IP 멀티캐스팅을 위한 센트로이드 기반의 백본코아트리 생성 알고리즘

(A Centroid-based Backbone Core Tree Generation Algorithm for IP Multicasting)

서 현 곤 † 김 기 형 ††
(Hyun Gon Suh) (Ki Hyung Kim)

요 약 본 논문에서는 공유 트리에 기반에서 IP 멀티캐스팅을 위한 센트로이드 기반 백본코아트리(Centroid-based Backbone Core Tree: CBCT) 생성 알고리즘을 제안한다. 코아기반트리(Core Based Tree: CBT)는 공유 트리를 이용하여 멀티캐스트 자료를 전달하는 것으로 소스 기반 트리에 비하여 각 라우터가 유지해야 하는 상태 정보의 양이 적고, 적용하기 간단한 장점을 가지고 있지만, 코아 라우터(Core router) 선택이 어렵고, 멀티캐스트 트래픽이 코아로 집중되는 문제점을 가지고 있다. 백본코아트리(Backbone Core Tree: BCT)는 CBT의 단점을 보완하기 위해 제안되었다. BCT는 각 멀티캐스트 그룹마다 특정한 코아 라우터를 선정하지 않는 대신 코아라우터 후보들을 백본코아트리(BCT)로 연결하고, 이 트리를 통하여 코아라우터 후보들이 서로 협동하므로써 위의 두 가지 문제점을 해결한다. 이때 BCT를 어떻게 구성하는가에 따라 멀티캐스트 성능이 크게 변하게 된다. 본 논문에서는 백본코아라우터 후보들 및 이들을 연결하는 BCT를 생성하기 위해 네트워크의 최소 신장 트리와 센트로이드를 이용하는 효율적인 알고리즘 CBCT를 제시한다. 제안된 알고리즘의 성능평가를 위해서 CBT와 CBCT 프로토콜의 성능비교 결과를 보인다.

키워드 : 멀티캐스트 라우팅, 공유기반트리, 백본코아트리, 센트로이드, 최소신장트리

Abstract In this paper, we propose the Centroid-based Backbone Core Tree(CBCT) generation algorithm for the shared tree-based IP multicasting. The proposed algorithm is based on the Core Based Tree(CBT) protocol. Despite the advantages over the source-based trees in terms of scalability, the CBT protocol still has the following limitations; first, the optimal core router selection is very difficult, and second, the multicast traffic is concentrated near a core router. The Backbone Core Tree(BCT) protocol, as an extension of the CBT protocol has been proposed to overcome these limitations of the CBT. Instead of selecting a specific core router for each multicast group, the BCT protocol forms a backbone network of candidate core routers which cooperate with one another to make multicast trees. However, the BCT protocol has not mentioned the way of selecting candidate core routers and how to connect them. The proposed CBCT generation algorithm employs the concepts of the minimum spanning tree and the centroid. For the performance evaluation of the proposed algorithm, we showed the performance comparison results for both of the CBT and CBCT protocols.

Key words : Multicast routing, Shared based tree, Backbone core tree, centroid, minimum spanning tree

1. 서 론

IP 멀티캐스트 라우팅 알고리즘은 멀티캐스트 패킷을 전달하기 위한 트리(Tree)를 생성하는 방법에 따라 크게 소스 기반 트리(Source based tree), 공유 기반 트리(Shared based tree), Qos기반 트리(Qos based tree)로 구분할 수 있다. 소스 기반 트리는 멀티캐스트 패킷을 전송하는 소스에서 멀티캐스트 패킷을 받는 목적지

† 비 회 원 : 대구대학교 정보통신공학부 BK21 교수
hgseo@scc.daegu.ac.kr
†† 종신회원 : 영남대학교 컴퓨터공학과 교수
kkim@yu.ac.kr
논문접수 : 2002년 12월 12일
심사완료 : 2003년 5월 9일

까지 최단 거리로 연결하는 트리를 생성하여 멀티캐스트 패킷을 전달하기 때문에 다른 멀티캐스트 라우팅 알고리즘보다 지연시간이 작은 장점이 있지만, 네트워크 내의 모든 멀티캐스트 정보를 유지해야 하는 단점을 가지고 있다. 즉, 활성화된 소스가 S개 있고, 멀티캐스트 그룹이 G개 있다면 멀티캐스트 정보를 유지하기 위해 O(S*G) 만큼의 메모리가 필요하게 된다. DVMRP(Distance Vector Multicast Routing Protocol)[1], PIM-DM (Protocol Independent Multicast-Dense Mode)[2], MOSPF(Multicast Extension to OSPF)[3], SSM (Source-Specific Multicast)[4] 등이 이에 속한다.

공유 기반 트리는 코아라우터(Core router) 또는 RP (Rendezvous Point)를 두고 각 소스가 유니캐스트(unicast)로 자료를 코아라우터에 전송하면 코아라우터로부터 멀티캐스트 목적지까지 최단거리로 구성되어진 멀티캐스트 트리를 이용하여 멀티캐스트 패킷을 전달한다. 소스 기반 트리에 비하여 멀티캐스트 패킷 전송 지연시간은 증가하지만 각 라우터에서 멀티캐스트 그룹을 유지하기 위한 정보는 O(G)가 소요됨으로 소스 기반 트리보다 효과적이다. 또한 공유 기반 트리는 확장성이 용이하고 구현하기 쉬운 장점을 가지고 있다. PIM-SM (Protocol Independent Multicast-Sparse Mode)[5], SMP(Simple Multicast Protocol)[6], 코아기반트리(Core Based Tree: CBT)[7-8]가 이에 속한다.

Qos 기반 트리는 end-to-end 지연과 대역폭의 효과적인 이용률을 높이기 위해 제안된 것으로 전체 네트워크의 위상 정보와 사용 가능한 링크의 용량(capacity) 등과 같은 추가적인 정보를 요구하기 때문에 아직 널리 사용되지 않고 있다. QoSMIC(Qos Sensitive Multicast Internet Protocol)[9], DDMC(Destination Driven Multicast) [10], QDMR(Qos Dependent Multicast Routing)[11] 등이 이에 속한다.

본 논문에서는 공유 기반 트리 기법중 하나인 CBT를 확장한 센트로이드 기반 백본코아트리(Centroid-based Backbone Core Tree: CBCT)를 생성하는 알고리즘을 제안한다. CBT는 공유 기반 트리의 대표적인 트리생성 프로토콜로서 많이 연구되고 있지만 다음과 같은 문제점을 가지고 있다. 첫째 CBT는 동적인 멀티캐스트 그룹정보로부터 최적의 코아 라우터를 얻어내기 어렵다는 한계를 가지고 있다. CBT에서 사용하는 부트스트랩 메카니즘(Bootstrap mechanism)은 미리정해진 코아 라우터후보들(Candidate core routers) 중에서 단순한 해싱 함수를 사용하여 코아라우터를 선택한다. 둘째로 코아라우터로 멀티캐스트 트래픽이 집중하는 현상이 발생한다 [12-13]. 백본코아트리(Backbone Core Tree: BCT)는 CBT의 단점을 보완하기 위하여 제안되었다[14-15]. BCT는 여러 개의 코아라우터 후보들을 연결하여 백본코아트리(BCT)을 구성하고 이들 백본 코아 트리내의 코아라우터들이 서로 협동하여 트리를 구성하므로 각 멀티캐스트 그룹마다 특정 코아라우터를 선택하지 않아도 되고, 또한 특정 코아라우터로의 트래픽집중을 막을 수 있다.

본 논문에서는 제안하는 CBCT 생성 알고리즘은 기존의 BCT를 발전시킨 것으로 주어진 네트워크 위상(Topology)로부터 최소신장트리(Minimum spanning tree)와 센트로이드(Centroid)를 이용하여 먼저 코아라우터후보들을 선정하고 이들을 최소비용의 백본코아트리로 연결하게 된다. 제시된 알고리즘의 성능을 평가하기 위하여 CBT와 BCT의 멀티캐스트 트리생성비용을 시뮬레이션을 통하여 비교하였으며, CBT보다 낮은 멀티캐스트트리 생성비용을 가짐을 보였다. 또한 제시된 알고리즘은 정적인 네트워크 위상정보를 이용하므로 동적으로 각 멀티캐스트 그룹마다 코아라우터를 선택하지 않아도 된다.

