인터넷에서 웹 프락시의 최적 개수와 위치 : 선형 구조와 트리 구조

(Optimal Number and Placement of Web Proxies in the Internet : The Linear & Tree Topology)

최 정 임 † 정 행 은 † 이 상 규 †† 문 봉 희 †††
(Jung-Im Choi) (Haeng-Eun Chung) (Sang-Kyu Lee) (Bong-Hee Moon)

요 약 최근 기하급수적으로 증가하는 WWW(World Wide Web)의 사용으로 인하여 전체적인 네트워크의 성능이 저하되어 서버가 클라이언트에 빠른 서비스를 할 수 없는 경우가 자주 발생되고 있다. 이러한 서비스 지연의 해결방안으로 웹 프락시를 사용하여 서버의 부하와 통신량을 줄이는 방법을 많이 사용하고 있다. 이와 같은 웹 프락시는 인터넷상에서의 위치에 따라 그 효용이 달라지게 되는데, 이와 관련하여 [4]에서는 선형 구조, [5]에서는 트리 구조의 인터넷상에서 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 가장 기본이 되는 선형 구조와 트리 구조에서의 서비스 노드들 각각의 계산된 비용이 정해진 지연 임계값을 만족하기 위해 필요한 최소한의 프락시의 개수와 그의 위치를 동시에 찾아내는 알고리즘을 제안한다.

Abstract With the explosive popularity of the World Wide Web, the low performance of network often leads web clients to wait a long time for web server's response. To resolve this problem, web caching (proxy) has been considered as the most efficient technique for web server to handle this problem. The placement of web proxy is critical to the overall performance, and Li et al. showed the optimal placement of proxies for a web server in the internet with the linear and tree topology when the number of proxies, M, is given[4, 5]. They focused on minimizing the over all access time. However, it is also considerable for target web server to minimize the total number of proxies while each proxy server guarantees not to exceed certain response time for each request from its clients. In this paper, we consider the problem of finding the optimal number and placement of web proxies with the linear and tree topology under the given threshold cost for delay time.

1. 서 론

지난 몇 년간 WWW와 관련된 다양한 도구들의 개발과 그 도구들의 사용이 용이하게 됨에 따라 WWW의 이용자수는 급격히 증가하였다[6]. 그러나 이러한 증가추세를 수용하지 못하는 제한된 자원(resource)으로 인해 전체적으로 네트워크의 성능이 저하되어 클라이언트로부터 전달되는 데이타 요구에 서버가 제때에 응답을 해주지 못하는 경우가 자주 발생되고 있다. 이와 같은 문제는 특히 서버나 네트워크의 혼잡(congestion), 부적절한 대역폭(bandwidth)을 가지는 링크, 그리고 전달지연(propagation delay)등으로 인해 야기되는데 이러한 원인을 제거하기 위해 일반적으로 웹 캐싱기법을 이용한다. 웹 캐싱이란 클라이언트가 서버보다 가까운 위치에서 데이타를 가져올 수 있게 하는 방법으로, 이를 통해 서버의 부하를 줄이고 전달 지연 시간이 줄어드는 효과를 가져온다. 캐싱이 이루어지는 곳으로는 웹 서버

† 학생회원 : 숙명여자대학교 전산학과
choijungim@hanmail.net
piazza43@hanmail.net
†† 정 회 원 : 숙명여자대학교 전산학과 교수
sanglee@sookmyung.ac.kr
††† 종신회원 : 숙명여자대학교 전산학과 교수
moon@sookmyung.ac.kr
논문접수 : 2000년 2월 16일
심사완료 : 2000년 12월 13일

