효과적인 Backbone Core Tree(BCT)생성 알고리즘

서현곤[0] 김기형
영남대학교 컴퓨터공학과
hgseo[0]@scc.taegu.ac.kr ,kkim@yu.ac.kr

Efficient Backbone Core Tree Generation Algorithm

Hyun-Gon Suh[0] Ki-Hyung Kim
Dept. of Computer Engineering, YeungNam University

요 약

본 논문에서는 many-to-many IP 멀티캐스팅을 위한 효율적인 Backbone Core Tree(BCT)생성 알고리즘에 대하여 제안한다. 본 논문의 제안기법은 Core Based Tree(CBT)에 기반을 두고 있다. CBT는 공유 트리를 이용하여 멀티캐스트 자료를 전달하기 때문에 Source based Tree에 비하여 각 라우터가 유지해야 하는 상태 정보의 양에 적고, 적용하기 간단하지만, Core 라우터 선택의 어려움과 트래픽이 Core로 집중되는 문제점을 가지고 있다. 이에 대한 보완책으로 Backbone Core Tree기법이 제안되었는데, 본 논문에서는 주어진 네트워크 위상 그래프에서 최소신장 트리를 만들고, 센트로이드를 이용하여 효율적인 BCT를 생성하는 알고리즘을 제안한다.

1. 서 론

IP 멀티캐스트 라우팅 알고리즘은 멀티캐스트 IP 패킷을 전달하기 위해 생성하는 트리(Tree)에 따라 크게 Source based tree, Shared Based tree, Quality of Service(Qos) based tree로 구분할 수 있다.[1]

Source based tree는 자료를 전송하는 소스에서 자료를 받는 목적지까지 최단 거리를 연결하는 트리를 생성하여 멀티캐스트 자료를 전달하기 때문에 다른 멀티캐스트 라우팅 알고리즘보다 지연(delay)시간이 작은 장점이 있지만, 네트워크 내의 모든 멀티캐스트 정보를 유지해야 하는 단점을 가지고 있다. 즉, 활성화된 소스가 S개 있고, 멀티캐스트 그룹에 G개 있다면 멀티캐스트 정보를 유지하기 위해 O(S*G)만큼의 메모리가 필요하다.

Shared based tree는 네트워크 내부에 RP(Rendezvous Point)를 두어 각 소스가 자료를 RP에 전송하면 RP에서 생성되어 있는 공유 패스를 이용하여 자료를 전달하는 것으로 Core Based Tree(CBT)가 대표적인 공유 트리 기법을 이용한다[2]. Source based tree에 비하여 지연시간은 증가하지만 각 라우터에서 멀티캐스트 그룹을 유지하기 위한 정보는 O(S)가 소요됨으로 확정성이 더 뛰어난 장점을 가지고 있다.

Qos based tree는 end-to-end 지연과 대역폭의 효과적인 이용률을 높이기 위해 제안된 것으로 전체 네트워크의 위상 정보와 사용 가능한 링크의 용량(capacity) 등과 같은 추가 적인 정보를 요구하기 때문에 아직 널리 사용되지 않고 있다.

본 논문에서는 many-to-many 멀티캐스트를 위해 공유 트리 기법을 이용하여 CBT을 기반으로 Backbone Core Tree(BCT)를 생성하는 방법을 제안한다. CBT는 many-to-many 멀티캐스팅에서 실제적으로 많이 사용되지만 다음과 같은 문제점을 가지고 있다. 첫째 코아(Core) 라우터의 선택에 어려운 점을 가지고 있다. 매번 새롭게 생성되는 멀티캐스트 그룹에서 최적의 코아 라우터를 선택하기 위해서 그룹의 멤버와 분산 정보와 같은 추가적인 정보가 있어야 한다. 둘째로 코아 라우터로 멀티캐스트 트래픽이 집중하는 현상이 발생한다. CBT에서는 모든 소스는 데이터 패킷을 코아로 보내면 코아가 받아서 공유 트리로 데이터를 재전송 해야 하기 때문에 코아 라우터의 성능과 코아와 연결된 링크의 대역폭이 증가해야 하는 부담을 가지게 된다.

BCT는 CBT의 단점을 보완하기 위해 제안된 것으로, 미리 네트워크 관리자에 의하여 전체 네트워크 중에서 Backbone core tree를 형성하여 데이터 패킷을 전송하는 것으로 CBT에 비해 20~40%의 트리 비용을 절감하는 효과를 가져왔다.[2]

그런데 기존의 연구에서는 효과적인 BCT의 생성에 대한 방법은 제안하고 있지 않기 때문에 본 논문에서는 전체 네트워크 위상에서 최소신장 트리를 생성하여 센트로이드(Centroid)를 찾아 센트로이드와 연결된 링크를 BCT에 추가함으로 최소비용을 가지는 BCT 생성 알고리즘을 제안한다.

