논문 04-29-11B-12 한국통신학회논문지 '04-11 Vol.29 No.11B

실시간 미디어 전송을 위한 응용계층 멀티캐스트 트리 구성 알고리즘

정회원 송 황 준*, 이 동 섭*

Application Layer Multicast Tree Constructing Algorithm for Real-time Media Delivery

Hwangjun Song*, Dong Sup Lee** *Regular Members*

요 약

최근까지 네트워크 계층에서 수행되는 IP 멀티캐스트는 많은 관심과 연구가 진행되고 있다. 하지만 유니캐스트 라우터들로 구성된 현재의 인터넷 망에서 IP 멀티캐스트의 적용은 불가능한 상태이다. 때문에 응용계층 멀티캐스트가 IP 멀티캐스트의 대안으로 제시되고 있다. IP 멀티캐스트가 네트워크 라우터들에 의존적인 반면 응용계층 멀티캐스트는 네트워크 계층과 독립적으로 수행된다. 본 논문에서는 실시간 미디어의 효과적인 전송을 위한 source에서 end-system들에 이르는 평균 지연 시간을 최소화하는 응용계층 멀티캐스트 트리 구성 알고리즘을 제안한다. 제안하는 알고리즘은 제어 변수로써 각 end-system들의 계산 수행능력과 네트워크 조건을 고려하며 트리를 구성하는 몇몇 end-system들에게만 부하가 집중되는 현상을 방지하도록 구성되었다. 제안하는 알고리즘에 의한 응용계층 멀티캐스트 트리는 clustering과 변형된 Dijkstra 알고리즘에 의해 구성된다. 즉, source와 proxy-sender들 사이의 트리와 각 cluster안에서 트리를 구성함으로써 전체 트리를 생성한다. 실험을 통하여 제안하는 알고리즘이 기존 알고리즘 보다 효과적임을 보였다.

Key Words : Overlay multicast, IP multicast, Real-time media, Time delay

ABSTRACT

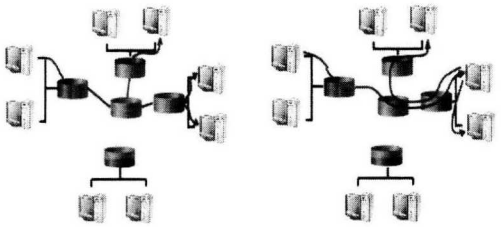
This paper presents an application layer multicast tree constructing algorithm to minimize the average time delay from the sender to end-systems for the effective real-time media delivery. Simultaneously, the proposed algorithm takes into account the computing power and the network condition of each end-system as a control variable and thus avoids the undesirable case that loads are concentratedto only several end-systems. The multicast tree is constructed by clustering technique and modified Dijkstra's algorithmin two steps, i.e. tree among proxy-senders and tree in each cluster. By the experimental results, we show that the proposed algorithm can provide an effective solution.

Ⅰ. 서 론

현재의 인터넷은 정보교환은 물론 멀티미디어 통신의 중추적 역할을 담당하고 있으며 지속적인 인터넷 사용자의 증가로 인해 인터넷을 통한 다양한 멀티미디어 서비스의 요구도 급격히 증가하고 있다. 멀티미디어 서비스에 있어서 멀티캐스트는 다양한 멀티미디어 서비스의 제공과 동시에 네트워크 효율 관점에서 매우 중요한 문제다[1]. 최근까지도 네트워크 계층에서 멀티캐스트 라우터에 의해 수행되는

* 홍익대학교 전파통신공학과 멀티미디어 통신 시스템 연구실(hwangjun@wow.hongik.ac.kr),
논문번호 : 040040-0202, 접수일자 : 2004년 2월 2일
※ 이 논문은 2004학년도 홍익대학교 교내연구비에 의하여 지원되었음.

IP 멀티캐스트는[2] 많은 연구와 관심을 갖는다. 하지만 현재 인터넷을 구성하는 대다수의 유니캐스트 라우터들은 IP 주소의 class D 주소를 인식하지 못한다. 때문에 IP 멀티캐스트가 실행되기 위해서는 인터넷 망을 구성하는 모든 라우터를 멀티캐스트 라우터로 교체해야만 한다. 이러한 문제로 인해 IP 멀티캐스트의 많은 관심에도 불구하고 실제 네트워크 망에서의 적은 시간과 경제적인 면에서 불가능한 실정이다. 최근 이러한 IP 멀티캐스트의 문제점들로 인해 현재의 IP 네트워크 환경에서 실현시킬 수 있는 응용계층 멀티캐스트(Application Layer Multicast or Overlay Multicast)가 대안으로 제시되고 있다. IP 멀티캐스트는 기존 라우터들을 멀티캐스트 라우터로 변경해야만 하는 반면 응용계층 멀티캐스트는 기존 라우터들을 변경할 필요가 없다. 또한 하드웨어와 네트워크 기술의 급속한 발전은 end-system의 처리속도와 네트워크 조건을 향상시키게 되었으며 응용계층 멀티캐스트는 현재 네트워크 상황에 더욱 적합하게 되었다. 만일 응용계층 멀티캐스트를 구성하는 end-system들의 처리속도와 저장능력, 그리고 네트워크 조건과 같은 자원들을 효과적으로 사용할 수 있다면 인터넷을 이용한 멀티캐스트의 효율을 높일 수 있다.