자신과 웹 프락시, 그리고 클라이언트 상에서의 웹 브라우저 등이 있다[1]. 클라이언트 위치에서의 캐싱은 클라이언트의 웹 브라우저에서 이루어지는데, 이는 짧은 시간 동안 한 클라이언트가 동일한 장소로 가는 경우에는 통신량을 줄일 수 있지만 여러 클라이언트가 같은 위치의 정보를 요구하는 경우에는 서로 도움을 주지 못한다[2]. 웹 서버에서 이루어지는 캐싱은 웹 서버가 다른 서버로의 포인터를 가지는 방법[1]으로, 클라이언트의 요구에 local copy fetching을 이용하여 응답을 보내어 요구를 더이상 하진 않지만 잠재적으로 긴 전달 지연이나 혼잡을 완화하지 못한다[3, 7]. 전체의 대기 시간(latency)을 줄이는 가장 효율적인 방법은 웹 프락시를 이용하는 것인데 웹 프락시란 클라이언트와 서버사이에 캐싱을 위해 둔 중간 서버로써 이는 서버의 부하를 줄이는 효과를 가져온다[8]. 그러므로 웹 프락시를 통한 캐싱이 가장 많이 쓰이고 있으며 이에 관한 연구가 활발히 이루어지고 있다[4, 5]. 이에 본 논문은 인터넷 상의 프락시가 될 수 있는 잠재 노드들이 선형 구조(linear topology)와 트리 구조(tree topology)의 네트워크 형태를 이룰 때 주어진 조건을 만족하면서 클라이언트의 요구에 응답할 수 있는 범위 내에서 필요한 최소의 웹 프락시 개수를 구하고, 이들의 알맞은 위치를 찾는 방법을 제시한다(단, 주어진 구조에서 모든 노드들은 가장 가까운 상위 웹 프락시에서 서비스 받는다).

본 논문의 구성은 다음과 같다.

먼저 2절에서는 관련된 연구를 소개하고 본 연구와 비교 설명하였고, 3절에서는 선형 구조 모델, 4절에서는 트리 구조 모델에 대해 각각의 주어진 조건을 만족하기 위한 최소한의 프락시의 개수와 그것들의 위치를 찾는 방법 및 알고리즘을 살펴보고 마지막으로 5절에서 결론을 기술한다.

2. 관련 연구와의 비교

본 논문과 관련된 연구로 인터넷에서 웹 프락시의 개수가 정해져 있을 때 최적의 위치를 찾는 연구[4, 5]가 있는데, 이 논문들에서 진술되었듯이 전체 트래픽과 통신 지연 두 가지 요소를 고려하여 웹 프락시들을 적절한 곳에 위치시킴으로써, 네트워크 자원과 트래픽 형태를 고르게 분포시켜서 네트워크에서 일어나는 전체 통신 지연을 최소화하는데 그 목적이 있다. [4]에서는 프락시가 될 수 있는 잠재 노드들이 간단한 선형 구조 모델일 때를 고려하여 웹 프락시의 개수가 M, 노드의 개수가 N으로 주어졌을 때 $O(N^2M)$의 시간 복잡도(time complexity) 안에 다이나믹 프로그래밍 알고리즘 (Dynamic Programming Algorithm)을 이용하여 문제점을 해결했다. 또 이와 관련하여 같은 문제를 트리 구조에 응용하여 해결해 낸 방법[5]을 제시했다. 이 연구 역시 다이나믹 프로그래밍 문제를 모델링하여 트리 구조에서 $O(N^2M^3)$의 시간 복잡도 안에 M개의 주어진 웹 프락시를 N개의 노드에 가장 적절하게 배치하는 알고리즘을 구하였다. 위의 문제들은 주어진 개수의 웹 프락시를 네트워크에 배치하기 위한 최적의 위치를 찾는 알고리즘을 기술한 반면, 본 논문에서는 하나의 서버로 운영되고 있는 시스템을 확장하게 될 때 각각의 웹 서버에 부과되는 서비스 총량이 정해진 지연 임계값(Ts)을 초과하지 못하게 함으로써 클라이언트에서 어느 한도의 QoS를 제공하는데 필요한 최소한의 웹 프락시의 개수와 그들의 배치를 찾는 방법을 제안한다. 이는 웹 서버의 관리가 클라이언트의 요구를 만족하면서 웹 프락시의 개수를 최적화하여 시스템 구축과 유지비용을 최소화할 수 있다는데 의의를 가지고 있다.