본 논문의 전체 구성은 다음과 같다. 2장은 본 논문에서 사용하는 용어와 기본개념을 설명하고, 3장은 BCT생성 알고리즘을 소개하고 4장에서 결론을 내린다.

2. 기본개념

2.1 CBT와 BCT

BCT의 기본적인 개념을 전체 네트워크를 몇 개의 영역으로 분할하여, 각 분할된 영역을 관장하는 하나의 코아를 선택하여 영역내의 멀티캐스팅은 CBT로 처리하고, 영역들 사이의 멀티캐스팅은 각 영역의 코아를 연결한 BCT를 이용하여 처리하는 것이다.[2]

그림 1은 CBT를 나타낸 것으로 5명의 사용자로 구성된 하나의 멀티캐스트 그룹으로 링크의 숫자는 link cost를 나타낸다. 모든 멀티캐스트 패킷은 코아로 전달되면 코아는 CBT를 이용하여 다른 멤버들에게 자료를 전달하게 된다. 그림 1은 B가 코아 라우터로 전체 멀티캐스트 그룹을 연결하는데 link cost가 45임을 나타내고 있다.

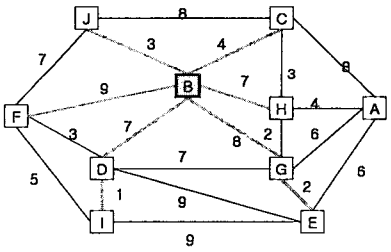

그림 1 CBT(Core Based Tree)

그림 2는 BCT를 나타낸 것으로 네트워크 관리자에 의하여 미리 라우터 B, D, H를 연결하는 링크를 BCT로 설정한 것이다. 그림 1과 같이 5명의 사용자가 같은 그룹의 멤버가 되었을 때 전체 멀티캐스트 그룹을 연결하는데 link cost가 33임을 알 수 있다.

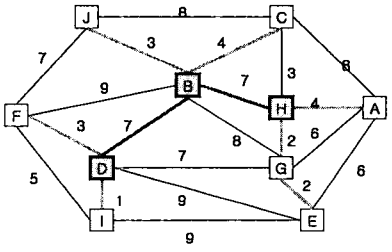

그림 2 BCT(Backbone Core Tree)

그런데, 그림2의 BCT는 그림3처럼 D, G, H를 BCT로 재구성하면 link cost가 32로 더 효과적임 알 수 있다. 또한 라우터 J에 연결된 사용자가 그룹에서 이탈했을 경우 그림 2에서 link cost는 30이지만 그림 3에서는 link cost가 22임을 알 수 있다. 즉, 주어진 네트워크에서 어떻게 BCT를 구성하느냐에 따라서 link cost의 차이가 있음을 알 수 있다.

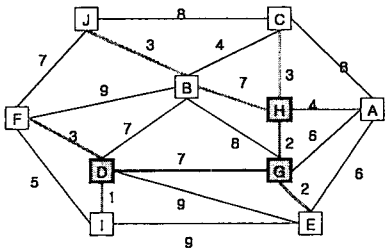

그림 3 BCT의 재구성

그런데 기존의 연구에서는 BCT개념과 CBT보다 효과적임을 소개할 뿐 어떻게 BCT를 구성할 것인지에 대한 상세한 알고리즘은 제시하고 있지 않다. 그래서 본 논문에서는 주어진 네트워크 위상에서 link cost를 최소로 하는 BCT생성 알고리즘을 제안한다.

2.2 센트로이드와 최소신장트리

n개의(n>1) 노드를 가진 트리 T에서 임의의 노드 v를 선택하여 v와 연결되어 있는 에지들을 T로부터 제거하면 T는 k개의 서브트리(subtree) $T_1, T_2, ..., T_k$로 나누어진다. 이때 v의 값 value(v)를 MAX{$|T_1|$, $|T_2|$, ... , $|T_k|$}라 정의하자. 여기서 $|T_i|$는 T_i에 속하는 노드의 수를 나타낸다. 노드 v중에서 value(v)가 가장 작은 노드를 T의 센트로이드(centroid)라 한다. T에서 각 노드 v에 대해 size(v)는 v자신을 포함하고 v를 루트(root)로하는 서브트리의 노드 개수를 의미한다. size(v)를 알면 정리 1과 2에 의하여 O(n)시간에 쉽게 T의 센트로이드를 구할 수 있다.

정리 1. T는 많아야 두 개의 센트로이드를 가질 수 있고 만약 두 개의 센트로이드를 갖는 경우에는 이들은 이웃한다.