(a) (b)

그림 1. IP 멀티캐스트와 응용계층 멀티캐스트의 비교: (a) IP 멀티캐스트, (b) 응용계층 멀티캐스트

IP 멀티캐스트와 응용계층 멀티캐스트의 구조는 그림 1의 (a), (b)와 같다. 그림 1의 (a)는 IP 멀티캐스트의 구성으로 라우터들은 전달받은 멀티캐스트 패킷을 분류하고 다음 라우터나 멀티캐스트 그룹에 가입한 end-system들로 패킷을 동시에 전달한다. 때문에 IP 멀티캐스트 프로토콜[10]을 사용하는 경우 같은 패킷을 반복해서 전송할 필요가 없다. 그림 1의 (b)는 멀티캐스트에 가입한 end-system들 중 일부를 IP 라우터와 같은 역할을 수행하도록 할뿐 라우터 자체에 어떠한 변화도 요구하지 않는다. 응용계층 멀티캐스트 트리의 생성과 유지를 위한 연구로 [1]에서 제안한 Narada는 자체편성(self-organizing)과 분산방식(fully distributed manner)을 통해 멀티캐스트 그룹에 가입한 end-system들로 Overlay 트리를 구성한다. [3]은 Delaunay Triangulations과 Hypercube 알고리즘을 통해 라우팅 프로토콜의 도움 없이 다음 홉을 계산하는 응용계층 멀티캐스트 트리 구성을 제안한다. [4]는 분산과 확장성을 고려하여 인접한 노드들의 위치 관계에 따라 계층적인 구성을 위한 방안을 제시하며, [6]에서는 원격 전송단에서 지역적인 부분 망으로 유니캐스트 전송 후 부분 망에서의 멀티캐스트 전송 방식을 제안한다. 그 외 응용계층 멀티캐스트를 위한 여러 알고리즘이 제안되고 있다[7, 8, 9, 12].

앞서 언급한 많은 장점에도 불구하고 응용계층 멀티캐스트는 고려되어야할 몇 가지 문제들을 갖는다. 첫째로, 응용계층 멀티캐스트는 IP 멀티캐스트와 동등한 수행능력을 갖지 못한다. 즉, IP 멀티캐스트와 비교해서 응용계층 멀티캐스트는 중복된 트래픽을 더 많이 발생시킬 수 있다. 패킷의 중복은 네트워크의 대역폭을 낭비하는 결과를 초래한다. 하지만 응용계층에서 발생되는 중복된 트래픽 양은 모든 end-system들까지 유니캐스트 전송을 하는 경우보다 상당히 적다. 둘째로, 멀티캐스트에 있어 더욱 중요한 문제는 지연 시간이다. 응용계층 멀티캐스트에서 최종 목적지에 도착하기까지 트래픽은 경로상의 몇몇 end-system들을 경유해야만 하기 때문에 IP 멀티캐스트에 비해 지연 시간이 증가하게 된다[1]. 지연 시간의 증가는 실시간 미디어 전송에 있어 중요한 문제가 된다. 때문에 본 논문에서는 효과적인 실시간 미디어 전송을 위해 송신단에서 멀티캐스트 멤버들 사이의 평균 지연 시간을 최소화하는 응용계층 멀티캐스트 트리 구성 알고리즘을 제안한다. 본 논문은 다음과 같이 구성된다. 제 2 장에서 멀티캐스트 트리 구성에 있어 고려되어야 할 문제들과 효과적인 멀티캐스트 트리 구성 알고리즘을 제안하며, 제 3 장에서 제안하는 알고리즘의 성능 평가와 실험 결과를 제시하고, 마지막 제 4 장에 결론을 기술하였다.

Ⅱ. 응용계층 멀티캐스트 트리 구성 알고리즘

이 장에서는 앞에서 언급한 바와 같이 송신단에서 멀티캐스트 멤버들 사이의 평균 지연 시간을 최소화하는 응용계층 멀티캐스트 트리 구성 알고리즘

을 고려한다. RTT(round trip time)를 시간 지연의 척도로 사용하며, 제어변수로 각 end-system들의 처리 능력과 네트워크 조건을 다룬다. 즉, end-system에서 제공할 수 있는 최대 스트림 수가 그 system의 제한된 자원으로 결정된다.

우선, 문제의 접근을 위해 다음의 가정을 한다.

① 멀티캐스트 그룹 멤버들간의 RTT는 ping명령을 통해 알수 있다.

② 각 end-system들에서 조절되는 스트림 수는 모든 멤버들에게 공고된다.

하지만, 위의 가정만으로 송신단으로부터 멀티캐스트 멤버들 사이의 평균 지연 시간을 최소화하는 최적의 멀티캐스트 트리를 찾기란 불가능하다. 그 이유는 응용계층 멀티캐스트는 end-system들로 구성되며 이러한 end-system들의 구성 관계는 full mesh 형태로 고려되어야 하기 때문에, end-system의 수에 따라 구성 가능한 멀티캐스트 트리의 수가 지수적으로 증가하기 때문이다.

II-1. 트리 구성을 위한 수식화

문제접근을 위한 시나리오는 다음과 같다. 멀티캐스트에 참여하는 end-system들은 RTT값에 따라 몇 개의 cluster로 나뉜다. 이 후 각 cluster의 end-system들 중 하나가 IP 라우터와 같은 역할을 수행하는 proxy-sender로 선택되며 cluster안의 다른 end-system들은 proxy-sender로부터 데이터를 전달받는다. 이처럼 proxy-sender가 원격지의 source 또는 상위 proxy-sender로부터 하나의 스트림을 전달받은 후 가까운 지역 안에 위치한 다른 end-system들에게 데이터를 전달하는 방식이 제한적인 전송능력을 갖는 원격 source로부터 모든 end-system들이 직접 데이터를 수신하는 방식보다 더 효과적이다. Proxy-sender의 설정 이후 source와 proxy-sender들로 구성된 트리와 각 cluster들 안에서 구성되는 트리는 지연시간을 최소화할 수 있는 전체 트리를 생성한다. 그림 2는 제안된 알고리즘에 의한 구성 예다. 이 경우 RTT값은 다음의 관계를 갖는다.

$$RTT^{s}_{p_i} = \begin{cases} RTT^{s}_{p_i} & : proxy-sender가\ source와\ 직접\ 연결된\ 경우 \\ RTT^{s}_{p_1} + RTT^{p_1}_{p_2} + \ldots + RTT^{p_{k-1}}_{p_k} + RTT^{p_k}_{p_i} & : 그\ 외의경우 \end{cases}$$

위 식에서 $RTT^{s}_{p_i}$는 소스와 proxy-sender p_i 사이의 RTT값이며 $RTT^{p_{k-1}}_{p_k}$는 source로부터 최하위 proxy-sender까지의 경로 상에 있는 인접한

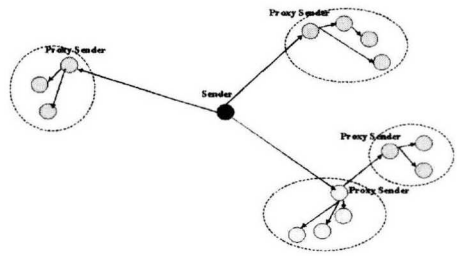


그림 2. 제안하는 응용계층 멀티캐스트 트리 구성 예

proxy-sender들 사이의 RTT값이다. 또한 source로부터 임의의 end-system까지 RTT는 다음과 같다.

