동기화된 실시간 미디어 멀티캐스트 서비스에 적합한 오버레이 멀티캐스트 트리 구성 알고리즘

준회원 주 현 철*, 정회원 송 황 준*

Overlay Multicast Tree Construction Algorithm for Synchronized Real-time Media Multicast Service over the Internet

HyunChul Joo* *Associate Member*, HwangJun Song* *Regular Member*

요 약

본 논문은 인터넷 상에서 동기화된 실시간 미디어 멀티캐스트 서비스에 적합한 오버레이 멀티캐스트 트리를 구성하는 알고리즘을 제안한다. 제안하는 알고리즘은 멀티캐스트 송신단에서 각 종단 시스템 사이의 전송 지연에 대한 평균과 변이를 최소화 하는 트리 구성을 찾는다. 그러나 위의 문제는 NP-완전하므로, 이러한 문제의 계산 복잡도를 낮추면서, 근사해를 찾기 위한 방법으로 OGA (Orthogonal Genetic Algorithm)을 이용하였다. 실험 결과에서 제안하는 알고리즘이 기존 알고리즘에 비해 동기화된 실시간 미디어 데이터 전송을 위한 효과적인 트리를 구성한다는 것을 보인다.

Key Words : Overlay Multicast, Average Delay, Delay Variance, Internet

ABSTRACT

This work presents an effective overlay multicast tree construction algorithm for synchronized real-time media multicast service over the Internet. The proposed algorithm is designed to minimize delay variance among group members to provide the synchronized service as well as average delay of group members in order to support the service in real-time. Basically, orthogonal genetic algorithm is employed to obtain the near optimal tree with a low computational complexity since the given problem is NP-complete. Finally, experimental results are provided to show the superior performance of the proposed algorithm.

Ⅰ. 서 론

현재의 인터넷은 정보교환은 물론 멀티미디어 통신의 중추적 역할을 담당하고 있다. 멀티미디어 서비스에 있어 QoS와 멀티캐스트는 다양한 멀티미디어 서비스의 제공과 네트워크 효율 관점에서 매우 중요한 문제이다[1]. 네트워크 계층에서 멀티캐스트 라우터에 의해 수행되는 IP 멀티캐스트는[2] 전송 지연과 네트워크 효율 측면에서 데이터를 그룹에 참가하고 있는 종단 시스템들에게 분산시키기 위한 가장 효과적인 방법이다. 그러나 IP 멀티캐스트는 라우터상의 계산 복잡도와 증가된 컨트롤 오버헤드로 인해 현재 네트워크상에 적용하기에는 불가능한 실정이다. 이러한 IP 멀티캐스트의 문제점들로 인하

※ 이 논문은 2005년도 포항공과대학교 및 한국과학재단/학술진흥재단 젊은과학자연구활동지원사업 (R08-2004-000-10084-0)의 지원에 의하여 연구되었음.

* 포항공과대학교 컴퓨터공학과 멀티미디어통신시스템 연구실 ({chul1978, hwangjun}@postech.ac.kr)

논문번호 : KICS2006-07-315, 접수일자 : 2006년 10월 16일, 최종논문접수일자 : 2006년 10월 23일

여 응용계층 멀티캐스트 (Application Multicast 또는 Overlay Multicast)가 대안으로 제시되고 있다. IP 멀티캐스트와는 달리 응용계층 멀티캐스트는 현재 그룹에 참가하고 있는 종단 시스템 일부가 기존 멀티캐스트 라우터의 역할을 대신 담당하므로, 기존 라우터의 변경을 요하지 않는다. 또한 최근에 들어서 종단 시스템들의 처리속도, 저장능력, 그리고 네트워크 조건의 향상으로 인해 응용계층 멀티캐스트는 현재 네트워크 상황에 적용이 용이하다.