3. 선형 구조 모델

선형 구조(linear topology)에서의 문제 해결을 위한 기본 모델은 다음과 같다. 프락시가 될 수 있는 잠재 노드의 번호를 웹 서버에서부터 순서대로 1, 2, 3, ..., N이라 붙인다. 이 때 큰 숫자일수록 웹 서버에서 먼 노드이다. T_i는 웹 서버에 접속하는 트래픽 중 노드 i를 지나는 트래픽의 비율이고, 각각의 노드는 서버로부터 서비스를 받기 위해서 자신과 서버 사이의 모든 노드를 거쳐야 하기 때문에 $T_1 \geq T_2 \geq \cdots \geq T_{N-1} \geq T_N$이 된다. P_i는 웹 서버와 노드 i 사이의 거리와 비례하는 전달 지연 값이다. 비용은 각 노드에서 누적된 전달 지연 값(P_i)과 그 노드의 트래픽 비율($T_i - T_{i+1}$)의 곱으로 나타낸다. 그러므로 웹 서버의 서비스를 받는 첫 번째 그룹의 경우에 다음과 같은 식을 이용하여 비용을 구한다.

$$P_i \times (T_i - T_{i-1}) \qquad (1)$$

여기서 i는 1부터 식의 값이 임계값을 넘지 않을 동안 계속 증가한다. 이렇게 웹 서버로부터 서비스 받는 영역을 설정한 후에는 남아 있는 노드들의 전달 지연 값은 다음 그룹의 프락시 서버가 될 노드의 전달 지연에 따라 달라져야 한다. <그림 1>에서 하나의 서비스 그룹에 포함된 노드들 중 가장 마지막 노드를 $p-1$이라 하면 노드 p는 다음 서비스 그룹의 프락시 서버가 된다.

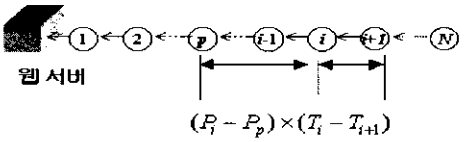


그림 1 노드 i의 비용

<그림 1>에서 현재 고려하는 노드가 i이고, 다음 그룹의 프락시 서버가 노드 $i+2$라 가정하면, 노드 i의 비용은 i와 자신이 서비스 받는 프락시로부터의 전달 지연값(P_i-P_p)과 노드 i 자신의 트래픽 비율인(T_i-T_{i+1})의 곱으로 나타낸다. 이를 이용하면 k번째 그룹의 비용 $cost_k$는 식 (2)와 같이 정의되어 진다.

$$cost_k = \sum_{i=p+1}^{i} (P_i - P_p)\times(T_i - T_{i+1}) \qquad (2)$$

위의 계산된 비용과 임계값을 고려하여 프락시들의 최소 개수와 그 위치를 찾는 두 개의 알고리즘(USS와 UBS)을 아래와 같이 제안한다. 알고리즘에서 N은 주어진 네트워크에서 노드들의 총 개수로, Ts는 주어진 네트워크의 전체 지연 시간이 넘지 않아야 하는 임계값이다. 이 때 알고리즘의 결과로 임계값을 만족시키기 위해 필요한 최소한의 프락시의 개수(M)와 위치에 해당하는 노드의 색인(index)을 동시에 얻을 수 있다.

3.1 USS(Using Sequential Searching) 알고리즘

·위에서 설명한 식 (2)를 이용하여 각 그룹의 비용(변수 cost값)과 임계값을 비교하면서, 현재 그룹이 어느 노드까지 포함할 것인지를 계산하여 다음 그룹의 프락시의 위치를 구할 수 있다.