$$RTT^{s}_{j} = RTT^{s}_{p_i} + RTT^{p_i}_{j}$$

여기에서 $RTT^{p_i}_{j}$는 i번째 cluster안의 j번째 end-system과 proxy-sender p_i 사이의 RTT값이다. 따라서 위 두 수식에 의한 전체 RTT 합은 다음과 같다.

$$\sum_{i=1}^{n_s} RTT^{s}_{p_i} + \sum_{i=1}^{n_s} \sum_{j=1}^{m_i - 1} \left(RTT^{s}_{p_i} + RTT^{p_i}_{j} \right)$$

위 식에서 는 구성된 cluster의 개수며 는 번째 cluster에 포함된 end-system들의 개수다. 따라서 평균 RTT는 수식 (1)과 같이 정리할 수 있으며 평균 RTT를 최소화하기 위한 방법은 다음과 같다.

II-1-1. Cluster의 수와 트리 결정을 위한 수식화

$$\frac{1}{N} \sum_{i=1}^{n_s} m_i RTT^{s}_{p_i} + \frac{1}{N} \sum_{i=1}^{n_s} \sum_{j=1}^{m_i - 1} RTT^{p_i}_{j}, \qquad (1)$$

$$s_s \le ST_s \ \text{ and } \ s_i \le ST_i$$

수식 (1)의 N은 멀티캐스트에 참여하는 end-system들의 개수이며 $\left(N = \sum_{i=1}^{n_s} m_i \right)$, s_s와 s_i는 각각 source에서 사용된 스트림 수와 i 번째 end-system에서 사용된 스트림 수이다. 또한 ST_s와 ST_i는 각각 source의 최대 스트림 수와 i 번째 end-system의 최대 스트림 수이다. 최적 해를 얻기 위해서는 모든 end-system들 사이의 관계를 full mesh 방법을 통해 고려해야만 하지만 end-system들의 개수에 따라 구성 가능한 트리의 경우는 지수적으로 증가하기 때문에 우리가 원하는 최적의 해를 찾기란 거의 불가능하다. 따라서 계산의 복잡도를 줄이기 위해 수식 (1)의 두 항을 독립적인 관계로 가정한다. 실제적으로 $RTT^{s}_{p_i} \gg RTT^{p_i}_{j}$의 관계일 경우 수식 (1)의 두 항은 서로 독립적인 관계를 갖는

다. 이와 같은 가정 하에 수식 (1)은 보다 단순한 형태를 가질 수 있다.

II-1-2. Cluster의 수와 트리 결정을 위한 단순화된 수식

가) cluster의 수와 cluster들 사이의 트리 결정

$$\frac{1}{N}\sum_{i=1}^{n_s} m_i RTT^s_{p_i}, \quad s_s \le ST_s \tag{2}$$

나) 각 cluster안에서의 트리 결정

$$\sum_{j=1}^{m_i-1} RTT^{p_i}_j, \ (\text{for } 1 \le i \le n_p), \quad s_i \le ST_i \tag{3}$$

수식 (2)와 (3)은 각각 cluster들 사이의 트리와 cluster 내부의 트리 생성을 의미한다. 즉, 수식 (2)는 각 cluster가 포함하는 end-system들의 개수에 따라 평균 RTT를 최소화 하는 source와 proxy-sender들 사이의 트리를 의미하고 수식 (3)은 각각의 cluster안에서 평균 RTT를 최소화 하는 트리를 의미한다. 최적의 해를 얻기 위해서 다음 사항들을 고려해야만 한다. 첫째, end-system들을 clustering하는 방법. 둘째, 각 cluster의 proxy-sender 설정 방법. 셋째, source와 proxy-sender 사이의 트리 구성 방법. 넷째, 각 cluster들 안에서의 트리 구성 방법. 이들 고려 사항들에 대해 다음 장에서 구체적인 내용을 기술한다.

II-2. Clustering과 트리 구성 알고리즘

이 장에서는 clustering 알고리즘과 각 cluster에서의 proxy-sender 설정에 관하여 기술하며 각 cluster 안에서 그리고 proxy-sender들 사이의 효과적인 트리 구성을 위한 변형된 Dijkstra 알고리즘을 구체적으로 제시한다.

II-2-1. Clustering 알고리즘과 proxy-sender 선택

멀티캐스트 멤버들의 clustering 방법과 각 cluster에서의 proxy-sender 선택문제는 제안하는 알고리즘의 성능에 상당한 영향을 갖는다. 본 논문에서는 clustering을 위해 k-mean clustering[9] 알고리즘을 적용하였다. k-mean clustering 알고리즘은 이미지 프로세싱 분야에서 이미 잘 알려진 알고리즘 중 하나다. clustering 알고리즘의 적용에 있어 cluster의 수를 결정하야만 하며 이를 위해 source에서 제공할 수 있는 스트림의 수를 고려해야만 한다. 때문에 cluster의 수는 그림 5와 같이 반복적인 방법을 통해 결정되어 진다.

최적의 proxy-sender 선택을 위해서는 수식 (1)의 두 항을 동시에 고려해야만 한다. 하지만 이와 같은 경우 고려해야할 경우의 수가 상당히 증가하기 때문에 각 cluster들 안에서 평균 RTT를 최소화하는 proxy-sender를 선택하기란 불가능하게 된다. 본 논문에서는 실험을 통해 source로부터 최소의 RTT를 갖고 동시에 스트림 수가 최대인 end-system을 proxy-sender로 선택하는 경우 최적해에 가장 가깝다는 결론을 얻을 수 있었다. 따라서 본 논문에서는 멀티캐스트에 참여하는 모든 멤버들이 거의 비슷한 크기의 스트림을 제공한다는 가정 하에 source 또는 상위의 proxy-sender에 가장 가까운 end-system을 proxy-sender로 선택한다.