앞서 언급한 장점에도 불구하고 응용계층에서 최종 목적지에 도착하기 위해 트래픽은 경로상의 몇몇 종단 시스템들을 경유해야만 하므로, 응용계층 멀티캐스트는 IP 멀티캐스트에 비해서 전송 지연이 증가하게 된다. 이러한 전송 지연의 증가는 실시간 미디어 전송에 있어 중요한 문제가 되며, 이 문제를 해결하기 위해 몇 가지 효과적인 응용계층 멀티캐스트 트리 구성 알고리즘들이 제안되었다[3,4,5,6]. CPT(Compact Tree)[3]에서는 최대 지연을 최소화시키기 위한 알고리즘을 제시하였으며, 트리는 루트로부터 점진적으로 트리의 반경을 최소화 시키는 방향으로 성장한다. MSN (Multicast Service Node) [4]에서는 각각의 MSN이 관할하는 클라이언트들의 수에 따라서 각 MSN에게 동적인 우선순위를 부여함으로서 평균 지연을 최소화시키고, 최대 지연을 최소화 시키는 문제에 대해서도 쉽게 확장할 수 있는 알고리즘을 제시하였다. MDA (Modified Dijkstra's Algorithm)[5]의 저자는 평균 지연을 최소화 시키는 문제를 해결하기 위한 중앙 집중화 방식의 알고리즘을 제시하였다. 멀티캐스트 트리는 클러스터링 알고리즘과 변형된 다익스트라 알고리즘을 이용하여 구성된다. Switch Tree[6]는 전반적인 트리 비용 또는 지연시간을 최소화시키기 위해 동적으로 트리를 구성해나간다.

본 논문에서는 동기화된 실시간 미디어 멀티캐스트 서비스에 적합한 오버레이 멀티캐스트 트리를 구성하는 알고리즘을 제안한다. 멀티캐스트 송신단에서 각 종단 시스템 사이의 전송 지연의 평균과 변이를 최소화하는 것은 동기화된 실시간 미디어 멀티캐스트 서비스를 원활하게 지원하기 위해 반드시 고려되어야 하는 사항이다.

Ⅱ. 본 론

2.1 문제 정의

본 논문에서 고려하는 시나리오는 그림 1과 같다. 멀티캐스트에 참여하는 종단 시스템들은 RTT

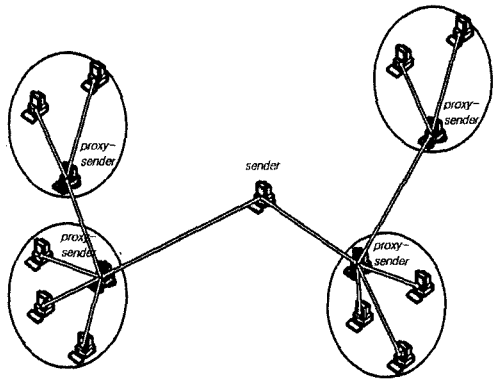


그림 1. 문제 접근을 위한 시나리오 모델

(Round-Trip-Time)값에 따라 몇 개의 클러스터로 나뉘며, 각 클러스터 내의 종단 시스템들 중 하나가 멀티캐스트 라우터와 같은 역할을 수행하는 프락시 송신자 (Proxy-sender)로 선출되어 있다고 가정하자. 선택된 프락시송신자는 송신단이나 다른 프락시송신자로부터 데이터를 전달받아 자신이 담당하는 클러스터 내의 다른 종단 시스템들에게 데이터를 전달하는 역할을 한다.

프락시송신자의 전송 지연은 다음과 같은 관계를 갖는다.

$$DELAY^{S}_{P_i} = \begin{cases} DELAY^{S}_{P_i} : \text{프락시송신자가 직접 송신단으로부터 데이터를 수신시} \\ DELAY^{S}_{P_1} + DELAY^{P_1}_{P_2} + \ldots + \\ DELAY^{P_{k-1}}_{P_k} + DELAY^{P_k}_{P_i} : \text{그 외의 경우,} \end{cases}$$

위 식에서 $DELAY^{S}_{P_i}$는 송신단 S와 프락시송신자 P_i 사이의 전송 지연이며, $DELAY^{P_{k-1}}_{P_k}$은 송신단 S에서 프락시송신자 P_i의 경로 상에 있는 인접한 프락시송신자 P_{k-1}와 P_k 사이의 전송 지연을 의미한다. 또한 송신단에서 임의의 종단 시스템 사이의 전송 지연은 다음과 같다.