이후 본 논문에서는 용어의 혼동을 막기 위해 다음과 같이 용어를 정리한다. 즉, CBT는 [7]에서 제안된 코아기반트리, BCT는 [14]에서 제안된 백본코아트리, CBCT는 본 논문에서 제안하는 센트로이드 기반 BCT 생성알고리즘에 의해 자동 생성된 백본코아트리이다.

본 논문의 전체 구성은 다음과 같다. 2장에서는 본 논문에서 사용하는 용어와 기본개념을 설명하고, 3장에서는 CBCT 생성을 위한 요구사항 및 CBCT생성 과정을 소개한다. 4장에서 CBCT 생성 알고리즘을 소개하고, 5장에서는 GT-ITM 모델에 대한 실험 결과를 소개한다. 마지막 6장에서 결론을 내린다.

2. 기본 개념

2장에서는 대표적인 공유 기반 트리기법인 CBT와 이를 확장한 BCT를 소개하고, 본 논문에서 사용하는 센트로이드(Centroid)의 개념, 센트로이드를 찾는 방법, 그리고 최소 신장 트리(Minimum spanning tree)에 대하여 설명한다.

2.1 CBT와 BCT

CBT는 하나의 코아라우터를 선정하여, 이것을 루트(root)로 하는 양방향 멀티캐스트 트리(bidirectional multicast tree)를 생성하여 멀티캐스트 패킷을 그룹 멤버들에게 전달하는 라우팅 프로토콜이다. 그림 1은 라우터 B가 코아라우터인 CBT를 나타낸 것으로 6명의 유

저(user)로 구성된 하나의 멀티캐스트 그룹을 표현한 것이다. 링크(link)위의 숫자는 링크 비용을 의미한다. 그림 1과 같이 멀티캐스트 그룹 멤버를 연결하는 멀티캐스트 트리를 생성할 경우 멀티캐스트 트리 생성 비용은 46이 된다. 여기서, 멀티캐스트 트리 생성 비용이란 주어진 멀티캐스트 그룹 멤버들 사이에 멀티캐스트 패킷을 전달하기 위해 필요한 자원(resource)을 의미한다 [14-15].

또다른 성능지수로서, 멀티캐스트 트래픽 집중 벡터(MTCV: Multicast Traffic Concentration Vector)가 있다. 이는 코아라우터에 연결된 가지(branch) 중에 멀티캐스트 패킷이 전달되는 가지 수로서, 라우터 B의 MTCV는 6으로 라우터 B에 멀티캐스트 트래픽이 집중됨을 알 수 있다[14-15].

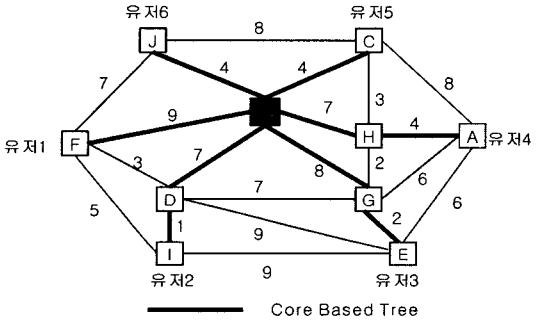

그림 1 CBT(Core Based Tree)

백본코아트리(BCT)는 CBT를 확장한 것으로 코아라우터 후보들을 백본 네트워크로 연결하여 이들 코아라우터 후보들이 서로 협동하므로써 공유트리를 형성하게 된다. 효과적인 코아라우터 후보들을 선택하기 위한 한 방법으로서 전체 네트워크를 몇 개의 서브영역(Sub region)으로 분할하고, 각 서브영역을 관장하는 하나의 코아라우터후보 또는 로컬 코아라우터(Local core router)를 선택하여 서브영역내의 모든 멀티캐스팅은 로컬코아라우터를 중심으로한 트리를 구성하여 처리한다. 또한 서브영역들 사이의 멀티캐스팅은 각 서브영역의 로컬코아라우터를 연결한 백본링크를 이용하여 멀티캐스트 패킷을 전달하는 라우팅 프로토콜이다. 그림 2는 BCT의 구조를 표현한 것으로, 각 서브영역의 로컬 코아라우터는 백본네트워크로 연결되어 있다. 서브영역1에서는 라우터 A가 로컬코아라우터이다. 서브영역4의 유저1, 유저2가 멀티캐스트 그룹 X를 구성하면 라우터 D는 서브영역4의 로컬코아라우터로써 유저1과 유저2에게 멀티캐스트 패킷을 전달하게된다. 만약 유저3이 멀티캐스트 그룹 X에 조인하게 되면 유저3은 서브영역 3에 있으므로 로컬코아라우터 C를 통해 멀티캐스트 트래픽을 전달받는다. 이때 로컬코아라우터 C는 백본 네트워크를 통하여 로컬코아라우터 D와 멀티캐스트 트래픽을 공유하게 된다. 즉, BCT의 로컬 코아라우터들은 자신이 관리하는 멀티캐스트 그룹 정보를 주기적으로 받음으로써 전체 네트워크의 멀티캐스트 그룹정보를 파악한다. 각 서브영역안에 있는 라우터들은 자신에게 가장 인접한 로컬코아라우터에게 조인(join)메시지, 프룬(prune)메시지 등과 같은 제어 메시지를 전달함으로써 멀티캐스트 그룹에 가입 및 탈퇴를 한다.

예를 들어, 그림 3은 네트워크 관리자에 의하여 미리 라우터 B, D, H를 로컬코아라우터로 지정하여 이들을 연결한 백본을 BCT로 지정한 것이다. 이 예제에서 그림 1과 같이 6명의 유저가 같은 멀티캐스트 그룹을 형성한다고 가정할 때 유저1은 세 개의 로컬코아라우터 중에 가장 인접한 D에게 조인(join)을 하고, 유저3은 H

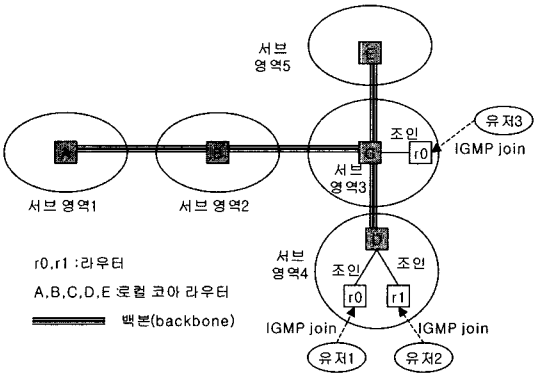

그림 2 BCT 구조(ARchitecture)

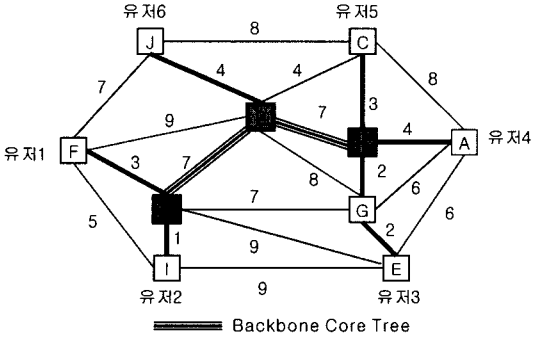

그림 3 BCT(Backbone Core Tree)

에게 조인을 한다. 같은 방법으로 6명의 유저가 멀티캐스트 그룹에 가입을 했을 때 전체 멀티캐스트 트리 생성 비용은 33이 된다. 이 값은 그림 1에서의 멀티캐스트 트리 생성 비용보다 작음을 알 수 있다. 또한 라우터 H의 MTCV는 4이고, 라우터 B의 MTCV는 3, 라우터 D의 MTCV는 3으로 전체적으로 특정 라우터에 멀티캐스트 트래픽이 집중되는 현상 역시 CBT보다 완화되었음을 알 수 있다. 이와 같이, BCT는 CBT보다 멀티캐스트 트리 생성 비용과 트래픽 집중 현상에서 효과적임을 알 수 있다.