II-2-2. 변형된 Dijkstra 알고리즘에 의한 트리

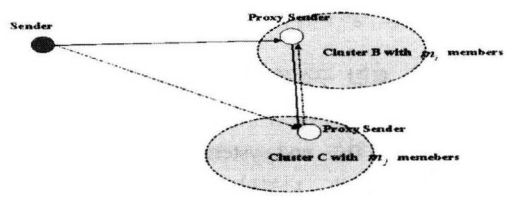


그림 3. Cluster의 위치에 따른 proxy-sender 사이의 관계

RTT를 최소화하기 위해 네트워크 계층 라우팅 알고리즘 중 하나의 노드에서 다른 모든 노드들에 이르는 최단경로를 결정하는 Dijkstra 알고리즘[10]을 적용할 수 있다. 하지만 제안하는 알고리즘은 end-system들의 스트림 조건을 고려해야 하므로 링크 비용만을 기준으로 계산되는 Dijkstra 알고리즘을 직접 적용할 수는 없다. 더욱이 proxy-sender의 RTT가 ΔRTT 만큼 변하는 경우 m_i 개의 멤버수를 갖는 i 번째 cluster의 RTT합은 $m_i \Delta RTT$ 만큼 변하게 되므로 source와 proxy-sender 사이의 링크 비용은 cluster의 크기에 따라 조절되어야만 한다. 그러므로 Dijkstra 알고리즘의 적용 이전에 source와 proxy-sender 사이의 RTT값에는 $1/m_i$ 만큼의 가중치가 부여되어야만 한다. 이를 이후부터 최소가중 RTT (minimum weighted RTT)라 한다. 기본적으로 최소가중 RTT를 적용한 proxy-sender들은 RTT 합을 최소화하기 위해 source 또는 상위 proxy-sender에 연결하여 트리를 구성하게 된다. 하지만 이러한 최소가중 RTT만으로 항상 최적의 트리를 구성할 수는 없다. 그 예로 그림 3에서와 같이

두개의 cluster가 위치한 경우에 대해 고려해 볼 수 있다.

만일 두 cluster 사이의 관계가 $\frac{RTT^{s}_{p_i}}{m_i} < \frac{RTT^{s}_{p_j}}{m_j}$ 이고 두 proxy-sender 사이의 RTT가 $RTT^{p_j}_{p_i}$ ($RTT^{p_j}_{p_i} = RTT^{p_i}_{p_j}$ 일 경우)라 한다면 평균 RTT를 최소화하기 위해 어떤 proxy-sender가 source와 연결되어야 하는지 고려해야만 한다. 그림 3에서 트리가 실선을 따라 구성된 경우의 RTT 합은 $m_i RTT^{s}_{p_i} + m_j(RTT^{s}_{p_i} + RTT^{p_j}_{p_i})$ 와 같고, 점선으로 구성된 경우의 RTT 합은 $m_j RTT^{s}_{p_j} + m_i(RTT^{s}_{p_j} + RTT^{p_j}_{p_i})$ 와 같다. 앞의 두 수식에 의해 다음과 같은 관계식을 정리할 수 있다.

① $m_i RTT^{s}_{p_i} + m_j(RTT^{s}_{p_i} + RTT^{p_j}_{p_i}) < m_j RTT^{s}_{p_j} + m_i(RTT^{s}_{p_j} + RTT^{p_j}_{p_i})$일 경우,

proxy-sender p_i 는 source 또는 상위 proxy-sender와 연결되어야 한다.

② $m_i RTT^{s}_{p_i} + m_j(RTT^{s}_{p_i} + RTT^{p_j}_{p_i}) > m_j RTT^{s}_{p_j} + m_i(RTT^{s}_{p_j} + RTT^{p_j}_{p_i})$일 경우,

proxy-sender p_j 는 $\frac{RTT^{s}_{p_i}}{m_i} < \frac{RTT^{s}_{p_j}}{m_j}$ 일 경우라 하더라도 source 또는 proxy-sender에 연결되어야만 한다.

①, ②의 부등식을 이용하여 최소가중 RTT의 적용 후 proxy-sender p_i 의 링크 비용이 p_j 보다 작음에도 불구하고 p_j 가 source 또는 상위 proxy-sender에 연결되어야만 하는 조건은 다음과 같다.

$$RTT^{p_j}_{p_i} < \frac{m_i + m_j}{m_i - m_j}(RTT^{s}_{p_j} - RTT^{s}_{p_i})$$

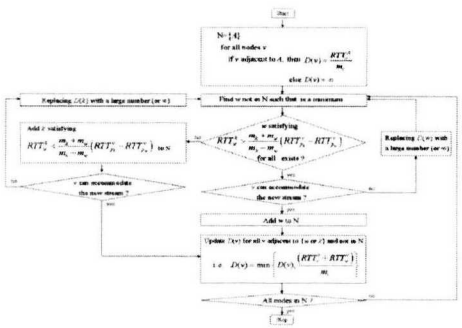


그림 4. 변형된 Dijkstra 알고리즘 순서도

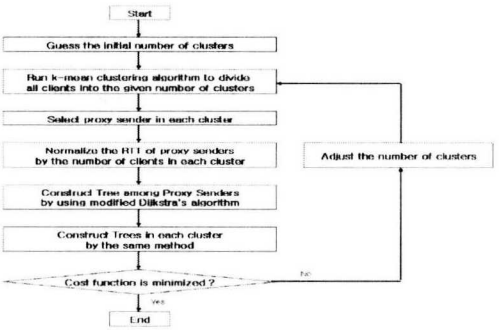


그림 5. 제안하는 응용계층 멀티캐스트 트리 구성 순서도

따라서 Dijkstra 알고리즘을 적용하기 위해 앞서 언급한 조건과 고려 사항들을 추가하여 변형된 Dijkstra 알고리즘을 구성하였다. 제안하는 변형된 Dijkstra 알고리즘은 그림 4의 순서에 따라 진행된다. 그림 4에서 N은 source로부터 최단거리 경로가 명확히 알려진 노드들의 집합이며, RTT^{A}_{v}는 A와 v 사이의 RTT를 의미한다. 또한 m_v는 v번째 cluster의 멤버 수를 나타내고 $D(v)$는 source로부터 최소 가중 RTT를 갖는 목적지 v까지 링크 비용을 나타낸다. 그림 4의 순서에 따라 proxy-sender들 사이에서와 cluster 내에서 각각 효과적인 멀티캐스트 트리를 구성한다. 또한 본 논문에서 제안하는 평균 지연 시간을 최소화하는 응용계층 멀티캐스트 트리는 그림 5의 순서를 통해 구성된다.