증명 : [4] 참조. □

정리 2. (1) T가 두 개의 센트로이드 c와 d를(c가 d의 자식이라 가정) 가질 필요 충분 조건은 n이 짝수이고, size(c)=n/2 이다.

(2)T가 하나의 센트로이드 c만을(c의 자식노드를 d_1, d_2,

... , d_k)가질 필요 충분조건은

$2 \cdot size(c) \geq n+1$

$2 \cdot size(d_i) \leq n-1, (1 \leq i \leq k)$ 이다.

증명 : [4] 참조. □

최소 신장 트리(Minimum Spanning Tree:MST)는 기존의 Prim 알고리즘 또는 Kruskal 알고리즘을 이용하면 쉽게 구할 수 있다.[5] Kruskal 알고리즘은 에지들의 값을 정렬하여 값이 적은 에지를 선택하여 최소 신장 트리를 생성하는 것으로 O(E logE)에 계산할 수 있다. Prim 알고리즘은 임의의 정점 N을 선택하여 N에 연결된 최소 비용 에지 e을 선택하여 이 에지를 MST에 추가하고, 에지 e에 연결된 다른 정점이 M일 경우, M에서 다시 최소 비용 에지를 찾아 MST에 추가하는 방법으로 최소 신장 트리를 생성한다.

3. BCT 생성 알고리즘

주어진 네트워크에서 BCT생성하기 위한 알고리즘은 다음과 같다.

입력 : 전체 네트워크 위상, 서브트리 $|T_i|$의 최소 노드 수 m(단, $m \leq \log n$)

출력 : Backbone Core Tree

[step 1] 주어진 네트워크에 대하여 최소 신장 트리 T_{sp}를 계산한다.

[step 2] T_{sp}를 postorder순으로 방문하여 각 노드 v에 대한 size(v)를 계산한다.

[step 3] T_{sp}에서 센트로이드 c를 구하고,c를 BCT에 추가한다.

[step 4] c를 T_{sp}에서 제거하여 $T_1, T_2, \ldots, T_k$을 생성한다.(센트로이드가 두 개인 경우 둘중에 하나만 선택)

[step 5] 되부름(recursion)방법으로 각 $|T_i|$를 구한다.

[step 6] $|T_i| \geq m$ 만족하는 서브트리 T_i의 루트 노드 d_i를 BCT에 추가한다. (센트로이드 c와 노드 d_i는 하나의 링크로 연결되어 있음)

[step 7] BCT에 포함된 노드를 가진 서브트리 T_i에서 서브트리 T_i의 센트로이드 c'와 상위 센트로이드 c가 인접하지 않은 경우 path(c,c')에 있는 노드를 BCT에 추가한다.

[step 1][step 2]는 기존에 있는 알고리즘을 이용하면 쉽게 해결할 수 있고, [step 3][step 4]는 정리 1과 2를 이용하면 O(n)에 계산이 가능하다. [step 5]에서 되부름 횟수는 O(log n)이다.

[step 6]에서 센트로이드 c의 링크 수가 x개 있다면 O(k)에 실행 가능하다. [step 7]에서는 서브트리 T_i의 센트로이드 c'와 상위 센트로이드 c와 인접하지 않을 경우 path(c,c')에 존재하는 모든 노드를 BCT에 추가한다.

4. 결론

본 논문에서는 many-to-many IP 멀티캐스팅을 위한 BCT 생성 알고리즘을 제안하였다. 주어진 BCT알고리즘은 주어진 전체 네트워크 위상에서 최소신장 트리 T_{sp}를 생성하여 센트로이드를 찾아 센트로이드와 연결된 링크를 BCT에 추가하는 방법을 이용하여 효과적인 BCT을 생성하였다. 센트로이드와 연결된 노드를 BCT에 추가하는 것은 코아 라우터(센트로이드)로 선정된 라우터의 부하 정보, 즉 서브 트리의 노드 수에 따라 결정된다.

앞으로의 과제는 임의의 네트워크 위상에 대하여 최적의 m값을 찾는 방법을 연구하는 것이 과제로 남아 있다.

참고문헌

[1] M. Capan, Survey of Different Multicast Routing Protocols, Croatian Academic and Research Network -CARNet, MIPRO'98

[2] A.J. Ballardie, Core Based Tree Multicast Routing Architecture, RFC2201, Sept. 1997

[3] Seok-Joo Koh, Shin-Gak Kang, Ki-Sjik Park, Enhanced Cores Based Tree for Many-to-Many UP Multicasting, Telecommunication Review Vol.11, N0.3 , pp. 485-493, 2001.5~6

[4] D. Knuth, The art of Programming : Fundamental Algorithm, Addison-Wesley, Vol.1, pp. 386-388, 1968

[5] Robert Sedgewick, Algorithms , Addison-Wesley, Second edition, pp.456-464, 1988