2.2 BCT의 재구성

BCT의 멀티캐스트 트리생성 비용은 어떻게 BCT를 구성하는가에 따라 변화한다. 즉, 코아라우터후보 또는 로컬코아라우터를 어떻게 선택하는가에 따라 멀티캐스트 트리 생성비용이 변화하게 된다. 그림 3의 BCT를 그림 4처럼 재구성하면 멀티캐스트 트리 생성 비용은 44로 CBT와 별다른 차이가 없다. 하지만 그림 5처럼 다시 BCT를 재구성하면 멀티캐스트 트리 생성 비용이 29가 된다.

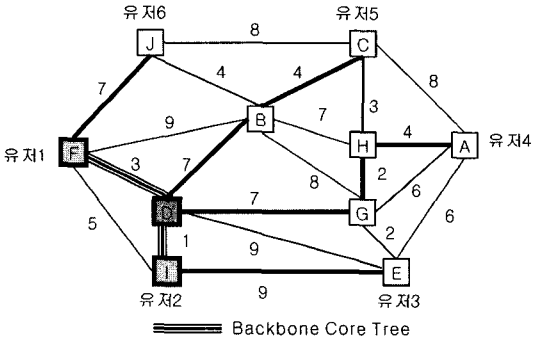

그림 4 BCT의 재구성 I

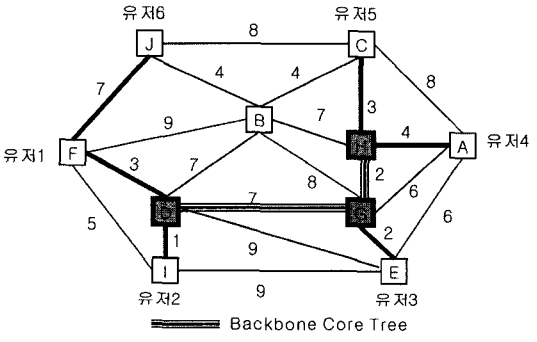

그림 5 BCT의 재구성 II

즉, 주어진 네트워크에서 어떻게 BCT를 구성하느냐에 따라서 멀티캐스트 트리 생성 비용이 차이가 남을 알 수 있다. 그런데 기존의 연구에서는 BCT의 개념과 BCT가 CBT보다 효과적임을 소개할 뿐 어떻게 BCT를 구성할 것인지에 대하여 언급을 하지 않고 있다. 이것이 다음 장에서 설명할 센트로이드와 최소 신장 트리를 이용하여 효율적인 BCT생성 알고리즘을 제시하고자 하는 동기이다.

2.3 센트로이드와 최소 신장 트리

센트로이드는 다음 장에서 BCT를 생성하는데 사용되는데, 센트로이드의 정의는 다음과 같다. n개의 노드를 가진 트리 T(n>1)에서 임의의 노드 v를 선택하여, 노드 v와 연결되어 있는 링크(또는 에지)들을 T로부터 제거하면 T는 k개의 서브트리(Subtree) T_1, T_2, ..., T_k로 나누어진다. 이때 노드 v의 값은 value(v)= MAX{$|T_1|$, $|T_2|$, ..., $|T_k|$} (단, $|T_i|$는 T_i에 속하는 노드의 수)라 정의할 때, 노드 v 중에서 value(v)가 가장 작은 노드 c를 트리 T의 센트로이드(Centroid)라 한다[16-17]. 그림 6에서는 노드 d의 value(d)가 3으로 다른 노드들의 value(v)보다 작기 때문에 노드 d가 트리 T의 센트로이드가 된다.

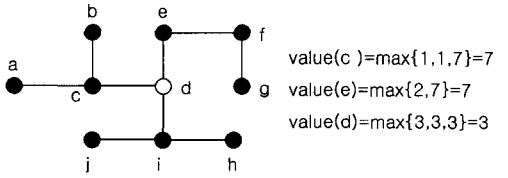

그림 6 센트로이드의 개념

임의의 트리 T의 센트로이드를 구하기 위해서, 먼저 T를 임의의 노드가 루트가 되는 루트 트리(rooted tree)로 변환하다. 또한 루트 트리에서 새로운 함수 size(v)를 다음과 같이 정의한다. 임의의 노드 v에 대한 size(v)는 노드 v자신을 포함하고 노드 v를 루트로 하는 서브 트리의 노드 개수를 의미한다.

다음은 트리에서 센트로이드를 쉽게 찾기 위해 필요한 정리이다.

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

증명 : 본 논문 부록 A에 증명을 하였다. □

Lemma 1.

(1) 트리 T가 두 개의 센트로이드 c와 d를 가질 필요

충분조건은 n이 짝수이고, size(c)=n/2 이다.

(2) T가 하나의 센트로이드 c만을(c의 자식노드를 d_1, d_2, ..., d_k)가질 필요충분조건은

$2 \cdot size(di) \leq n-1$,

$2 \cdot size(c) \geq n+1$ ($1 \leq i \leq k$, n은 트리 T의 노드 수).

증명 : 본 논문 부록 B에 증명을 하였다. □

트리 T에서 각 노드 v의 size(v)는 트리 T를 포스트오더(postorder)순서로 방문하면 O(n)시간에 계산할 수 있다. 그림 7은 그림 6을 루트 트리(rooted tree)로 변환한 것으로 (a)는 노드 b를 루트 노드로 변환했을 때 각 노드의 size(v)값을 나타낸 것이고, 그림 7의 (b)는 노드 e를 루트 노드로 했을 때 각 노드의 size(v)를 나타낸 것이다.

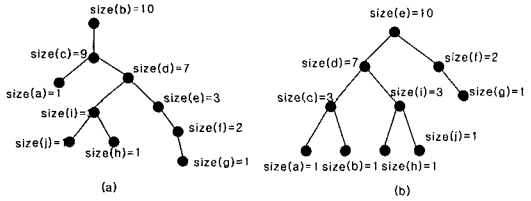

그림 7 임의의 노드를 루트로 변환했을 때 각 노드의 size(v)

정리 2. n개의 노드를 가진 임의의 트리 T에서 센트로이드는 O(n)시간에 구할 수 있다.

증명 : 임의의 트리 T를 O(n)시간 안에 루트 트리로 만들 수 있고 또한 각 노드에 대하여 O(n)시간에 lemma1을 적용하여 센트로이드를 구할 수 있다. □

예를 들어, lemma1을 그림 7의 (a)에 적용하면 노드 d만 lemma1의 (2)를 만족한다. 그림 7의 트리 노드 수가 짝수이지만(n=10) lemma1의 (1)을 만족하는 노드가 없기 때문에 하나의 센트로이드만 존재한다. 노드 d에 대하여 lemma1의 (2)를 적용하면 아래의 수식 ①, ②, ③ 모두가 참임으로 노드 d가 그림 7의 (a)트리의 센트로이드가 된다.

$$2 \cdot size(d) = 2 \cdot 7 \geq 10+1 \quad ①$$

$$2 \cdot size(i) = 3 \leq 10-1 \quad ②$$

$$2 \cdot size(e) = 3 \leq 10-1 \quad ③$$

같은 방법으로 그림 7의 (b)에 lemma1의 (1)과 (2)를 적용하면 역시 노드 d가 센트로이드가 됨을 알 수 있다.

다음은 본 논문에서 사용하는 두 번째 개념인 최소 신장 트리(Minimum Spanning Tree: MST)를 소개한다. MST란 각 에지의 가중치가 부여된 그래프 G(가중치 그래프: weighted graph)에서 모든 노드를 연결하는 에지들의 가중치 합이 최소인 트리를 말한다[18]. MST는 기존의 Prim 알고리즘 또는 Kruskal 알고리즘을 이용하면 쉽게 구할 수 있다[18]. Kruskal 알고리즘은 에지들의 값을 정렬하여 값이 작은 에지를 선택하여 최소 신장 트리를 생성하는 것으로 O(E logV)에 계산할 수 있다(단, E는 에지의 수이고 V는 노드의 수이다). Prim 알고리즘은 임의의 노드 n을 선택하여 n에 연결된 최소 비용 에지 e을 선택하여 이 에지를 MST에 추가하고, 에지 e에 연결된 다른 노드가 m일 경우, 노드 m에서 다시 최소 비용 에지를 찾아 MST에 추가하는 방법으로 O((E+V)logV)시간에 최소 신장 트리를 생성한다.

3. CBCT 생성

본 장에서는 CBCT를 생성하기 위한 요구사항을 분석하고 MST와 센트로이드를 이용한 효율적인 CBCT 생성 알고리즘을 소개한다.