Ⅲ. 실험 결과

이 장에서는 트리 구조와 평균지연 시간 그리고 계산 수행의 복잡도 관점에서 제안하는 알고리즘의 성능을 비교하였다. 즉, 성능평가 척도로써 평균 RTT와 CPU-time, 그리고 각 end-system들에 할당된 스트림 수를 사용하였다. 제안한 알고리즘의 성능은 멤버들의 분포에 따라 영향을 받을 수 있기 때문에 다음 두 가지 경우에 대해 실험을 수행하였다. 각 멤버들 사이의 RTT 값에 따라 ① end-system들의 위치가 밀집된 분포를 이루는 경우와 ② 그 반대로 산재된 경우에 따라 각각 실험하였으며, 네트워크 상황을 고려하여 제안하는 알고리즘과 유사한 expanded ring 알고리즘[12]과 성능을 비교하였다. 실험의 복잡도를 줄이기 위하여 모든 멤버들은 비슷한 계산 수행능력과 네트워크 조건, 그리고 비슷한 수의 스트림을 제공할 수 있는 것으로 가정하였다.

Ⅲ-1. End-system들의 위치가 밀집된 경우

그림 6은 멀티캐스트에 참여한 end-system들이 밀집되어 분포된 경우를 나타내며 source의 스트림 수는 3으로 가정하였다.

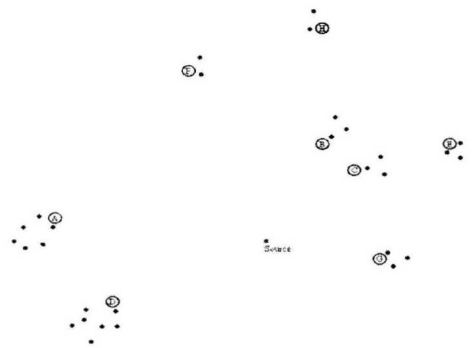


그림 6. 멀티캐스트에 참여한 29개의 end-system들을 8개로 Clustering한 결과

최적 성능은 멤버들을 8개의 cluster로 그룹 지었을 때 형성된다. 그리고 각 cluster 안에서 source와 가장 가까운 위치의 end-system이 proxy-sender로서 선택된다. 그림 7은 proxy-sender들 사이에서 제안한 변형된 Dijkstra 알고리즘을 수행한 결과를 테이블로 나타내었고, 그림 8은 cluster B에서 적용된 변형된 Dijkstra 알고리즘의 결과를 테이블로 나타내었다.

Step	N	D(A), P(A)	D(B), P(B)	D(C), P(C)	D(D), P(D)	D(E), P(E)	D(F), P(F)	D(G), P(G)	D(H), P(H)
0	S (server)	120,S	[71,S]	**70,S**	93,S	[114,S]	[192,S]	[70,S]	[124,S]
1	SC	120,S	96,C		**93,S**	116,C	178,C	120,C	157,C
2	SCD	120,S	**96,C**			116,C	178,C	120,C	157,C
3	SCDB	120,S				**116,C**	178,C	120,C	157,C
4	SCDBE	**120,S**					178,C	120,C	157,C
5	SCDBEA						178,C	**120,C**	157,C
6	SCDBEAG						179,B		**161,B**
7	SCDBEAGH						**179,B**		
8	SCDBEAGHF								

그림 7. Proxy-sender들 사이의 변형된 Dijkstra 알고리즘 테이블

Step	N	$D(A_2), P(A_2)$	$D(A_3), P(A_3)$	$D(A_4), P(A_4)$	$D(A_5), P(A_5)$	$D(A_6), P(A_6)$
0	ProxySender (P_A)	[29, P_A]	16, P_A	**9, P_A**	20, P_A	12, P_A
1	$P_A A_4$	29, A_4	16, P_A		20, P_A	**12, P_A**
2	$P_A A_4 A_6$	28, A_6	**16, P_A**		20, P_A	
3	$P_A A_4 A_6 A_3$	25, A_3			**20, P_A**	
4	$P_A A_4 A_6 A_3 A_5$	**25, A_3**				
5	$P_A A_4 A_6 A_3 A_5 A_2$					

그림 8. Cluster B에서의 변형된 Dijkstra 알고리즘 테이블

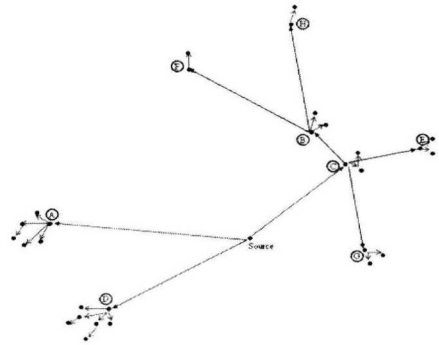


그림 9. 밀집분포상태에서 제안한 알고리즘에 의한 응용계층 멀티캐스트 트리

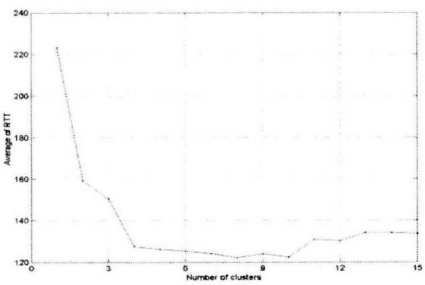


그림 10. 밀집분포상태에서 cluster의 수에 따른 평균 RTT

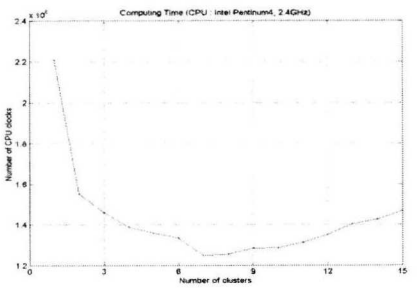


그림 11. 밀집분포상태에서의 cluster수에 따른 CPU-time

그림 9는 그림 7, 8의 결과와 같이 변형된 Dijkstra 알고리즘에 의해 생성된 응용계층 멀티캐스트 트리를 나타낸다. 그림 9를 통해 형성된 멀티캐스트 트리에서는 몇몇 멤버들에 부하가 집중되지 않고 각 멤버들에게 고르게 분산되어 있는 것을 볼 수 있다.