이 과정을 반복하면, 알고리즘의 결과 값으로 얻게 되는 k값은, 모든 그룹의 비용($cost_k$)이 지연 임계값을 만족시키는 최소한의 서버의 개수가 된다. 한 그룹에서 임계값을 만족시키는지를 계산하기 위해 $O(N)$ 시간 복잡도가 필요하고 이러한 과정을 순서대로 모든 노드를 스캔(순차 검색)하면서 N번 반복해야 하므로 최종 $O(N^2)$의 시간 복잡도가 필요하다.

Algorithm 1 : USS
```
/*  N :  고려할 노드의 개수
    Ts : 임계값
    Pi : 노드 i의 전달 지연 값
    Ti : 노드 i의 트래픽 비율
    k :  프락시의 개수
    PROXYi : i 번째 프락시의 위치
    cost : 노드나 그룹의 통신 비용 */
USS(프락시)
{
   PROXY0← 0 /* 0번째 프락시는
                웹 서버를 의미 */
   Tn+1← 0  /* 더미(dummy) 노드 */
   k← 0      /* 프락시의 개수(count) */
   i←PROXYk+1
   cost ← TPROXYk
   while (i≦N+1)
     cost ←
        Σ_{i=PROXYk+1}^{i} (Tj − Tj+1)×(Pj − P_PROXYk)
     if (cost ≧Ts)
        PROXYk+1 ← i−1
        k ← k+1
     endif
     i ← i+1
  endwhile
}
```

3.2 UBS(Using Binary Searching) 알고리즘

UBS의 기본 아이디어는 이진 탐색 방법을 이용하는 것을 제외하고는 USS와 거의 비슷하다. UBS 알고리즘의 시간 복잡도는 $O(MNlogN)$임을 알 수 있고, 이 시간 복잡도는 결과값으로 나오는 웹 프락시의 개수 M에 따라 그 값이 크게 달라질 수 있다. 즉, 주어진 지연 임계값에 비해 네트워크의 지연 시간이 너무 커서 웹 프락시의 개수가 거의 노드의 개수(N)만큼 필요한 경우에 시간 복잡도는 $O(N^2logN)$으로, 그 반대의 경우인 임계값이 크거나 네트워크의 자체가 거의 없는 경우에는 웹 프락시의 개수가 constant에 가깝다고 보고 시간 복잡도를 $O(MlogN)$으로 나타낼 수 있다. 따라서 주어진 네트워크의 지연 시간과 임계값을 고려하여, 필요한 웹 프락시의 개수를 대략 추정하면 두 알고리즘 중에 어떤 것을 이용하는 것이 더 효과적인지를 미리 판단하여 사용할 수 있다.

Algorithm 2 : UBS
```
/*  N,  Ts,  Pi,  Ti,  PROXYi,  cost,  k
        값은 Algorithm 1(USS)과 동일 */
UBS(프락시)
{ proxy[0], k ← 0
  low ← 1
  while  (low < high)
     high ← N, cost−T_PROXYk, k←k+1
     PROXY0 ← Findproxy(low, high)
```

```
    low ← PROXY_k + 1
  endwhile
}
FindProxy(low, high)
  /* 프락시 노드 찾는 함수 */
{ mid ← ⌜ (low + high) / 2 ⌝
  while ( low ≦ high )
      cost ← Σ_{j=proxy(k-1)+1}^{mid-1} (T_j − T_{j+1}) × (P_j − P_{PROXY_{k-1}})
    if (cost < Ts) then
          low ← mid + 1
          mid ← ⌜ (low + high) / 2 ⌝
    else if (cost = Ts) then
          return mid - 1
    else
          high ← mid − 1
          mid ← ⌜ (low + high) / 2 ⌝
    endif
  endwhile
  return mid - 1
}
```

앞에서 제시한 알고리즘의 타당성을 설명하기 위해, $L_{i,j}$ 가 i 번째 그룹의 j 번째 노드를 나타낸다고 가정하자.($0 \le i \le M$, $1 \le j \le xi$ 이고, x_i는 i 그룹의 노드 수이다.)

정리 1 : i 번째 그룹에 $L_{i+1,1}$노드를 포함시키는 것은 불가능하다.