$$DELAY^{s}_{j} = DELAY^{s}_{P_i} + DELAY^{P_i}_{j}, \quad (1)$$

위 식에서 $DELAY^{P_i}_{j}$은 i번째 클러스터 내의 j번째 종단 시스템과 프락시송신자 P_i 사이의 전송 지연이다. 따라서 평균 지연은 다음과 같은 관계를 갖는다.

$$\frac{1}{N}\sum_{i=1}^{n_P} DELAY^{S}_{P_i} + \frac{1}{N}\sum_{i=1}^{n_P}\sum_{j=1}^{m_i-1}\left(DELAY^{S}_{P_i} + DELAY^{P_i}_{j}\right), \quad (2)$$

위 식에서 N은 멀티캐스트 그룹에 참여하는 총 종단 시스템들의 개수이고 ($N=\sum_{i=1}^{n_P} m_i$), n_P는 구성된 총 클러스터의 개수이며, m_i은 i번째 클러스터 내에 포함된 종단 시스템들의 개수이다. 일반적으로 $DELAY_{P_i}^{S}$은 $DELAY_{j}^{P_i}$에 비해 매우 길므로, 평균 지연은 다음과 같이 근사화할 수 있다.

$$\frac{1}{N}\sum_{i=1}^{n_P} m_i * DELAY_{P_i}^{S}. \quad (3)$$

유사한 방법으로 i번째 클러스터 내의 j번째 종단 시스템의 변이는 다음과 같이 근사화할 수 있다.

$$VAR_j^S \approx VAR_{P_i}^S = \left| DELAY_{P_i}^S - \frac{1}{N}\sum_{j=1}^{n_P} m_j * DELAY_{P_j}^S \right|. \quad (4)$$

앞서 언급했듯이, 멀티캐스트 송신단에서 각 종단 시스템 사이의 전송 지연의 평균과 변이를 최소화하는 것은 동기화된 실시간 미디어 멀티캐스트 서비스에 있어 중요한 요소이다. 또한 각 종단 시스템의 처리속도, 저장능력, 그리고 네트워크 조건은 한계가 있으므로, 이러한 제약사항을 고려해야 한다. 따라서 이러한 문제를 다음과 같이 나타낼 수 있다.

문제 정의 : 다음식을 최소화하는 오버레이 멀티캐스트 트리 구성하기.

$$\frac{1}{N}\sum_{i=1}^{n_P} m_i * \{(1-\alpha) * DELAY_{P_i}^S + \alpha * VAR_{P_i}^S\} \quad (5)$$

$$subject\ to\ s_S \le ST_S\ and\ s_{P_i} \le ST_{P_i} \quad (6)$$

위 식에서 α는 상수이고 ($0 \le \alpha \le 1$), s_S와 s_{P_i}는 각각 송신단과 i번째 프락시송신자에서 사용된 스트림 수이며, ST_S와 ST_{P_i}은 각각 송신단과 i번째 프락시송신자의 최대 스트림 수이다. 식 (6)은 송신단과 각 프락시송신자의 스트림 수 제약사항과 관련된다.

위의 최적화 문제는 NP-완전하며, 분석적인 해를 얻는 것은 매우 어려운 일이므로, 제안한 알고리즘은 계산 복잡도를 낮추면서 근사해를 찾기 위해 OGA (Orthogonal Genetic Algorithm)[7, 8]을 이용하였다. OGA는 GA (Genetic Algorithm)[9] 연산중에서 교차 (Cross-over) 연산의 검색 범위를 줄이기 위해 실험계획방법 (Experimental Design Method) [10]을 적용시킨 알고리즘이다. 2절에서 실험계획방법에 대해 간략히 기술하며, 3절에서 제안한 트리 구성 알고리즘을 기술한다.