3.1 BCT 생성을 위한 요구사항

BCT를 이용한 IP 멀티캐스팅에서는 어떻게 BCT를 구성할 것인가가 중요한 문제가 된다. 따라서 BCT를 결정하기 위해서 다음과 같은 문제를 고려해야 한다.

1) BCT를 구성하는 로컬코아라우터들 사이의 링크 비용은 최소가 되어야 한다. BCT를 구성하는 로컬코아라우터들 사이의 링크 비용이 크게 되면 전체적으로 멀티캐스트 트리를 생성하는 비용이 증가하고 BCT 유지비용이 증가하게 된다.

2) 전체 네트워크를 몇 개의 서브영역으로 나눌 때 각 서브영역에 속한 라우터들의 수는 균형 있게 유지해야 한다. 서브영역 A의 라우터 수는 100개이고, 서브영역 B의 라우터 수가 6개라면, 결국 서브영역 A의 로컬 코아라우터에 멀티캐스트 트래픽이 집중되어 CBT가 가지고 있던 문제점을 그대로 가지기 때문이다.

3) BCT는 쉽게 확장 할 수 있어야 한다. 한번 백본 네트워크를 구성하게 되면 쉽게 변경하는 경우는 드물기 때문에 BCT를 확장해야 하는 경우 기존의 BCT를 포함하면서 새로운 라우터를 쉽게 추가할 수 있어야 한다.

효과적인 BCT를 생성하기 위해서는 위의 3 가지 요구사항을 만족해야 한다. 다음에 제시할 CBCT 생성 알고리즘은 이 요구사항들을 모두 만족한다.

3.2 최소 신장 트리와 센트로이드를 이용한 CBCT 생성

BCT를 구성하는 로컬코아라우터는 주기적으로 멀티캐스트 그룹 정보를 서로 교환해야 한다. 따라서 BCT를 구성하는 로컬코아라우터들 사이의 링크 비용이 증가하게 되면 멀티캐스트 그룹 정보를 교환하기 위한 전송비용이 증가하게 됨으로 BCT를 구성하는 로컬코아라우터들 사이의 링크비용은 최소가 되어야 한다. 본 논문에서는 이러한 제약 조건을 만족시키기 위해 주어진 네트워크 그래프 G에서 모든 라우터를 연결하는 최소 신장 트리(MST)를 생성하여 이들 중에서 로컬코아라우터를 선택한다. 이렇게 할 경우 로컬 코아라우터들을 연결하는 링크비용은 최소가 된다.

또한 본 논문에서는 각 서브영역의 라우터 수를 균등하게 분배하고, 종단 라우터(Terminal router)에서 가장 인접한 BCT의 로컬코아라우터까지 홉 수(Hop count)를 일정하게 유지하기 위하여 센트로이드(Centroid)를 이용하여 CBCT를 생성한다. 먼저 네트워크 그래프 G에서 MST를 생성하고, MST에서 센트로이드를 찾아 센트로이드와 연결된 링크를 제거하면 k개의 서브트리가 생성되는데 이때 각 서브트리를 하나의 서브영역으로 간주한다. 여기서 각 서브트리의 노드 수는 정리 2에 의하여 모두 n/2이하가 됨으로 특정한 서브영역에 많은 라우터가 할당되는 것을 예방할 수 있다.

3.3 CBCT 생성 과정

본 논문에서 CBCT를 생성하는 과정은 다음과 같다. 먼저 주어진 네트워크 그래프 G에서 최소 신장 트리를 구한다(단, 최소 신장 트리를 T_{sp}라 표현한다). 다음으로 T_{sp}에서 센트로이드를 구한다. 그림 8에서 노드 c가 T_{sp}의 센트로이드라 가정할 때 센트로이드 c는 k개의 서브트리를 가진다. 노드 d_1, d_2, ... , d_k를 각 서브트리의 루트 노드라 가정할 때 각 서브트리에 속하는 노드 수가 m보다 클 경우 서브트리의 루트 노드를 CBCT에 추가한다. 예를 들어, 서브트리 T_2가 $|T_2|\geq m$ 조건을 만족하면 노드 d_2를 CBCT에 추가한다. 여기서 m은 종단라우터에서 가장 인접한 CBCT의 로컬코아라우터까지의 홉 수(Hop count)인데, 이 m값에 따라 CBCT에 속하는 로컬코아라우터 수가 결정된다. m값에 대한 자세한 것은 5장에서 설명하겠다.

그 다음 단계로 서브트리의 노드 수가 m보다 큰 모든 서브트리에 대하여, 해당 서브트리의 센트로이드 c′를 찾아 CBCT에 속하는 로컬코아라우터를 계산한다. 그림 8에서 서브트리 T_i에서 $|T_i| \geq m$라 할 때 서브트리 T_i의 센트로이드 c′를 구하면 센트로이드 c′와 연결된 서브트리 $T_{i1}, T_{i2}, ..., T_{ii}, ..., T_{ik}$가 생성된다(단, $1 \leq i \leq k$). 서브트리 T_i의 센트로이드 c′와 연결된 노드 $d_{i1}, d_{i2}, ..., d_{ii}, ..., d_{ik}$에 대하여 CBCT에 추가여부를 결정한다. 예를 들어, 서브트리 $|T_{ii}|\geq m$ 라면 노드 d_{ii}를 CBCT에 추가한다. 만약 서브트리 $|T_{ii}|<m$라면 노드 d_{ii}는 CBCT에 추가하지 않는다. 같은 방법으로 모든 서브트리에 대하여, 서브 트리의 노드 수와 m을 비교하여 CBCT를 생성한다.

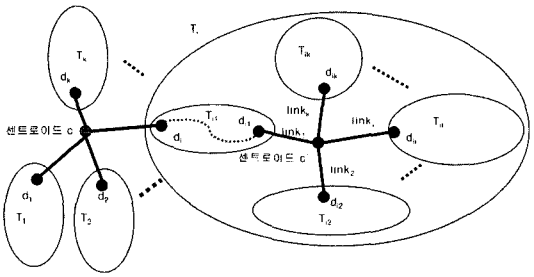

그림 8 CBCT 생성 과정

4. CBCT 생성 알고리즘

본 논문에서 제안하는 CBCT생성 알고리즘은 그림 9와 같다.

```
Procedure CBCT_Build(G, m)
//G : Network graph
//m : Number of max hop counts from terminal node to
      BCT
begin
  Node first_node= take any node from Network Graph G;
  Tsp=Minimum_Spanning_Tree(first_node);
  numNodes=FindCentroid(Tsp. root, m); // root 는 Tsp의
           루트노드
  CBCTGeneration(); // CBCT의 모든 코아라우터후보들을
           연결하여 CBCT를 생성
end

Procedure Int FindCentroid(root, m)
//리턴값: 서브트리의 노드수 (=size(root))
begin
  numNodes=size(root);//postorder순으로 모든 노드를 방문
           하여 size(root)를계산
  if(numNodes>= m)
  {
     Centroid = CalCentroid(root, numNodes); //트리의 센
           트로이드를 계산
     MarkCBCT(Centroid); //Centroid는 CBCT의 코아라
           우터후보중 하나이다.
      For each adjacentNode k of Centroid:
          numNodes=FindCentroid(k, m);
          if(numNodes >= m) MarkCBCT(k);
     }
     return numNodes;
  }
  return numNodes;
end
```

그림 9 CBCT 알고리즘

CBCT_Build 프로시저는 네트워크 그래프 G와 m을 입력으로 받아, CBCT를 생성하는 프로시저이다. 먼저, 주어진 네트워크 그래프 G에서 임의의 노드를 루트로 지정하여, 최소 신장 트리를 구한다. 최소 신장 트리 T_{sp}가 만들어 지면, *FindCentroid*(T_{sp},*root, m*)함수를 호출한다. *FindCentroid*(*root, m*)함수의 매개변수 root는 트리의 루트 노드를 가리키며, m은 종단 노드(또는 종단 라우터)에서 CBCT의 코컬 코아라우터까지 가는 경로상의 최대 홉수이다. *FindCentroid*(*root, m*)함수의 동작은 다음과 같다. 먼저 각 노드를 포스트오더(postorder) 순으로 방문하여 각 노드의 size(v)를 계산한 후 최종적으로 루트노드의 size(root)를 (즉, size(root)는 트리에 속하는 노드의 수가 된다.) 변수 numNodes에 저장한다. 이 단계는 O(n)시간에 계산된다.