그림 10과 그림 11은 각각 cluster 수에 따른 평균 RTT와 CPU-time을 나타내며 이들을 통해서

cluster의 수가 6에서 10사이에 계산 수행의 복잡도가 가장 작으며 평균 RTT를 최소화할 수 있다.

Ⅲ-2. End-system들의 위치가 산재된 경우

그림 12는 그림 6과 비교해서 end-system들의 위치가 산재해 있는 경우를 나타낸다. source의 스트림 수는 3으로 밀집 분포의 경우와 같다. 그림 12에서와 같이 멤버들은 7개의 cluster로 그룹지어 지며, 각 cluster에서 source에 가장 가까운 end-system이 proxy-sender로 선택되었다.

앞 절의 밀집 분포형태와 같은 방법으로 proxy-sender 사이의 트리와 각각의 cluster안에서 트리가 생성되었다. 그림 13과 그림 14는 각각 proxy-sender 사이의 테이블을 나타내며 해당하는 응용계층 멀티캐스트 트리의 구성은 그림 15와 16을 통해 알 수 있다. 그림 14와 16은 제안한 변형된 Dijkstra 알고리즘을 수행한 결과이며 그림 13과 15는 최소가중 RTT만을 적용하여 응용계층 멀티캐스트 트리를 구성한 결과를 나타내고 있다. 그림 13과 14의 테이블 Step 3을 보면 변형된 Dijkstra 알고리즘에 의해 proxy-sender G가 N에 포함되는 경우 구성되는 전체 트리 구조와 최소가중 RTT만으로 proxy-sender C가 N에 포함되는 경우 구성되는 전체 트리의 구조가 분명한 차이를 보이고 있다. 이러한 차이는 그림 17을 통해 더욱 분명해진다. 그림 17은 최소가중 RTT 만으로 proxy-sender를 선택하는 경우에 비해 변형된 Dijkstra 알고리즘을 적용하는 경우 생성되는 트리의 평균 RTT 값을 줄일 수 있음을 나타내고 있다. 또한 cluster 수에 따른 CPU-time 관계를 나타내는 그림 18을 보면 cluster의 수가 6과 10사이 일 때 계산 수행의 복잡도가 가장 작으며 평균 RTT를 최소화시킬 수 있음을 보여주고 있다.

그림 12. 멀티캐스트에 참여한 28개의 end-system들을 7개로 clustering한 결과

그림 13과 14의 테이블 Step 3을 보면 변형된 Dijkstra 알고리즘에 의해 proxy-sender G가 N에 포함되는 경우 구성되는 전체 트리 구조와 최소가중 RTT만으로 proxy-sender C가 N에 포함되는 경우 구성되는 전체 트리의 구조가 분명한 차이를 보이고 있다. 이러한 차이는 그림 17을 통해 더욱 분명해진다.

Step	N	D(A), P(A)	D(B), P(B)	D(C), P(C)	D(D), P(D)	D(E), P(E)	D(F), P(F)	D(G), P(G)
0	S (server)	110,S	117,S	179,S	[388,S]	[301,S]	[368,S]	[120,S]
1	SA		117,S	179,S	505,A	310,A	323,A	334,A
2	SAB			179,S	465,B	310,A	317,B	334,A
3	SABC				465,B	310,A	317,B	318,C
4	SABCE				465,B		317,B	318,C
5	SABCEF				465,B			318,C
6	SABCEFG				465,B			
7	SABCEFGD							

그림 13. Proxy-sender들 사이의 최소가중 RTT만을 적용한 Dijkstra 알고리즘 테이블

Step	N	D(A), P(A)	D(B), P(B)	D(C), P(C)	D(D), P(D)	D(E), P(E)	D(F), P(F)	D(G), P(G)
0	S (server)	110,S	117,S	[179,S]	[388,S]	[301,S]	[368,S]	120, S
1	SA		117,S	339,A	505,A	310,A	323,A	120, S
2	SAB			339,A	465,B	310,A	317,B	120, S
3	SABG			269,G	341,G	310,A	317,B	
4	SABGC				341,G	310,A	317,B	
5	SABGCE				341,G		317,B	
6	SABGCEF				341,G			
7	SABGCEFD							

그림 14. Proxy-sender들 사이의 변형된 Dijkstra 알고리즘 테이블

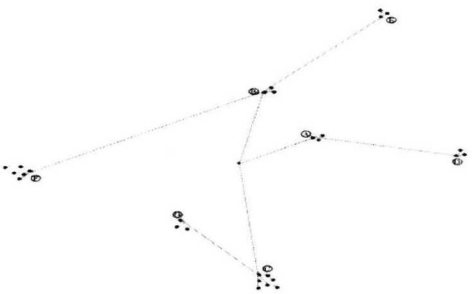


그림 15. 최소가중 RTT만을 적용한 Dijkstra 알고리즘에 의한 트리

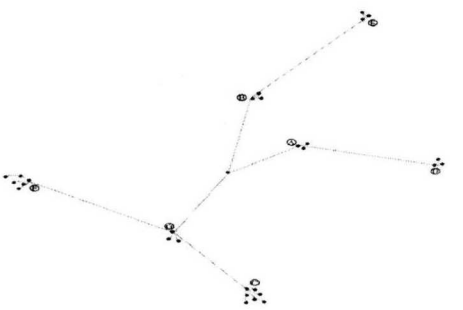


그림 16. 변형된 Dijkstra 알고리즘에 의한 트리

그림 17은 최소가중 RTT 만으로 proxy-sender를 선택하는 경우에 비해 변형된 Dijkstra 알고리즘을 적용하는 경우 생성되는 트리의 평균 RTT 값을 줄일 수 있음을 나타내고 있다. 또한 cluster 수에 따른 CPU-time 관계를 나타내는 그림 18을 보면 cluster의 수가 6과 10사이 일 때 계산 수행의 복잡도가 가장 작으며 평균 RTT를 최소화시킬 수 있음을 보여주고 있다.