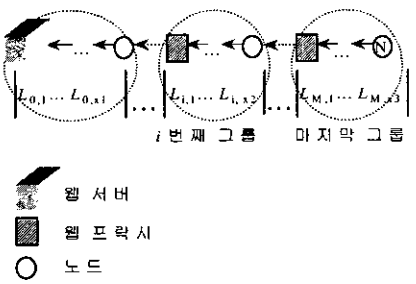


그림 2 주어진 네트워크의 그룹

증명 : $L_{i,1}$에서 $L_{i,xi}$의 사이에 있는 노드들은 i 번째 그룹에 속한다. i 번째 그룹의 전체 비용($cost_i$)은 임계값 Ts보다 작아야 한다. (i번째 그룹의 전체 비용 = $\sum_{j=1}^{x_i} cost_{L_{i,j}}$) 만약 i 번째 그룹의 맨 마지막 노드의 다음 노드를 이 그룹에 포함시킨다면, 식 (3)과 같은 결과 즉 지연 임계값을 만족하지 않기 때문에 주어진 조건에 어긋난다.

$$\sum_{j=1}^{x_i} cost_{L_{i,j}} + cost_{L_{i+1,1}} \ge Ts \qquad (3)$$

그러므로 i 번째 그룹에서 오른쪽 방향에 있는 어떤 노드도 포함시킬 수 없다. ■

정리 2 : 알고리즘 USS와 UBS에 따르면 선형구조 네트워크에서 주어진 트래픽 비율, 전달 지연, 임계값을 가지고 필요한 웹 프락시의 최소 개수인 M을 구할 수 있다.

증명 : 만약 필요한 웹 프락시의 최소 개수가 M이 아니라면, M 보다 작은 값이 결과로 나오기 위해서 둘 또는 그 이상의 그룹이 하나로 합쳐져야 한다. 다시 말해서 한 개 이상의 프락시를 제거하더라도 주어진 조건을 만족해야 한다. <그림 2>로부터 i-1 번째, i 번째, i +1 번째 그룹을 고려하면 아래의 <그림 3>과 같다.

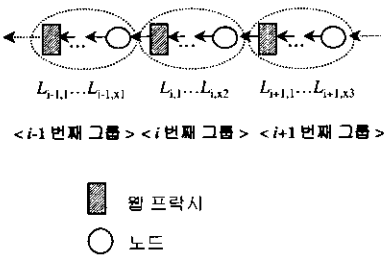


그림 3 그룹 설정

<그림 2>와 <그림 3>에 따르면 두 개 이상의 그룹이 하나로 합쳐지는 경우를 세 가지로 나누어 생각할 수 있다. 먼저 i 번째 그룹이 i+1 번째 그룹을 포함하는 경우와 i 번째 그룹이 i−1 번째 그룹에 포함되는 경우, 그리고 i 번째 그룹의 일부가 i−1 번째 그룹에 포함되고 그 나머지는 i+1 번째 그룹에 포함되는 경우가 있다. 첫 번째 경우에는 i 번째 그룹이 i+1 번째 그룹을 포함한다면, $L_{i+1,1}$에 웹 프락시를 놓을 필요가 없게 된다. 그러나 지연 임계값을 만족해야 한다는 조건을 고려하면 i 번째 그룹과 i+1 번째 그룹을 하나의 그룹으로 합치는 것은 불가능하다. 정리 1에 의해 i 번째 그룹은 $L_{i+1,1}$을 포함할 수 없기 때문이다. 두 번째 경우에서 만약 i 번째 그룹이 i−1 번째 그룹에 포함될 수 있다면

$L_{i,1}$의 노드에 웹 프락시를 놓을 필요가 없게 된다. 그러나 $i-1$ 번째 그룹 역시 정리 1에 의해 현재의 그룹($i-1$ 번째 그룹)에 $L_{i,1}$의 노드를 포함시킬 수 없다. 세 번째 경우도 앞의 두 가지 경우로 미루어 볼 때, 불가능함을 쉽게 알 수 있다. 이와 같이 세 가지 경우에서 모두 웹 프락시의 개수를 더 이상 줄일 수 없으므로, M이 주어진 네트워크에서 지연 임계값을 만족시키는 최소의 웹 프락시 개수이다. ■

4. 트리 구조 모델

프락시가 될 수 있는 잠재 노드들이 트리 구조(tree topology)를 이룰 때의 문제 해결 방안을 위한 기본 모델은 <그림 4>와 같다.