If문에서 numNodes가 m보다 작은 경우에는 numNodes를 리턴하고 *FindCentroid*(*root, m*)함수를 종료한다. 하지만 If문에서 numNodes가 m보다 크거나 같은 경우에 트리의 센트로이드를 먼저 계산한다. *CalCentroid (root, numNodes)*함수는 Lemma1을 이용하여 트리의 센트로이드를 계산하는 함수로 정리 2에서와 같이 O(n) 시간에 할 수 있다. *markCBCT(Centroid)*함수는 변수 *Centroid*를 나중에 CBCT에 추가 할 수 있도록 표시(marking)하는 함수이다.

다음으로 센트로이드와 연결된 서브트리에 대하여 *FindCentroid(k, m)* 함수를 수행한다. 여기서, k는 센트로이드와 연결된 모든 노드를 의미한다. *FindCentroid (k, m)*는 되부름(recursion)방법으로 반복수행하면서 센트로이드를 찾는다. 이때 되부름은 O(log*n*)번 된다. 즉, 센트로이드와 연결된 서브트리의 노드 수가 m이하이면 더 이상 서브트리를 찾지 않지만, 서브트리의 노드 수가 m보다 크거나 같은 경우 서브트리에 속하는 센트로이드를 계산한다. 변수는 numNodes는 센트로이드와 연결된 각 서브트리의 노드 수를 저장하는 변수인데, numNodes의 값이 m보다 크거나 같으면 서브트리의 노드 k를 CBCT에 추가할 수 있도록 표시(marking)을 한다. *CBCT-Generation()*함수는 *markCBCT()*함수에 의하여 표시된 노드를 연결하여 CBCT를 생성하는 함수이다.

정리 3. 임의의 CBCT가 n개의 로컬 코아 라우터로 구성되었을 때 이들을 연결하는 링크들의 가중치 합은 네트워크 그래프 G에서 n개의 로컬 코아 라우터를 포함하는 다른 어떤 경로의 가중치 합보다 항상 작거나 같다.

증명 : MST는 네트워크 그래프 G의 모든 노드를 연결하는 에지들의 가중치 합이 최소인 트리를 말한다. MST에 포함된 임의의 노드 x, y, z가 있을 때 이들은 연결하는 경로는 이들을 포함하는 다른 어떤 경로들의 가중치 합보다 항상 작거나 같다. CBCT는 MST에 의하여 생성되는 트리로 MST의 부분집합이다. 따라서, CBCT의 모든 노드는 MST에 포함되어 있으면, 이들을 연결하는 링크들의 가중치 합은 이들을 포함하는 다른 경로들의 가중치 합보다 항상 작거나 같다. □

5. 실 험

본 논문에서 제안하는 CBCT생성 알고리즘의 성능을 비교하기 위해서 네트워크 위상(Topology) 생성 패키지(*Network Topology Generation Package*)인 GT-ITM (Georgia Tech Internetwork Topology Models)을 이용하여 실험 모델을 생성하였다[19]. 실험 모델은 Pure Random 네트워크모델, Waxman I 네트워크모델, Transit-Stub 네트워크모델을 사용하였다. 또한 각 모델에 대하여 각각 30개, 50개, 100개의 노드들을 생성하여, 1-20사이의 링크 비용을 랜덤하게 할당하고 역시 랜덤하게 생성된 멀티캐스트 그룹 멤버들에 대한 멀티캐스트 트리 생성 비용을 비교하였다. 코아라우터를 선택하는 방법은 첫째, CBT에 의한 방법과 둘째 네트워크 관리자가 직관적으로 라우터를 로컬코아라우터로 선택하여 BCT를 구성하는 방법과 마지막으로 본 논문에서 제안하는 CBCT 생성 알고리즘에 의한 로컬코아라우터 생성 방법으로 구분하여 똑같은 실험 상황에 대하여 멀티캐스트 트리 생성 비용을 비교하였다.

일반적으로 공유 트리를 사용하는 멀티캐스트 라우팅 프로토콜은 멀티캐스트 그룹이 Dense 모드보다 Sparse 모드에서 보다 좋은 성능을 가지기 때문에 본 논문에서는 Spares 모드에 대하여 실험하였다[5]. 표 1은 각 실험 모델에서 CBT, BCT, 그리고 제안된 알고리즘인 CBCT의 성능비교를 나타낸 것으로 성능지수로는 각

표 1 CBT, BCT, 그리고 제안된 CBCT의 성능 비교

코아 선택 방법	GT-ITM 모델	멀티캐스트그룹내의 노드의 수		
		30	50	100
CBT	Pure random	36.95	58.2	106
	Waxman I	38.77	45.1	61.6
	Transit-stub	56.5	99.5	89.38
BCT	Pure random	36.54	66.6	107.03
	Waxman I	37.9	48	80.7
	Transit-stub	35.4	75.45	65.39
CBCT	Pure random	27	50.29	86
	Waxman I	28.9	35.6	56.64
	Transit-stub	35	72.2	61.1

알고리즘에서의 멀티캐스트 트리 생성 비용의 평균치를 사용하였다.

Pure Random 네트워크모델에서 CBT와 BCT의 결과를 비교할 때 거의 비슷한 결과를 보임을 알 수 있다. 하지만 본 논문에서 제안하는 CBCT방법과 기존의 방법과 비교할 경우 본 논문에서 제안하는 CBCT 생성 방법이 평균 24%의 멀티캐스트 트리 생성 비용을 절감한다. 그림 10은 Pure Random 네트워크모델에서 CBT와 BCT 및 본 논문에서 제안하는 CBCT와의 성능을 비교한 그래프이다.

그림 11은 Waxman I 네트워크모델에서 CBT와 BCT 및 본 논문에서 제안하는 CBCT와의 성능을 나타낸 것이다. Waxman I 네트워크모델에서도 본 논문에서 제안하는 CBCT 방법으로 할 때 평균 29%의 멀티캐스트 트리 생성 비용을 절감한다.

Transit-Stub 네트워크모델의 경우 BCT를 형성하는 방법과는 관계없이 모든 경우 CBT에 비해 항상 좋은 성능을 보임을 알 수 있다. 본 실험에서는 CBT에 비해 본 논문에서 제안하는 CBCT가 48%이상 멀티캐스트 트리 생성비용을 절약함을 알 수 있다. 그림 12는 Transit-Stub 네트워크모델에서 성능 관계를 비교한 것이다.

이와 같이 본 논문에서 제안하는 CBCT 생성 방법이 네트워크 관리자에 의하여 BCT를 선정하는 방법이나 하나의 코아를 선정하는 CBT보다 효과적으로 멀티캐스트 트리를 생성함을 알 수 있다.

본 논문에서 제안하는 CBCT 생성 알고리즘은 m값에 따라 CBCT에 포함되는 로컬코아라우터 수가 결정된다. 즉, m값이 증가하면 종단 라우터에서 CBCT의 가장 인접한 로컬코아라우터까지의 홉 수가 커지기 때문에 CBCT에 포함된 로컬코아라우터 수가 감소하게