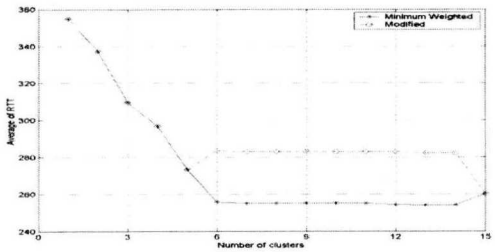


그림 17. end-system들의 산재분포일 경우 cluster수에 따른 평균 RTT

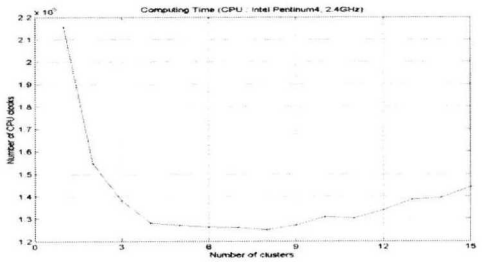


그림 18. end-system들의 산재분포일 경우 cluster수에 따른 CPU-time

III-3. Expanded Ring 알고리즘과의 성능비교

응용계층 멀티캐스트 관점에서 직접적인 성능비교를 위해 기존의 다른 알고리즘을 찾기란 쉽지 않다. 그 이유는 제안하는 알고리즘들 마다 서로 다른 네트워크 조건하에서 응용계층 멀티캐스트를 고려하기 때문이다. 이 장에서는 제안한 변형된 Dijkstra 알고리즘과 expanded ring 알고리즘[12]의 성능을 비교 실험 하였다. Expanded ring 알고리즘은 본 논문에서 제안한 알고리즘과 유사한 상황에서 고려되었으며 성능측정의 기준으로 평균 RTT를 사용하였다. 제안한 알고리즘과 expanded ring 알고리즘에 의해 구성된 트리 결과는 그림 19, 20, 21을 통해서 보여주고 있다. 또 그림 22와 23에서는 앞서 언급한 평균 RTT를 기준으로 제안한 알고리즘과

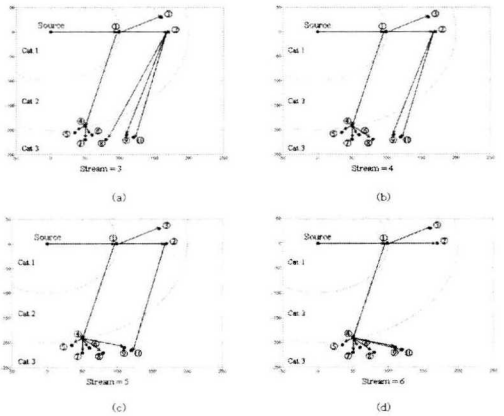


그림 19. expanded ring 알고리즘에 의한 트리: (a) end-system들의 스트림 수가 3일 경우, (b) end-system들의 스트림 수가 4일 경우, (c) end-system들의 스트림 수가 5일 경우 그리고 (d) end-system들의 스트림 수가 6일 경우

expanded ring 알고리즘의 성능을 비교하고 있다.

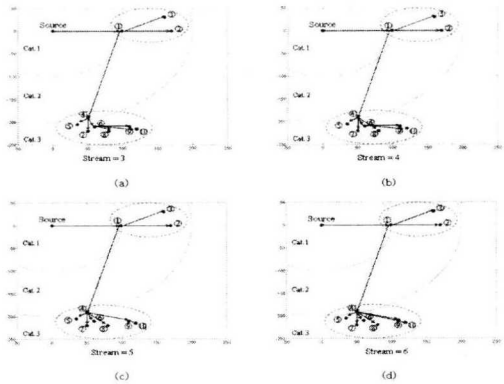


그림 20. 밀집분포에서 제안한 알고리즘에 의한 트리: (a) end-system들의 스트림 수가 3일 경우, (b) end-system들의 스트림 수가 4일 경우, (c) end-system들의 스트림 수가 5일 경우 그리고 (d) end-system들의 스트림 수가 6일 경우

end-system들의 분포가 밀집상태 또는 산재상태에 상관없이 expanded ring 알고리즘을 적용하여 멀티캐스트 트리를 구성하는 경우 그림 19에서와 같이 member들의 스트림 수에 따른 트리를 구성한다. 반면 제안한 알고리즘은 end-system들의 분포에 따라 다른 멀티캐스트 트리를 구성한다. 그림 20에서 end-system들이 상대적으로 밀집되어 분포하고 스트림 수가 5보다 작은 경우 제안한 알고리즘에 따라 end-system 8, 9, 10은 자신과 가장 가까운 end-system에 연결하지만 그림 19와 같이 expanded ring 알고리즘의 경우 원격지에 있는 end-system 1,

2에 연결되는 모습을 볼 수 있다. 이러한 이유는 제안한 알고리즘은 멤버들의 RTT 관계에 따라 clustering되고 cluster안에서 상위 멤버가 결정되는 반면 expanded ring 알고리즘에서는 정해진 category에 따라 상위 멤버가 결정되기 때문이다. 따라서 제안하는 알고리즘이 평균 RTT를 줄일 수 있다. 하지만 그림 19, 20의 (c)와 같이 스트림 수가 하위 멤버 수만큼 증가하는 경우 두 알고리즘에 따라 생성되는 트리는 같은 구성을 이루게 된다.

그림 21과 같이 end-system들이 산재해 있는 경우 expanded ring 알고리즘과의 차이는 더욱 분명해 진다. 변형된 Dijkstra 알고리즘에 의해 source로부터의 스트림이 end-system 3에 할당된다.

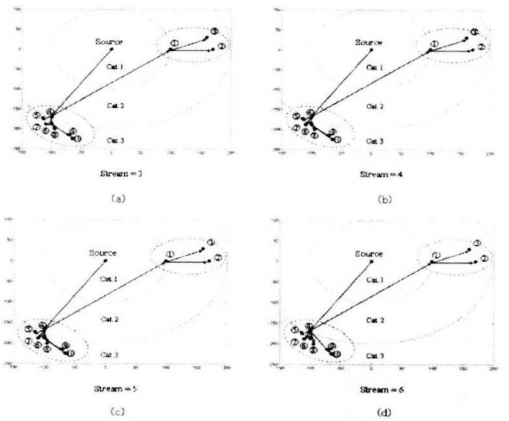


그림 21. 산재분포에서 제안한 알고리즘에 의한 트리: (a) end-system들의 스트림 수가 3일 경우, (b) end-system들의 스트림 수가 4일 경우, (c) end-system들의 스트림 수가 5일 경우 그리고 (d) end-system들의 스트림 수가 6일 경우

이 경우 제안한 알고리즘에 의한 멀티캐스트 트리의 평균 RTT가 expanded ring 알고리즘의 경우보다 상당히 감소함을 그림 22를 통해 알 수 있다. 하지만 트리 구성에 있어서 제안한 알고리즘은 반복적인 트리 구성 과정을 거치게 되므로 계산 수행의 복잡도가 expanded ring 알고리즘 보다 커질 수 있다.