2.2 실험계획방법

몇 개의 값 (Level)들을 가질 수 있는 인자 (Factor)들을 조합하여, 목적에 맞는 최상의 조합을 발견하기 위해서는 가능한 모든 조합을 검사해야만 한다. 일반적으로 n개의 값들을 가질 수 있는 k개의 인자들이 있을 시, 모든 조합의 수는 n^k이다. 인자들의 개수와 인자들이 가질 수 있는 값의 개수가 작을 경우 가능한 모든 조합을 검사할 수 있으나, 그렇지 않을 경우 가능한 모든 조합을 검사하는 것은 어려우므로, 이들 중 대표 조합만을 검사하는 것이 바람직하다.

이러한 목적을 위해 직교 설계 (Orthogonal Design) [10]가 개발되었다. 직교 설계는 각각 다른 수의 인자와 인자들이 가질 수 있는 값에 대해 대표 조합을 나타내는 직교 배열 (Orthogonal array)을 만들어 낸다. n개의 값들을 가질 수 있는 k개의 인자들에 대한 직교 배열을 $L_m(n^k)$라고 하자. 여기서 L은 Latin-square이고, m은 대표 조합의 개수를 의미한다. 1부터 3까지 3개의 값들을 가질 수 있는 4개의 인자들에 대한 직교 배열표는 표 1과 같다. 일반적으로 많은 수의 조합을 원소로 가진 직교 배열은 목적에 맞는 더 좋은 결과를 만들어 낼 수 있으나, 큰 계산 복잡도를 요구한다. 본 논문에서는 계산 복잡도를 낮추면서 근사해를 구하기 위해 $L_9(3^4)$를 사용하였다.

표 1. $L_9(3^4)$ 직교 배열표

Combination	*Factor 1*	*Factor 2*	*Factor 3*	*Factor 4*
1st	1	1	1	1
2nd	1	2	2	2
3rd	1	3	3	3
4th	2	1	2	3
5th	2	2	3	1
6th	2	3	1	2
7th	3	1	3	2
8th	3	2	1	3
9th	3	3	2	1

2.3 OGA 기반의 트리 구성 알고리즘

제안한 시스템에서 각 클러스터 내에서 가장 많은 스트림 수를 지원할 수 있는 종단 시스템이 프락시 송신자로 선출되며, 각 프락시송신자는 주기적

인 측정을 통해 다른 프락시송신자들과 할당된 클러스터 내에 포함된 종단 시스템에 대한 최신의 그룹 정보를 유지하고 있으며, 이러한 그룹 정보를 주기적으로 송신단에게 전송한다. 그룹 정보에는 할당된 클러스터 내에 포함된 종단 시스템의 개수와 자신과 다른 프락시송신자 사이의 지연시간 정보를 포함하고 있다. 송신단은 이러한 그룹 정보를 기반으로 하여, 아래의 제안한 알고리즘을 사용하여 멀티캐스트 트리를 구성하며, 구성된 멀티캐스트 트리 정보를 각 프락시송신자에게 전송한다. 아래에서 제안한 트리 구성 알고리즘의 세부적인 각 단계와 수도코드를 기술하고 있다.

Step 1. 모든 노드는 서로서로 연결되어 있고 (Full- mesh), 그래프 상의 M개의 링크들은 각각 1부터 M까지 넘버링 되어있다고 가정하자. 멀티캐스트 트리는 다음과 같이 벡터 형태로 표현할 수 있다.

$$Multicast\ tree\ vector\ T=(e_1,e_2,e_3,\ldots,e_M),$$

$$where\ e_i=\begin{cases}1: \text{링크 } i\text{가 멀티캐스트 트리 } T\text{에 포함되어 있을시,}\\ 0: \text{그 외의 경우.}\end{cases}$$

Step 2. $e_i(1\le i\le M)$에 0 또는 1을 임의로 할당하여 초기 모집단을 이루는 K개의 멀티캐스트 트리 벡터들을 만든다.

Step 3. 모집단에서 임의로 3개의 멀티캐스트 트리 벡터를 선출한다.

Step 4. 선택되어진 3개의 멀티캐스트 트리 벡터들을 랜덤하게 4개의 서브 벡터로 나누고, 실험계획 방법을 적용하여 직교 배열을 만든다. (Orthogonal Cross-over) 예를 들어, 선택되어진 3개의 멀티캐스트 트리 벡터들이 아래와 같다고 하자.