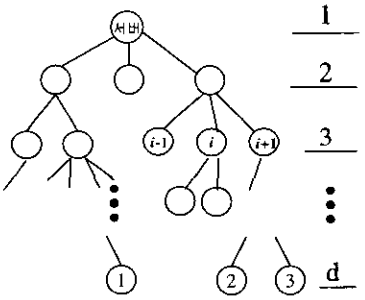


그림 4 트리 구조 모델

N개의 노드로 이루어진 트리에서 임의의 노드를 i라 하면 (이 때, $1 \le i \le N$), 노드 i는 서버로부터의 거리 차이 때문에 생기는 전달 지연 (P_i) 값과 자신과 자신의 하위의 노드들 때문에 생기는 트래픽에 대한 비율 (T_i) 값을 갖고 있다. 다시 말해 T_i는 노드 i를 통해 웹 서버에 접근하는 전체 트래픽의 비율을 뜻하므로, T_i 값은 노드 i의 부모 노드의 트래픽 비율 값보다 항상 작거나 같다. 전달 지연 P_i는 웹 서버에서 노드 i까지의 거리에 비례하므로 트리 모형에서는 루트에서 리프 노드로 갈수록 그 값은 커지게 된다. 각 노드의 비용은 그 노드가 리프 노드인지 아닌지에 따라 다르게 계산된다. 리프 노드인 경우의 비용은 그 노드의 트래픽 비율로만 나타나지만, 리프 노드가 아닌 노드의 비용은 그 노드의 트래픽 비율뿐만 아니라 자신의 하위 노드들의 비용도 함께 고려해야 한다.

주어진 트리 구조 모델에서 임의의 노드 i에 대해 부모 노드는 PA_i라 하고, 노드 i의 자식 노드들의 집합을 CH_i라 하며 노드 i를 루트로 하는 서브 트리에서 i를 제외한 노드의 집합은 SU_i로 정의하자. 노드 i의 비용 중 자신의 트래픽 비율만의 비용은 $cost(i)=T_i-\sum_{j\in CH_i} T_j$ 이고, 노드 i를 서브 트리의 루트로 했을 때, 하위 노드 $k(k\in SU_i)$의 비용을 $cost(i, k)$라고 하면 이 값은 식 (4)와 같이 정의된다.

$$cost(i,k)=(P_k-P_i)\times(T_k)+\sum_{s\in CH_k} cost(i,s) \qquad (4)$$

예를 들면 <그림 5>에서 노드 i를 서브 트리의 루트로 했을 때 노드 6의 비용은 $cost(9,6)=(P_6-P_9)\times(T_6)$이 되고, 리프 노드가 아닌 노드 4의 비용은 $cost(9,4)=(P_4-P_9)\times(T_4)+cost(9,1)+cost(9,2)+cost(9,3)$이 된다.

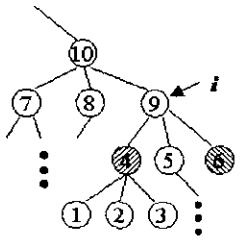


그림 5 트리 구조의 한 부분

주어진 문제의 해결을 위한 프락시의 개수와 위치를 찾는 알고리즘을 설명하기 위해 다음과 같은 표시법을 사용한다. V는 주어진 트리 구조의 노드들의 집합을 나타내고, 트리의 깊이를 d라 한다. $PROXY_i$는 알고리즘의 결과로 계산되는 i 번째 프락시의 위치를 나타내고 총 프락시의 개수를 M으로 표기한다. 또 T_S는 각각의 프락시 서버가 보장하는 지연 임계값(Threshold delay cost)이다. 지연 임계값 T_S는 최대 노드 비용 $\max_{i\in V}\{cost(i)\}$ 보다 항상 크거나 같다고 가정한다.