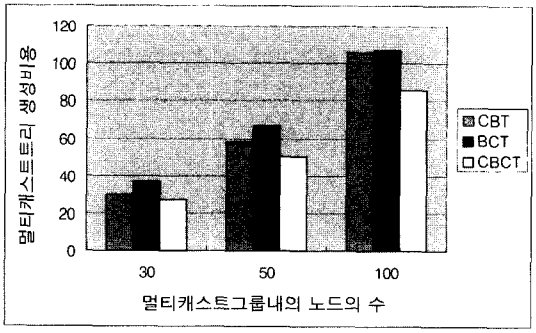

그림 10 Pure Random 네트워크모델에서 제안된 CBCT의 성능

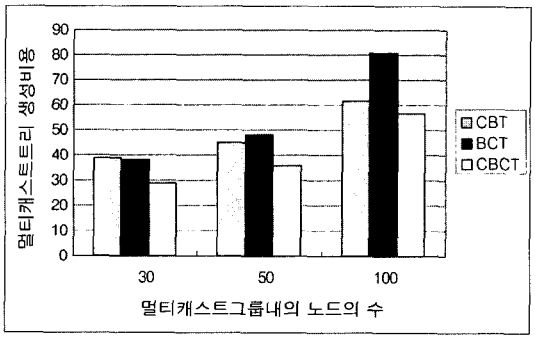

그림 11 Waxman I 네트워크모델에서 제안된 CBCT의 성능

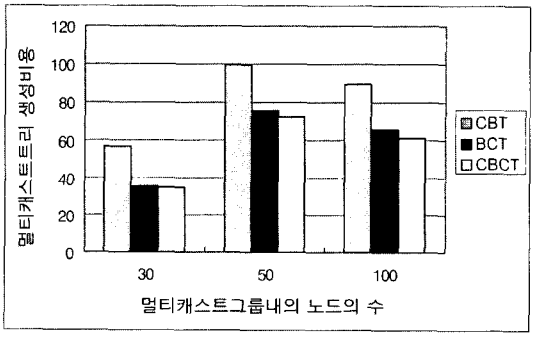

그림 12 Transit-Stub 네트워크모델에서 제안된 CBCT의 성능

되고, m값이 작아지면 홉 수가 작아지기 때문에 CBCT에 포함된 로컬코아라우터 수가 증가하게 된다. CBCT의 로컬코아라우터 수가 증가하면 로컬코아라우터끼리 주고받는 멀티캐스트 그룹 정보의 양이 증가함으로 CBCT 유지비용이 증가하게 된다. 따라서, CBCT의 장점을 최대로 하고, CBCT 유지비용을 최소로 하는 최적의 m값 결정이 중요하게 된다.

본 논문에서는 최적의 m값을 선택하기 위하여 역시 Pure Random 네트워크모델, Waxman I 네트워크모델, Transit-Stub 네트워크모델에 대하여 30, 50, 100개의 노드를 가지는 네트워크 모델을 생성하여 m값을 변경하면서 멀티캐스트 트리 생성 비용과 로컬 코아라우터 수의 관계 변화를 비교해 보았다.

m값을 $\log n$, $\log \frac{n}{2}$, $\log \frac{n}{2^2}$, $\log \frac{n}{2^3}$, $\log \frac{n}{2^4}$으로 바꾸면서 실험하였다. 예를 들어 n=30일 때 m값은 다음과 같다.

$m = \log n = \log 30 < \log 32 = \log 2^5 = 5$,

$$m = \log \frac{30}{2^2} = \log \frac{30}{4} < \log 8 = \log 2^3 = 3,$$

$$m = \log \frac{30}{2^3} = \log \frac{30}{8} < \log 4 = \log 2^2 = 2,$$

$$m = \log \frac{30}{2^4} = \log \frac{30}{16} < \log 2 = \log 2^1 = 1$$

여기서 $m = \log \frac{30}{2^4}$ 일 경우 m값이 1보다 작기 때문에 이 경우는 실험에서 제외하였다. 즉, m=1이라면 네트워크 그래프 G에서 종단 라우터를 제외한 모든 라우터가 CBCT에 포함되기 때문에 이와 같은 경우는 현실에서 적용하는 경우가 없기 때문에 실험에서 제외하였다.

그림 13은 Pure Random 네트워크모델에서 m값의 변화에 따른 멀티캐스트 성능을 시뮬레이션 한 결과이다. 즉, 그림 13의 (a)는 m값에 따른 로컬코아라우터 수를 표현한 것이고, 그림 13의 (b)는 m값에 따른 멀티캐스트 트리 생성비용을 나타낸 것이다. 그림 13의 (a)에서 m값이 작아짐에 따라 CBCT에 속하는 로컬코아라우터 수는 증가함을 알 수 있다. 하지만 그림 13의 (b)에서 멀티캐스트 트리 생성 비용은 m값의 변화에 큰 영향 없이 거의 일정함을 알 수 있다. 그런데 CBCT에 속하는 로컬코아라우터 수가 증가하게 되면 CBCT의 유지비용이 증가되기 때문에 가능한 m값을 크게 하는 것이 바람직하다. 즉, m값은 로컬코아라우터 수에만 영향을 주고 멀티캐스트 트리 생성 비용에는 거의 영향이 없음으로 m값을 최대한 크게 하는 것이 바람직하다. 결국 m값이 $\log n$일 때 최소의 로컬 코아 라우터로 CBCT를 구성하게 됨을 알 수 있다.

그림 14와 그림 15는 각각 Waxman I 네트워크모델과 Transit-stub 네트워크모델에 대하여 m값의 변화에 따른 로컬 코아라우터 수와 멀티캐스트 트리 생성 비용의 변화를 표현한 것이다. 두 모델 역시 Pure Random 네트워크모델과 같이 m값이 멀티캐스트 트리 생성 비용에는 거의 영향이 없으며, CBCT에 속하는 로컬 코아 수에만 영향을 주기 때문에 m값이 $\log n$일 때 가장 좋은 CBCT를 생성함을 알 수 있다.

위의 실험을 통해 멀티캐스트 생성 트리 비용은 m값에 영향을 거의 받지 않으며 CBCT에 속하는 로컬코아라우터 수만 m값에 영향을 받음을 알 수 있었다. 따라

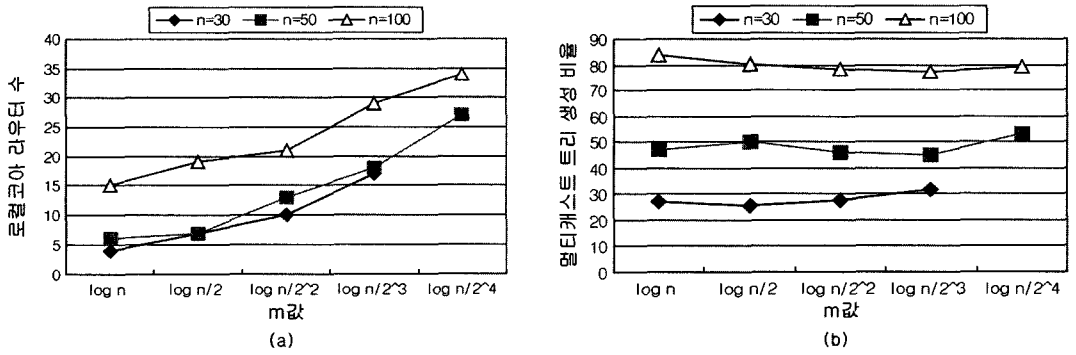

그림 13 Pure Random 네트워크모델

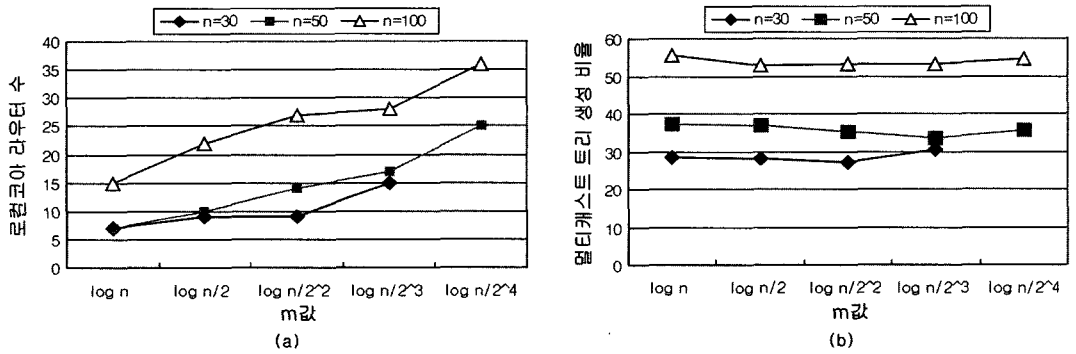

그림 14 Waxman I 네트워크모델

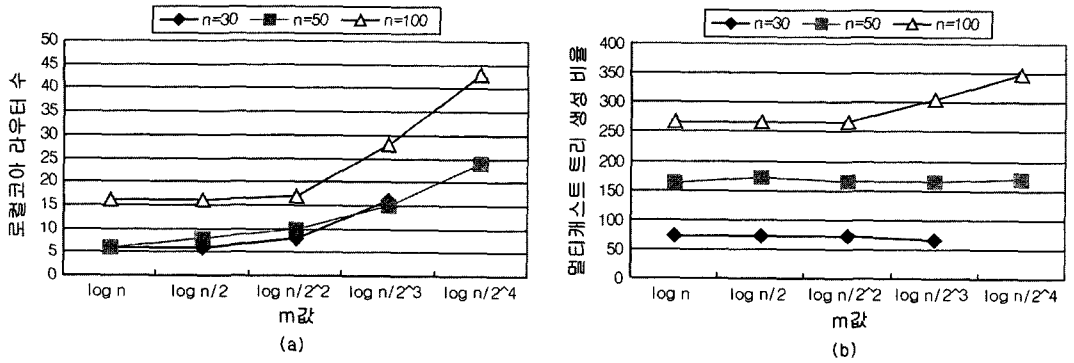

그림 15 Transit-Stub 네트워크모델

서, m값이 클수록 CBCT에 속하는 로컬코아라우터 수는 감소하고, CBCT 유지비용은 작게 됨으로 m값이 $\log n$일 때 CBCT의 유지비용을 최소로 하는 CBCT를 생성하게 된다.