*Multicast tree Vector*1 = {1,0,0,1,0,1}
*Multicast tree Vector*2 = {0,1,0,1,1,0}
*Multicast tree Vector*3 = {0,0,1,0,0,1}

이 멀티캐스트 트리 벡터들을 아래와 같이 랜덤하게 4개의 서브벡터로 나누고, 표 1을 통해 대표 조합인 9개의 멀티캐스트 트리벡터를 만든다. 여기서 각 서브벡터는 3개의 값을 가질 수 있는 인자에 해당한다.

	Factor 1	Factor 2	Factor 3	Factor 1
Level 1	1 0	0	1	0 1
Level 2	0 1	0	1	1 0
Level 3	0 0	1	0	0 1

Multicast tree Vector1 = {1,0,0,1,0,1}
Multicast tree Vector2 = {1,0,0,1,1,0}
Multicast tree Vector3 = {1,0,1,0,0,1}
Multicast tree Vector4 = {0,1,0,1,0,1}
Multicast tree Vector5 = {0,1,0,0,0,1}
Multicast tree Vector6 = {0,1,1,1,1,0}
Multicast tree Vector7 = {0,0,0,0,1,0}
Multicast tree Vector8 = {0,0,0,1,0,1}
Multicast tree Vector9 = {0,0,1,1,0,1}

위의 과정을 거친 후, 직교 배열 내의 각 멀티캐스트 트리 벡터에 대해 변이 (Mutation) 연산을 적용한다. 다시 말해, 멀티캐스트 트리 벡터 내의 각각의 e_i는 일정확률 p_{flip}로 0은 1로 1은 0으로 바꾼 후, 마찬가지로 전체 e_i가 일정확률 p_{toggle}로 바꾼다.

Step 5. 직교 배열 내의 각 멀티캐스트 트리 벡터는 트리가 아닐 수 있으므로, check-and-repair 연산을 통해 트리로 만든다. 세부적인 과정은 그림 2와 같다.

Step 6. 직교 배열 내의 각 멀티캐스트 트리 벡터와 모집단에서 임의로 선택한 3개의 멀티캐스트 트리 벡터에 대해 식 (5)를 적용하여 평가한다. 이 중에서 값이 가장 작은 2개의 멀티캐스트 트리 벡터는 다음 세대 모집단의 원소가 된다.

Step 7. Step 3부터 6까지 과정을 K/2번 반복한다.

Step 8. 종료 조건을 만족할 경우 알고리즘은 종료하고, 그렇지 않을 경우 Step 3으로 되돌아가서

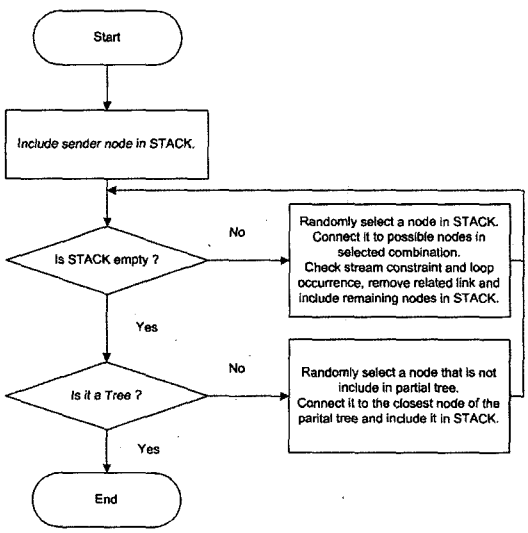


그림 2. check-and-repair 연산의 흐름도

```
Initialization:
  P = {Randomly create an initial population of K multicast tree
       vectors}
  Generation_number = 0;
Population Evolution:
  while(Generation_number < Maximum_Generation_Number) do
    P' = ø;
    while (K/2 times) do
      Randomly select three multicast tree vectors from P.
      Perform orthogonal cross-over, mutation, and check-and-
      repair operations on them.
      P'=P' ∪ {After evaluation is done, generate 2 multicast tree
              vectors for the next generation}
    End of while loop
    Generation_number = Generation_number + 1
    P = P'
  End of while loop
```

다음 세대에 대해 위의 과정을 반복한다. 종료 조건은 보통 특정 세대수 만큼 반복하거나, 현재 모집단과 다음 세대의 모집단을 비교하였을 시, 변화율이 임계값 이하가 될 때까지 진행된다.