알고리즘은 먼저 트리의 최하위 레벨 d에 있는 노드들로부터 레벨 1에 있는 루트까지 레벨 단위로 수행이 되는데, 레벨 l에서 작업중인 임의의 노드를 i라 할 때 노드 i와 그 서브 트리의 비용 ($cost(i)+\sum_{j\in CH_i} cost(i,k)$)을 구하여 임의의 변수(Algorithm 3에서는 SCh으로 쓰임)에 저장하고 그 값이 임계값 T_S보다 크면, 노드 i의 자식 노드들 중 가장 큰 비용을 갖는 노드 k를 찾아 새로운 프락시로 정해주고, 노드 집합 V에서 노드 k와 그의 서브 트리에 있는 모든 노드들을 제외시킨다. 나머지 노드들로 서브 트리의 비용(SCh)값을 재계산 하고, 위의 과정을 재계산된 서브 트리의 비용(SCh)값이 지연 임계값보다 작을 때까지 계속한다. 이러한 과정을 루트까지 수행하게 된다. Algorithm 3의 라인 1에 있는

while loop이 끝났을 때, 남는 노드들은 웹 서버로부터 직접 서비스를 받게 된다.

Algorithm 3

/* $M \leftarrow 0$, $Total \leftarrow 0$, $SCh \leftarrow 0$, $l \leftarrow d$ */
1: **while** ($l \geq 1$)
2: **for each node** i **at level** l
3: $SCh \leftarrow cost(i) + \sum_{k \in CH_i} cost(i,k)$
4: **while** $SCh > Ts$
5: $Mch \leftarrow j$
such that $\max_{j \in CH_i}\{cost(i,j)\}$
6: $PROXY_M \leftarrow Mch$, $M \leftarrow M+1$
7: $SCh \leftarrow SCh - cost(i, Mch)$
8: $V \leftarrow V - (\{Mch\} \cup SU_{Mch})$
endwhile
endforeach
9: $l \leftarrow l-1$
endwhile

정리 3: 트리 $T[V,E]$와 각 노드 i에 대한 P_i, T_i값과 지연 임계값 T_S가 주어졌을 때, Algorithm 3에 의해 찾아지는 M은 최소값이 되며 각 프락시들의 $cost$는 임계값을 넘지 않는다.

증명 : 노드 i를 루트로 하는 서브 트리를 $ST(i)$라 하고 $ST(i)$의 총비용을 $cost(ST(i)) = cost(i) + \sum_{k \in Ch_i} cost(i,k)$라 하자.

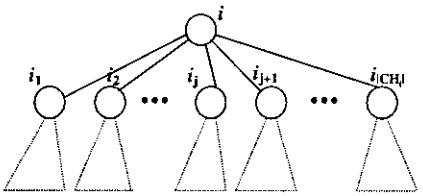


그림 6 서브 트리 ST(i)

<그림 6>에서 현재 고려중인 노드가 i이고 CH_i가 $\{i_1, i_2, \cdots, i_{|CH_i|}\}$일 때, 노드 i의 자식 노드들의 이름이 각각을 루트로 하는 서브 트리의 비용들간의 관계를 $cost(ST(i_1)) \geq cost(ST(i_2)) \geq \cdots \geq cost(ST(i_{|CH_i|}))$가 되게끔 정해졌다고 가정하자. 알고리즘 3에서 6번째 줄의 새로운 프락시에 선정되는 노드 j의 서브 트리의 총 비용은 항상 T_S를 넘지 않음을 볼 수 있다. 따라서 선택된 각각의 프락시가 자신의 클라이언트에게 서비스해야 하는 총비용은 T_S를 넘지 않는다.

