \cdot dist(j,i) \le \sum_{i \in C_s(r), i \notin C_s(j)} p(i) \cdot dist(r,i) + \sum_{i \in ST_r, i \notin C_s(r), i \notin C_s(j)} p(i) \cdot dist(r,i) + \sum_{i \in C_s(j), i \notin C_s(r)} p(i) \cdot dist(r,i) + \sum_{i \in C_s(j), i \in C_s(r)} p(n_i) \cdot dist(r,i)$$

이를 ①의 경우에 적용하면,

$$\sum_{i \in C_s(j), i \notin C_s(r)} p(i) \cdot dist(j,i) + \sum_{i \in ST_r, i \notin C_s(j), i \notin C_s(r)} p(i) \cdot dist(j,i) + \sum_{i \in C_s(r), i \notin C_s(j)} p(i) \cdot dist(j,i) \le \sum_{i \in C_s(r), i \notin C_s(j)} p(i) \cdot dist(r,i) + \sum_{i \in ST_r, i \notin C_s(j), i \notin C_s(r)} p(i) \cdot dist(r,i) + \sum_{i \in C_s(j), i \notin C_s(r)} p(i) \cdot dist(r,i)$$

이고 이를 간단히 정리하면

$$\sum_{i \in C_s(r), i \notin C_s(j)} p(i) \cdot dist(j,r) \le \sum_{i \in ST_r, i \notin C_s(j), i \in C_s(r)} p(i) \cdot (dist(r,i) - dist(j,i)) + \sum_{i \in C_s(j), i \notin C_s(r)} p(i) \cdot dist(r,j)$$

이 되므로, 노드 r에서 노드 i까지의 경로에서와 노드 j에서 노드 i까지의 경로에서 동시에 포함되어지는 경로를 노드 f ($f \in ST_r, f \notin C_s(j), f \notin C_s(r)$)에서 노드 i까지라고 할 때,

$$\sum_{i \in C_s(r), i \notin C_s(j)} p(i) \cdot dist(j,r) \le \sum_{i \in ST_r, i \notin C_s(j), i \in C_s(r)} p(i) \cdot \{(dist(r,f) + dist(f,i)) - (dist(j,f) + dist(f,i))\} + \sum_{i \in C_s(j), i \notin C_s(r)} p(i) \cdot dist(r,j)$$

이 되고 이를 정리하면,

$$\sum_{i \in C_s(r)} p(i) \le \sum_{n_i \in ST, \text{and } n_i \notin C_s(n_i) \text{and } n_i \notin C_s(n_i)} p(n_i) + \sum_{n_i \in C_s(n_i)} p(n_i)$$

이고 이는 $p(C_s(r)) \le p(C_s(r^k))$ 이 됨으로 임의의 모든 $C_s(r^k)$에 대해 $p(C_s(r)) \ge p(C_s(r^k))$가 성립한다는 전제

에 모순된다. 그러므로 노드 r과 노드 j는 같은 노드가 된다.

②번과 ③의 경우에도 같은 방법으로 노드 r과 노드 j는 같은 노드가 됨을 알 수 있고, 따라서 루트 노드가 s인 트리 그룹에서 비용변환점이 $opt_loc(s)$이 됨을 알 수 있다. ■

3. 위치설정 알고리즘

주어진 문제의 해결을 위한 프락시의 개수와 위치를 찾는 알고리즘을 설명하기 위해 다음과 같은 표시법을 사용한다. 트리 T의 깊이를 $d(T)$라 하고, 노드 i에 대해 부모 노드는 PA_i, 노드 자식 노드들의 집합을 CH_i라 하며 노드 i를 루트로 하는 서브트리를 ST_i로 표기한다. 그리고 $p(ST_i)$는 ST_i에 있는 노드들의 통신 가중치의 총합을 나타낸다. $Proxy(ST_i)$는 ST_i에서의 프락시의 위치를 나타내고, $GROUP_i$는 알고리즘의 결과로 계산되는 i 번째 프락시가 속한 서비스 그룹의 루트 노드를 나타내며 총 서비스 그룹의 개수는 M으로 표기한다. 또 Ts 는 각각의 프락시 서버가 보장하는 통신 임계값 (Threshold delay cost)이다.

```
Algorithm

/* M ← 0, l ← d(T) */
while (l ≥ 1) {
  for each node i at level l {
    if ( l == 1)  Proxy( ST_i) ← i
    else Proxy( ST_i) ← FindProxy (i, i)
    while cost(i, Proxy( ST_i)) > Ts) {
        GROUP_M ← ST_x // s.t. max p(ST_x) where x∈ CH_i
        M ← M+1
        p(ST_i) = p(ST_i) - p(ST_x)
        T ← T - ST_x
        if (l != 1) Proxy( ST_i) ← FindProxy(i, i)
    endwhile
  endforeach
  l ← l - 1
endwhile

FindProxy(a, b){
  if (a == b)
    if (CH_a == φ)  Proxy( ST_b) ← a
    else if (p( ST_m) > p(ST_b) - p(ST_m))
        FindProxy (Proxy( ST_m), b)
    else  Proxy( ST_b) ← a
  else
    if (p( ST_a) > p(ST_b) - p(ST_a))  Proxy( ST_b) ← a
    else FindProxy (PA_a, b)
}
```

알고리즘은 먼저 트리의 최하위 레벨 $d(T)$에 있는 노드들로부터 레벨 1에 있는 루트까지 레벨 단위로 수행이 되는데, 레벨 l 에서 작업중인 임의의 노드를 i라 할 때 $FindProxy()$ 함수를 이용하여 노드 i를 루트로 하는 서브트리 ST_i의 프락시를 찾는다. 이때 ST_i의 통신비용 $cost(i, Proxy(STi))$를 구하여 그 값이 임계값 Ts보다 크면, 노드 i의 자식 노드들 중 가장 큰 통신 요청양을 갖는 ST_x를 찾아 새로운 그룹으로 정해주고 주어진 트리에서 ST_x를 제외시킨다. 나머지 노드들로 노드 i의 $p(ST_i)$를 재 계산하고, 이를 $cost(i, Proxy(STi))$ 값이 통신 임계값보다 작을 때까지 계속한다. 이러한 과정을 루트까지 수행하게 되었을 때 남는 노드들은 웹 서버로부터 직접 서비스를 받게 된다. 위 알고리즘의 결과로 나오게 되는 프락시의 개수 M이 최소가 됨은 쉽게 알 수 있다.

4. 결론

본 논문에서는 인터넷에서 프락시가 될 수 있는 잠재노드들이 양방향 통신을 지원하는 트리 구조를 가질 때, 모든 프락시들이 주어진 임계값을 초과하지 않으면서 클라이언트의 요구를 처리하는데 필요한 프락시의 최소 개수와 각각의 위치를 구하는 방법에 대한 효율적인 알고리즘을 제안했다. 게다가 본 논문에서는 웹 서버 시스템에서 프락시 서버의 위치가 서비스 영역인 서브트리의 루트노드로 고정되지 않고 그룹 안의 임의의 노드가 모두 프락시 서버가 될 수 있는 시스템 모델을 고려하여 기존 논문에서의 시스템 모델보다 실제적이다. 본 논문이 제안하는 방법은 시스템이 웹 프락시의 최소 개수를 구하고 그 프락시들을 적절한 곳에 배치함으로서 제한된 시간 안에 클라이언트의 요구를 보장해 주고, 시스템 구축 및 유지 비용을 최소화 하는데 그 의의가 있다.

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.