여기서 멀티캐스트 트리 벡터의 변이에 대한 함수는 다음과 같이 정의할 수 있다.

$$VAR_{tree} = \sum_{i=1}^{n_p} m_i * VAR_{P_i}^{S}. \quad (7)$$

위 함수는 식 (5)의 2번째 항에 해당하며, 송신단과 각 종단 시스템 사이의 지연시간의 변이에 해당한다. 식 (5)를 적용한 평가값을 계산함에 있어, α값이 0일 때는 평균 지연만을 고려하나, α값이 증가함에 따라 VAR_{tree} 함수의 값이 더 많은 비중을 차지하게 된다. 마지막으로 α값이 1일 때는 VAR_{tree} 함수의 값만이 고려된다.

Ⅲ. 실험결과

이 절에서는 제안하는 알고리즘의 성능 평가에 대한 결과를 기술한다. 첫 번째 실험에서는 α값에 따른 제안한 알고리즘의 트리 구성 변화에 대해 기술하며, 두 번째 실험에서는 기존의 알고리즘들과 제안한 알고리즘을 비교 및 분석한다. 평균 지연과 위의 VAR_{tree} 함수가 알고리즘의 성능을 비교하는 척도로서 사용되었다.

실험 환경은 OPNET을 사용하여 구현하였고, 주요 구성 사항은 다음과 같다.

(1) 실시간 미디어 데이터의 프레임 속도는 30 frames/sec이고, 각 프레임의 크기는 10000 bytes로 설정하였다.

(2) 실험에서 사용된 네트워크 구성은 그림 3과 같으며, 링크 대역폭은 100 Mbps로 설정하였다.

(3) 실험에서 구성된 총 클러스터의 개수는 10개이고, 각 프락시송신자의 최대 스트림 수는 2이며, 각 클러스터 내에 포함된 종단 시스템의 개수는 각각 1에서 10사이의 임의값으로 설정하였다. 또한, 실험의 간편함을 위해 프락시송신자와 할당된 클러스터 내에 포함된 종단 시스템까지의 전송 지연은 0으로 가정하였다. (그림 3에서 괄호안의 두 개의 인자는 각각 프락시송신자의 최대 스트림 수와 할당된 클러스터 내에 포함된 종단 시스템의 개수를 의미한다.)

(4) 제안한 알고리즘의 세부적인 파라미터 설정은 표 2와 같다.

표 2. 제안한 알고리즘의 파라미터 구성.

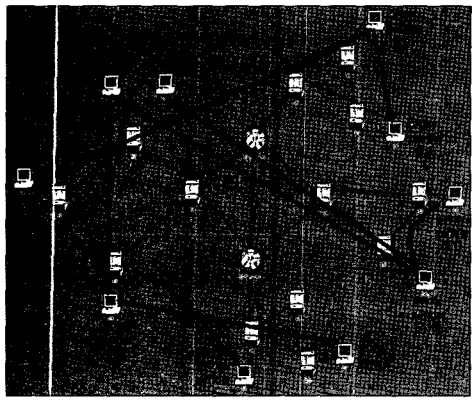
속성	값
종료 조건	세대수 100번 동안 반복
모집단 크기 (K)	200개
p_{flip}	0.07
p_{toggle}	0.02
α	$0 \le \alpha \le 1$