6. 결 론

본 논문은 IP 멀티캐스팅에서 CBT의 단점을 보완하기 위해 제안된 BCT에서 멀티캐스트 트리 생성 비용을 최소로 하는 효율적인 CBCT 생성 알고리즘을 제안하였다. 제안된 CBCT 생성 알고리즘은 네트워크 그래프 G에서 최소 신장 트리를 생성하고, 최소 신장 트리에서 센트로이드를 찾아서 CBCT를 생성한다. 본 논문에서 제안하는 CBCT생성 알고리즘은 CBT나 네트워크 관리자에 의하여 임의로 설정된 BCT보다 멀티캐스트 트리 생성 비용이 적고, BCT의 확장이 용이하다. CBCT에 속하는 로컬 코아 라우터의 수는 m값, 즉 종단 라우터에서 가장 인접한 CBCT의 로컬 코아 라우터까지의 홉수에 의하여 결정되며, m값이 $\log n$일 때 CBCT의 유지비용을 최소로 하는 CBCT를 생성함을 실험을 통해 알 수 있었다.

앞으로의 과제는 보다 다양한 실험 모델에서 CBCT의 성능을 평가하여 Qos를 보장하는 CBCT 생성 알고리즘을 제안하는 것이다. 또한 end-to-end 지연을 최소로 하는 CBCT 생성 방안 제안과 CBCT에 결함이 발견되었을 때 결함을 해결할 수 있는 기법을 연구하는 것이 앞으로의 연구과제이다.

부록 A.

정리 1. 의 증명

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

증명 :

증명을 두부분으로 나누어 (1)에서는 트리 T가 3개 이상의 센트로이드를 가질 수 없음을 증명하고 (2)에서는 두개의 센트로이드는 이웃하게 됨을 증명한다.

(1) 트리 T에는 세 개의 센트로이드 x, y, z가 존재할 수 없음을 증명해보자. 트리 T를 일반성을 잃지 않기 위해 그림 16과 17의 두 가지 구조로 나누어서 생각해 본다.

먼저 트리 T가 그림 16의 경우라고 할때, 노드 x, y, z가 T의 센트로이드들이라 가정해보자. 단, 센트로이드 x가 포함된 서브트리 T_x의 크기(즉, T_x의 노드의 수)를 $|T_x|$라 하고, T_y의 크기를 $|T_y|$, T_z의 크기를 $|T_z|$라 한다.

노드 x에 대하여 $value(x) \geq 1+|T_y|+|T_z|$ 된다 (단, $value(v)=MAX\{|T_1|, |T_2|, ..., |T_k|\}$). 이것은 $|T_x|>|T_y|$, $|T_x|>|T_z|$을 의미한다.

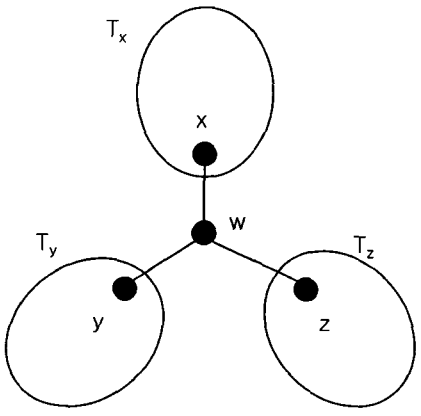

그림 16 센트로이드가 3개인 트리 I

노드 y에 대하여 value(y)≥1+|T_x|+|T_z|되고 |T_y|>|T_x|, |T_y|>|T_z|된다.

역시 노드 z에 대하여 value(z)≥1+|T_x|+|T_y|을 만족하고, |T_z|>|T_x|, |T_z|>|T_y|가 성립 된다.

여기서, 노드 x의 관점에서 노드 w의 value(w)는

value(w)=MAX{|T_x|, |T_y|, |T_z|} ≤ value(x). ④

노드 y의 관점에서 노드 w의 value(w)는

value(w)=MAX{|T_x|, |T_y|, |T_z|} ≤ value(y). ⑤

노드 z의 관점에서 노드 w의 value(w)는

value(w)=MAX{|T_x|, |T_y|, |T_z|} ≤ value(z). ⑥

그런데 x, y, z는 T의 센트로이드임으로

value(w) > value(x),

value(w) > value(y),

value(w) > value(z). ⑦

수식 ④, ⑤, ⑥에 의해서 노드 w가 T의 센트로이드가 된다. 이것은 트리 T가 세 개의 센트로이드(x, y, z)를 가진다는 가정에 위배 되고, 수식 ④, ⑤, ⑥와 ⑦은 서로 모순임을 알 수 있다. 즉 트리 T에는 세 개의 센트로이드가 존재할 수 없다.

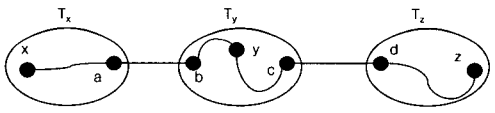

그림 17 센트로이드가 3개인 트리 II

이번에는 두 번째 경우로서 트리 T가 그림 17의 구조를 가진다고 생각해보자. 이때 트리 T는 세 개의 센트로이드(노드 x, y, z)를 가진다고 가정하자. 즉, 노드 y가 노드 x와 노드 z사이의 경로중에 존재함을 알 수 있는데, 다음과 같은 수식이 성립하게 된다.

value(x) ≥ value(a) ≥ |T_y|+|T_z| ⑧

value(y) ≥ value(b) ≥ |T_x| ⑨

value(y) ≥ value(c) ≥ |T_z| ⑩

value(z) ≥ value(d) ≥ |T_y|+|T_x| ⑪

그런데, 노드 x, y, z가 모두 트리 T의 센트로이드들이므로 value(x), value(y), value(z) 각각의 값은 다른 노드들의 value 값보다는 항상 작아야 한다. 이는 위의 수식 ⑧, ⑨, ⑩, ⑪과 위배됨을 알 수 있다. 따라서 위의 트리 T에는 세 개의 센트로이드가 존재한다는 가정이 모순이다.

(2) 다음으로 트리 T에 두개의 센트로이드가 존재할 경우 이들은 이웃함을 증명을 하기로 하자. 노드 v와 u를 트리 T의 센트로이드라 하고, 서로 이웃하지 않는다고 가정해보자.

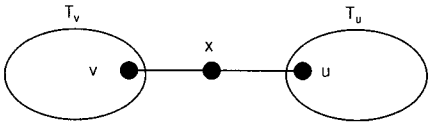

그림 18 v, u가 센트로이드인 트리

그림 18처럼 노드 v와 노드 u사이의 경로에 노드 x가 존재한다고 할 때, 그 경로사이에 어떤 노드 x도 존재 할 수 없음을 보이면 노드 v, u가 이웃함을 증명하게 된다.

노드 v, u에 대하여,

value(v) ≥ 1+|T_u|, ⑫

value(u) ≥ 1+|T_v|. ⑬

노드 x에 대하여,

value(x) = MAX{|T_v|,|T_u|} ⑭

즉, |T_v|, |T_u| 중 큰 값이 value(x)가 된다. 그런데, 노드 v, u가 가정에 의해 센트로이드임으로 value(x)는 value(v)와 value(u) 보다 커야한다. 여기서, 수식 ⑫, ⑬과 수식 ⑭를 동시에 만족하는 노드 x는 존재하지 않는다. 따라서 센트로이드 v와 u는 서로 이웃한다. □

부록 B.