Algorithm 3에서 4번째 줄의 while loop이 끝났을 때, 남아 있는 노드 i의 자식 노드들을 $i_{j+1}, i_{j+2}, \cdots, i_{|CH_i|}$라 하면 노드 i를 고려하여 생긴 프락시의 개수는 j개이고 남은 서브 트리의 총 비용은 $cost(i) + \sum_{j<k \leq |CH_i|} cost(i, i_k)$임을 알 수 있다. 이때, j는 $cost(ST(i))$의 값을 임계값 T_S보다 작거나 같게 만들기 위해 삭제되어지는 노드의 최소 개수가 되고 동시에 $cost(ST(i_1)) \geq cost(ST(i_2)) \cdots cost(ST(i_{|CH_i|}))$조건에 의해 노드 i의 $|CH_i| - j$개의 자식 노드로 만들 수 있는 서브 트리의 비용 중 최소값은 $cost(i) + \sum_{j<k \leq |CH_i|} cost(i, i_k)$이 됨을 쉽게 알 수 있다.

따라서 $ST(i)$의 노드들은 $j+1$개 보다 적은 개수의 프락시들로 지연 임계값을 넘지 않으면서 서비스를 받을 수 없으며 이러한 지역적인 최적값들의 선택은 전체의 최적값을 보장함을 볼 수 있다. 따라서, Algorithm 3에서 구한 프락시의 개수 M은 주어진 조건을 만족하는 최소의 개수이다. ■

5. 결 론

본 논문에서는 인터넷에서 선형 구조와 트리 구조 모델을 갖는 웹 서버 시스템에서 모든 프락시들이 주어진 지연 임계값을 초과하지 않으면서 클라이언트의 요구를 처리하는데 필요한 프락시의 최적 개수와 위치를 구하는 방법에 대한 3개의 알고리즘을 제안했다. 정해진 프락시의 개수에 대해 가장 최적의 위치를 찾는 기존 연구와는 달리 본 연구는 시스템이 웹 프락시의 최소 개수를 구하고 그 프락시들을 적절한 곳에 배치함으로서 제한된 시간 안에 클라이언트의 요구를 보장해 주고, 시스템을 구축하고 유지하는 비용을 최소화하는데 그 의의가 있다.

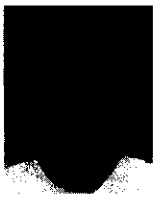
참 고 문 헌

[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] D. Patterson and J. Hennessy, Computer Organization and Design: the Hardware/Software Interface, 2nd edition, Mogran Kaufman, 1997.
[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] M.F. Arlitt and C.L. Willianson, "Internet Web Servers: Wordload Characterization and Performance Implications", IEEE Transactions on Networking, Vol. 5, No. 5, October 1997.
[7] P. Scheuermann, J. Shim and R. Vingralek, "A Case for Delay-Conscious Caching of Web Documents", Computer Networks and ISDN Systems, Vol. 29, No. 8-13, September 1997.
[8] Radhika Malpani, Jacob Lorch, and David Berger, "Making world wide web caching servers cooperate," Proceedings of World Wide Web Conference, 1996.

최 정 임

1999년 숙명여자대학교 전산학과 졸업. 2001년 숙명여자대학교 대학원 컴퓨터과학 전공 졸업. 2001년 ~ 현재 한국 IBM 소프트웨어 사업부. 무선통신 솔루션 정보기술 지원 전문. 관심 분야는 정보통신, 무선 통신



정 행 은

1999년 숙명여자대학교 전산학과 졸업. 2001년 숙명여자대학교 대학원 컴퓨터과학 전공 졸업. 2001년 ~ 현재 SK teletech 소프트웨어 개발 연구원. 관심 분야는 정보통신, 무선 통신

이 상 규
정보과학회논문지 : 정보통신
제 28 권 제 1 호 참조

문 봉 희
정보과학회논문지 : 정보통신
제 28 권 제 1 호 참조