3.1 α값에 따른 제안한 알고리즘의 트리 구성 변화

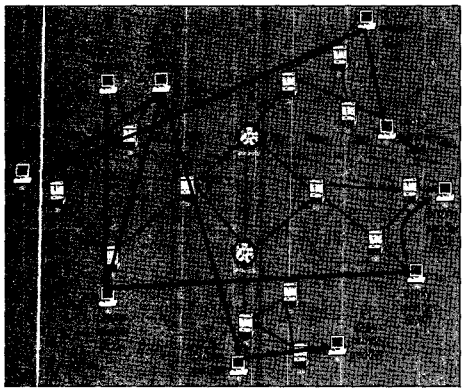
표 3과 그림 3에서는 α값의 변화에 따른 제안한 알고리즘의 트리 구성 변화를 기술하였다. 표 3에서 볼 수 있듯이, α값이 증가함에 따라, 평균 지연은 다소 증가하나, VAR_{tree} 함수의 값은 감소하는 트리를 구성하는 경향을 보이고 있다. 이와 같은 현상은 α값이 증가함에 따라, 식 (5)를 적용한 평가값을 계산함에 있어 VAR_{tree} 함수의 값이 차지하는 비중이 커지기 때문이다. 그림 3을 보면, α값이 1로 수렴함에 따라, 지연시간이 긴 클러스터의 프락시송신자는 송신단으로부터 직접 데이터를 수신하며, 반면에 지연시간이 짧은 클러스터의 프락시송신자는 경로를 우회하여 데이터를 수신하게 됨을 볼 수 있다. 또한 식 (5)에서 VAR_{tree} 함수 부분이 클러스터 내에 포함된 종단 시스템의 수에 가중화되어 있으므로, 종단 시스템이 많은 프락시송신자에 대한 지연시간의 변이가 더 작아짐을 관측할 수 있었다.

표 3. α값에 따른 제안한 알고리즘의 성능.

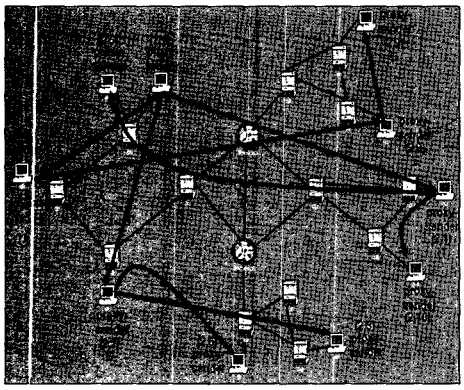
Measure / α	평균 지연	VAR_{tree}
0.0~0.4	466 ms	6.146
0.5	472 ms	5.723
0.6	489 ms	4.943
0.7~0.8	539 ms	3.773
0.9~.1.0	540 ms	3.769



(a)



(b)

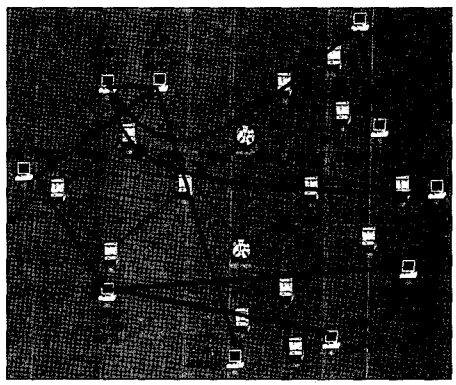


(c)

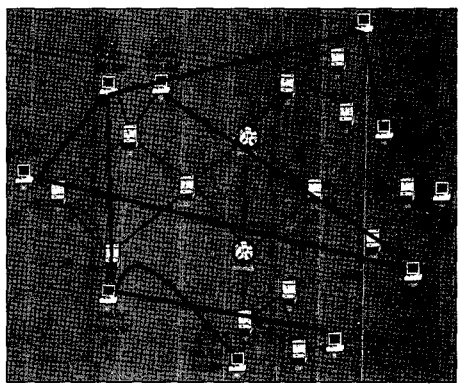
그림 3. α값에 따른 트리 구성의 변화: (a)α값이 0.0~0.4일 때, (b)α값이 0.6일 때, 그리고 (c)α값이 0.9~1.0일 때