Lemma 1. 의 증명

(1) 트리 T가 두 개의 센트로이드 c와 d를 가질 필요충분조건은 n이 짝수이고, size(c)=n/2 이다.

(2) T가 하나의 센트로이드 c만을(c의 자식노드를 d_1, d_2, ..., d_k)가질 필요충분조건은

2 · size(di) ≤ n-1,

2 · size(c) ≥ n+1 (1≤i≤k, n은 트리 T의 노드 수).

증명 :

(1) n개의 노드를 가지는 트리 T에 대하여, 노드 c와 d를 연결하는 에지(c, d)에 대하여, s(d, c)는 노드 d의 관점에서 볼때 노드 c로부터 시작되는 서브트리의 크기 size(c)라고 가정하자.

그러면,

s(c,d) + s(d,c) = n. ⑮

만약, c와 d가 센트로이드라면 [16]으로부터

s(c,d) = s(d,c). ⑯

수식 ⑮에 수식 ⑯을 대입하면

s(c, d)+s(c, d) =n,

2·s(c, d) = n,

s(c, d) = size(c) = n/2. ⑰

수식 ⑰이 성립하기 위해서는 n이 짝수일 때만 가능하다.

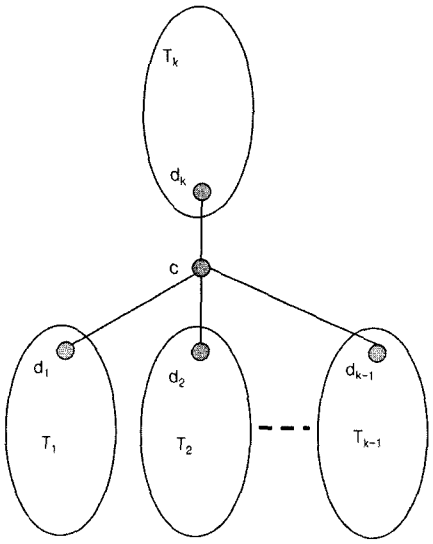

그림 19 노드 c가 센트로이드인 트리 T

(2) 먼저 트리 T가 한개의 센트로이드 c만을 가질 첫 번째 필요충분조건에 대한 증명을 해보기로 하자. 그림 19에서 트리 T의 센트로이드 c는 k개의 자식을 가지는 노드라 할 때 노드 c의 서브트리간에는 다음과 같은 식이 성립한다.

$$|T_1| \le |T_1| + |T_2| + \ldots + |T_k| - |T_1|,$$
$$|T_2| \le |T_1| + |T_2| + \ldots + |T_k| - |T_2|,$$
$$\cdots\cdot$$
$$|T_k| \le |T_1| + |T_2| + \ldots + |T_k| - |T_k|. \quad ⑱$$

수식 ⑱을 정리하면

$$2\cdot|T_1| \le |T_1| + |T_2| + \ldots + |T_k| = n-1,$$
$$2\cdot|T_2| \le |T_1| + |T_2| + \ldots + |T_k| = n-1,$$
$$\cdots\cdot$$
$$2\cdot|T_k| \le |T_1| + |T_2| + \ldots + |T_k| = n-1. \quad ⑲$$

즉, 센트로이드 c와 이웃하는 모든 노드 di 에 대해 위의 첫 번째 필요충분조건인 $2\cdot size(di) \le n-1$가 성립함을 알 수 있다.

이번에는 두 번째 필요충분조건을 증명해본다. 그림 19에서

$$|T_k| = n - (|T_1| + |T_2| + \ldots + |T_{k-1}| + 1),$$
$$= n - size(c). \quad ⑳$$

수식 ⑲와 ⑳으로부터

$$2\cdot|T_k| \le n-1,$$
$$2\cdot|T_k| = 2n-2size(c) \le n-1,$$
$$2\cdot size(c) \ge n+1. \ \square$$

참 고 문 헌

[1] D. Waitzman, S. Deering and C. Partridge, "Distance Vector Multicast Routing Protocol," RFC 1075, Nov. 1988.

[2] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A Helmy, D. Meyer and L. Wei, "Protocol Independent Multicast Dense Mode Specifications," Internaet-Draft:draft-ietf-pim-v2-dm-01.txt, Nov. 1998.

[3] J. Moy, "Multicast Extension to OSPF," RFC 1584, Mar. 1994.

[4] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," Internet-Draft: draft-holbrook-ssm-00.txt, Mar. 2000.

[5] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma and L. Wei, "Protocol Independent Multicast-Sparse Mode(PIM-SM): Protocol Specification," RFC2362, June 1998.

[6] R. Perlman, C. Lee, A. Ballardie, J. Crowcroft, Z. Wang and T. Maufer, "Simple Multicast : A Design for Simple, Low-Overhead Multicast," Internet-Draft:draft-perlman-simple-multicast-01.txt, Nov. 1998.

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

[8] A.J Ballardie, "Core Based Trees (CBT version 2) Multicast Routing-Protocol Specification," RFC2189, Sept. 1997.

[9] M. Faloutsis, A. Banerjea and R. Pankaj, "QoSMIC : Quality of Service Sensitive Multicast Internet protocol," ACM SIGCOMM'98, Sept. 1998.

[10] A. Shaikh, et al., "Destination Driven Routing for Low Cost Multicast," IEEE JSAC Vol. 15, No. 3, Apr. 1997.

[11] L. Guo, et al., "QDMR: an Efficient Qos Dependent Multicast Routing Algorithm," Proceeding of 5th IEEE Real-time Technology and Application Symposium (RTAS '99), Vancouver, Canada, June 1999.

[12] Seok J. Koh, Myung K. Shin, Jong H. Yi, Jin H. Hahm and Chee H. Park, "Non-Core based Shared Tree Architecture for IP Multicasting," Electronic Letters, Vol.35 , No.11, pp. 872-873, May 1999.

[13] Kenneth L. Cavert, Ellen W. Zegura and Michael J. Donahoo, "Core Selection Methods for Multicast Routing," Techinal Report GA30332-0280, http://www.cc.gatech.edu/tech_reports, College of Computing, Georgia Tech.

[14] Seok-Joo Koh, Shin-Gak Kang and Ki-Sjik Park, "Enhanced Cores Based Tree for Many-to-Many

IP Multicasting," Telecommunications Review Vol.11, N0.3, pp. 485-493, May-June 2001.
[15] Seok-Joo Koh, Shin Gak Kang, "Enhancement of the CBT Multicast Routing Protocol," ICPADS2001 (published by IEEE Computer Society), 26-29 June, Kyungju, Korea 2001.
[16] D. Knuth, "The Art of Programming : Fundamental Algorithms," Addison-Wesley, Vol.1, pp. 386-388, 1968.
[17] Frank Harary, "Graph Theory(2nd edition)," Addison-Wesley, pp. 35-36, 1971.
[18] Robert Sedgewick, "Algorithms(2nd edition)," Addison-Wesley, pp. 456-464, 1988.
[19] Ellen W. Zegura, Kenneth Calvert and M. Jeff Donahoo. "A Quantitative Comparison of Graph-based Models for Internet Topology," IEEE/ACM Transactions on Networking, Dec. 1997.

서 현 곤
1992년 경성대학교 전산통계학과 졸업(이학사). 1994년 경성대학교 전산통계학과 대학원 졸업(이학석사). 2000년 영남대학교 컴퓨터공학과 대학원 박사수료 1994년~1997년 (주)동양에레베이터 기술연구소 주임연구원. 1997년~2001년 김천대학 컴퓨터정보처리계열 겸임전임강사. 2001년~현재 대구대학교 정보통신공학부 BK21교수 관심분야는 애드 혹 네트워크, 멀티캐스팅, 라우팅 프로토콜, 웹기반 응용

김 기 형
1990년 한양대학교(공학사). 1992년 한국과학기술원(공학석사). 1996년 한국과학기술원(공학박사). 2001년 AdForce, Inc 선임연구원. 1997년~현재 영남대학교 컴퓨터공학과 교수. 관심분야는 Ad Hoc Networks, Multicasting, Simulation, Embedded System