Ⅳ. 결 론

실시간 미디어 전송에 있어 지연시간은 매우 중요한 Qos문제 중 하나다. 때문에 본 논문에서는 source에서 멀티캐스트에 참여한 end-system들에 이르는 평균 RTT를 최소화방법을 고려하였으며 수식

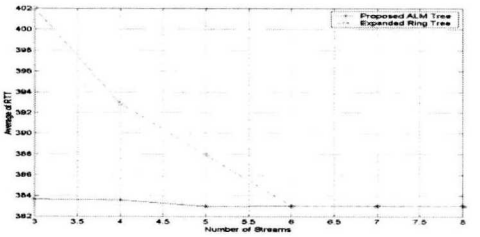


그림 22. 밀집분포에서 제안한 알고리즘과 expanded ring 알고리즘의 평균 RTT 비교

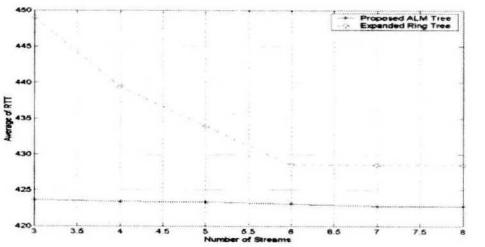


그림 23. 산재분포에서 제안한 알고리즘과 expanded ring 알고리즘의 평균 RTT 비교

적 정리를 통해 실시간 미디어 전송에 효과적인 응용계층 멀티캐스트 트리 구성 알고리즘을 제안하였다. 또한 제안한 알고리즘은 멀티캐스트 트리를 구성함에 있어 몇몇 end-system으로만 부하가 집중되는 현상을 피할 수 있도록 하기 위해 모든 end-system들의 네트워크 조건과 계산 수행능력을 고려하였다.

본 논문에서 제안한 응용계층 멀티캐스트 트리는 clustering 알고리즘과 변형된 Dijkstra 알고리즘에 의해 구성된다. 즉, source와 proxy-sender들 사이의 트리와 각각의 cluster안에서 트리를 생성하여 하나의 전체 멀티캐스트 트리를 구성한다. 비교 실험을 통하여 제안한 알고리즘이 기존의 알고리즘 보다 효과적인 멀티캐스트 트리를 구성함을 보였다. 하지만 본 논문에서 제안한 멀티캐스트 트리구성 알고리즘은 초기 트리 설정단계만을 언급하고 있다. 때문에 응용계층 멀티캐스트 트리 구성의 완벽한 해결은 생성된 멀티캐스트 트리의 효과적인 유지 및 관리에 대한 구체적인 고려가 필요하다. 인터넷 망에서의 end-system들은 고정적이지 않기 때문에 초기 트리 설정이후 유동적인 end-system들을 고려한 트리 유지 알고리즘이 연구되어져야 할 것이다.

참 고 문 헌

[1] Y. Chu, S. Rao, S. Seshan and H. Zhang, "A case for end-system multicast," ACM SIGMETRICS, Santa Clara, June 2000.

[2] S. Deering, "Host Extensions for IP Multicasting," RFC1112, August 1989.

[3] J. Liebeherr and M. Nahas, "Application-layer Multicast with Delaunay Triangulations," IEEE GLOBECOM, 2001.

[4] L. Mathy, R. Canonico, S. Simpson, and D. Hutchison, "Scalable adaptive hierarchical clustering," IEEE Communication Letter, Vol. 6, No. 3, March 2002.

[5] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application level multicast infrastructure," Proceedings of 3rd Usenix Symposium on Internet Technologies & Systefeeder (USITS 2001), San Francisco, Mar. 2001.

[6] J. Park, S. J. Koh, S. G.. Kang, D. Y. Kim, "Multicast Delivery Based on Unicast and Subnet Multicast," IEEE Communications Letters, Vol. 5 No. 4 April 2001.

[7] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A large scale and decentralized application-level multicast infrastructure," IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002.

[8] S. Y. Shi and J. S. Turner, "Multicast routing and bandwidth dimensioning in overlay network," IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002.

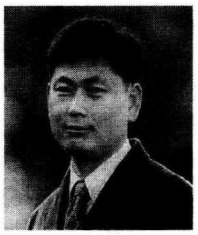
[9] S. Banerjee and B. Bhattacharjee, "Scalable secure group multicast over IP multicast," IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002.

[10] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-down approach featuring the Internet*, Addison Wesley, 2001.

[11] W. K. Pratt, *Digital Image Processing*, Wiley Interscience, 1991.

[12] C. K. Yeo, B. S. Lee and M. H. Er, "A framework for multicast video streaming over IP networks," Journal of Network and Computer Applications, pp. 273-28, Vol. 26, 2003.

송 황 준 (Hwangjun Song) 정회원



1990. 2 : 서울대학교 공과대학 제어계측과 학사

1992. 2 : 서울대학교 공과대학 제어계측과 석사

1999. 5 : EE-System, University of Southern California, Los Angeles, USA (공학 박사)

2000.9 ~ 현재 : 홍익대학교 공과대학 전자전기공학부 조교수

<관심분야> Multimedia communication/signal processing, Packet video, Network protocols necessary to implement a functional real-time image/video application

이 동 섭 (Dong Sup Lee)

2000. 2 : 홍익대학교 공과대학 전자전기공학부 학사

2004. 2 : 홍익대학교 공과대학 전파통신공학과 석사

2004. 3 ~ 현재 : 터보테크 (주) IT R&D Center 연구원

<관심분야> Multicast, Overlay networks, Peer to Peer systems, Internet and Application-software