3.2 기존 알고리즘들과 제안한 알고리즘의 성능 비교 및 분석

이번 실험은 1절과 동일한 실험환경에서 제안한 알고리즘과 기존 알고리즘들과의 성능을 비교 및 분석하며, 표 4와 그림 4에서 그 결과를 기술하였다. 표 3과 4를 비교해보면, α값이 0일 때, 제안한 알고리즘을 사용하여 구성된 트리가 평균지연에 대해 멀티플 유니캐스트 (Multiple-unicast)를 제외하고, 기존의 알고리즘들 보다 좀 더 좋은 성능을 보이고 있으나, VAR_{tree} 함수의 값은 여전히 크다. (단, 멀티플 유니캐스트는 송신단 측 물리적인 링크상의 많은 대역폭을 요구한다. 구성된 오버레이 멀티캐스트 트리를 통해 전송된 트래픽 측면에서 멀티플 유니캐스트는 물리적인 링크상의 최대 부하가 22.5 Mbps이며, 다른 알고리즘들은 7.5 Mbps ~ 10 Mbps로 측정되었다.) 그러나, 표 3에서 볼 수 있듯이, 제안한 알고리즘에서 α값을 증가시킴으로서, 평



(a)



(b)

그림 4. 제안한 알고리즘과 기존 알고리즘들과의 트리 구성 비교: (a)CPT, 그리고 (b)MDA

표 4. 기존 알고리즘들과의 성능 비교

알고리즘	평균 지연	VAR_{tree}
CPT	515 ms	9.870
MDA	479 ms	8.889
Multiple-unicast	354 ms	5.980

균 지연은 다소 증가하더라도 VAR_{tree} 함수의 값을 감소시키는 트리를 구성함을 수 있음을 알 수 있다. 즉 제안한 알고리즘은 식 (5)의 α값을 조절함으로서, 동기화된 실시간 미디어 데이터 전송을 위한 효과적인 트리를 구성할 수 있다.

Ⅳ. 결 론

본 논문은 멀티캐스트 송신단에서 각 종단 시스템 사이의 전송 지연에 대한 평균과 변이를 동시에 고려하는 트리 구성 알고리즘을 제안한다. 실제로 인터넷 상에서 동기화된 실시간 미디어 멀티캐스트 서비스에 있어 전송 지연에 대한 평균과 변이는 중요한 요소이다. 제안하는 알고리즘은 계산 복잡도를 낮추면서, 근사해를 찾기 위한 방법으로 OGA를 이용하였으며, 식 (5)의 α값을 조절함에 의해 동기화된 실시간 미디어 데이터 전송에 적합한 트리를 구성할 수 있다. 실험 결과에서 제안하는 알고리즘이 기존 알고리즘 보다 동기화된 실시간 미디어 서비스를 위해 효과적인 트리를 구성한다는 것을 보였다.

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] 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, pp. 1444-1455, Oct. 2002.

[4] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," In Proceedings of IEEE INFOCOM, June 2003.

[5] Hwangjun Song, Dong Sup Lee, and Hyung Rai Oh, "Application layer multicast tree constructing algorithm for real-time media delivery," accepted for publication in Computer Communications, Sept. 2005.

[6] D. Helder and S. Jamin, "End-host multicast communication using switch-trees protocols," in Proceedings of Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC), May 2002.

[7] D. C. Montgomery, Design and Analysis of Experiments, 3rd ed. New York: Wiley, 1991.

[8] K. T. Fang and Y.Wang, Number-Theoretic Methods in Statistics. New York: Chapman & Hall, 1994.

[9] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," Computer, Vol. 27, Issue 6, pp. 17-26, June 1994.

[10] Q. Zhang and Y. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," IEEE Tr. on Evolutionary Computation, Vol. 3, No. 1, pp. 53-62, April 1999.

주 현 철 (HyunChul Joo) 준회원

2005년 2월 한양대학교 컴퓨터공 학과 (학사)

2005년 3월~현재 포항공과대학교 컴퓨터공학과 (석사)

<관심분야> IPTV, 영상압축, 오버레이 멀티캐스트

송 황 준 (HwangJun Song) 정회원

1990년 2월 서울대학교 제어계측공학과 (학사)

1992년 2월 서울대학교 제어계측공학과 (석사)

1999년 5월 Univ. of Southern California, EE-Systems (박사)

2000년~2005년 2월 홍익대학교 전자전기공학부

2005년 2월~현재 포항공과대학교 컴퓨터공학과

<관심분야> 영상통신시스템, 오버레이 멀티캐스트, Ad-Hoc