피어의 안정성과 지연을 동시에 고려한 P2P 오버레이 멀티캐스트 트리 구성 알고리즘

준회원 권 오 찬*, 정회원 윤 장 우**, 송 황 준*

A P2P Overlay Multicast Tree Construction Algorithm Considering Peer Stability and Delay

Oh Chan Kwon* *Associate Member*, Changwoo Yoon**, Hwangjun Song* *Regular Members*

요 약

본 논문은 인터넷상에서 안정적인 멀티미디어 서비스를 제공하기 위한 P2P (Peer-to-Peer) 오버레이 멀티캐스트 트리 구성 알고리즘을 제안한다. 제안하는 알고리즘은 멀티캐스트 트리를 구성 할 때 링크의 지연뿐만 아니라 피어의 안정성 (Peer Stability)까지 고려한다. 실제로 피어는 매우 동적이고 불안정한 행동을 하기 때문에 안정적인 서비스를 제공하기 위해서 피어의 안정성은 필수적으로 고려해야한다. 그리고 제안하는 알고리즘에서는 멀티캐스트 트리의 상태에 따라서 링크의 지연과 피어의 안정성 사이의 가중치를 적응적으로 조절한다. 기본적으로 낮은 계산 복잡도로 근사해를 구하기 위해서 유전자 알고리즘 (Genetic Algorithm)을 이용한다. 마지막으로 실험 결과에서는 제안하는 알고리즘의 성능을 보인다.

Key Words : P2P Overlay Network, Multicast Tree, Peer Stability, Link Delay, Quality-of-Service

ABSTRACT

This paper presents a P2P (Peer-to-Peer) overlay multicast tree construction algorithm to support stable multimedia service over the Internet. While constructing a multicast tree, it takes into account not only the link delay, but also peer stability. Since peers actually show dynamic and unstable behavior over P2P-based network, it is essential to consider peer stability. Furthermore, the weighting factor between link delay and peer stability is adaptively controlled according to the characteristics of the multicast tree. Basically, Genetic algorithm is employed to obtain a near optimal solution with low computational complexity. Finally, simulation results are provided to show the performance of the proposed algorithm.

Ⅰ. 서 론

최근 들어 P2P 네트워킹 기술은 크게 각광받고 있다. Gnutella, PPLive, Skype, 그리고 BitTorrent 등 이미 많은 P2P 어플리케이션들[1]이 인터넷 상에서 사용되고 있다. 지금까지는 대부분 파일 공유를 위한 어플리케이션들이 주를 이루었다. 그러나 최근에는 멀티미디어 서비스를 위한 P2P 기반의 어플리케이션들이 많이 제안되고 있다. 최근 기술 보고서[2]에 의하면 동영상 스트리밍 트래픽이 전체 인터넷

※ 이 논문은 KCC/MKE/KEIT의 IT R&D 프로그램의 지원을 받아 작성 되었습니다. [2009-S-018-01, IPTV 융합서비스 및 콘텐츠 공유를 위한 개방형 IPTV 플랫폼 기술 개발]

* 포항공과대학교 컴퓨터공학과 멀티미디어통신 및 네트워킹 연구실(ochanism@postech.ac.kr, hwangjun@postech.ac.kr)

** 한국전자통신연구원 융합서비스 네트워킹연구팀(cwyoon@etri.re.kr)

논문번호 : KICS2010-11-564, 접수일자 : 2010년 11월 29일, 최종논문접수일자 : 2011년 3월 31일

트래픽의 약 10%를 차지하며, 앞으로 급속도로 증가할 것으로 예상되고 있다. 일반적으로 멀티미디어 서비스는 파일 공유 서비스와는 달리 대여폭, 지연, 그리고 지터 (jitter) 등의 측면에서 매우 엄격한 QoS (Quality of Service) 조건을 요구한다. 따라서 P2P 네트워크상에서 끊김 없는 동영상 스트리밍을 제공하기 위해서는 피어의 역할이 더욱 중요해질 것으로 예상된다. 지금까지 제안된 대부분의 P2P 오버레이 멀티캐스트 기술들은 지연만을 고려하여 개발되어왔다[1]. 그러나 피어의 안정성 또한 끊김 없는 동영상 스트리밍을 제공하기 위해 반드시 고려되어야 할 중요한 요소이다. 왜냐하면 피어는 수시로 참여와 탈퇴를 하는 불안정한 존재이기 때문이다. 따라서 피어의 안정성을 고려하여 안정적인 P2P 오버레이 멀티캐스트 구조를 구성하는 연구가 필요하다.

본 논문에서는 피어의 안정성과 지연을 고려한 P2P 오버레이 멀티캐스트 트리 구성 알고리즘을 제안한다. 그리고 피어의 안정성과 지연 사이의 효율적인 trade-off 관계를 조절하기 위하여 그들 사이의 가중치를 조절하는 알고리즘을 제안한다.

Ⅱ. 관련 연구

일반적으로 P2P 오버레이 멀티캐스트는 트리 구조와 메쉬 구조와 같이 크게 두 가지로 분류될 수 있다. 먼저 트리 구조에서 피어는 한명의 부모와 다수의 자식을 갖게 된다. 단, 루트 피어는 부모를 갖지 않다. 피어는 부모로부터 데이터를 받고 그 데이터를 다시 자식들에게 전달한다. 기존 P2P 오버레이 멀티캐스트 트리 기술로써 Yoid, ALMI, 그리고 HMTP 등이 트리 구조에 속한다[1]. 트리 구조에서는 푸쉬 (push) 전달 방식을 사용하기 때문에 지연이 짧은 것이 장점이다. 그러나 피어가 탈퇴했을 때 떠난 피어의 자손들은 전부 연결이 단절되는 단점을 갖고 있다. 반면에 메쉬 구조에서는 피어가 이웃이라고 불리는 여러 피어와 연결되어 있고, 부모/자식 관계가 없다. 그리고 메쉬 구조에서는 풀 (pull) 전달 방식을 사용하여 모든 피어가 서로 간에 데이터를 주고받게 된다. CoolStreaming, Bullet, 그리고 Chainsaw 등이 메쉬 구조의 예이다[1]. 일반적으로 메쉬 구조는 피어의 탈퇴나 실패에 강인하지만, 지연이 길고 컨트롤 오버헤드가 크다는 단점이 있다. 지금까지 대부분의 P2P 오버레이 멀티캐스트 트리 구성 알고리즘은 지연만을 최소화하기 위해 노력하였다[1]. 대부분의 알고리즘들은 피어의 행동을 고려하지 않았다. Y. Tian[3]과 F. Wang[4]은 피어의 행동을 확률적으로 모델링하려는 시도를 하였다. 그들은 피어가 세션에 참여한 후에 세션에 머무르는 시간이 파레토 분포를 따른다는 것을 보였다. 그러나 세션 초기에 이런 확률적 모델만으로 피어의 행동을 예측하는 것은 힘들다. 그 이유는 위의 모델에서는 현재 세션에 머무른 시간에 대한 정보를 요구하기 때문이다. 따라서 정확한 피어의 행동을 예측하기 위해서는 세션이 시작하기 이전의 피어에 대한 추가적인 과거 정보가 필요하다. 평판 (reputation)[5]은 피어의 과거 정보를 대표 할 수 있는 하나의 해결책으로 볼 수 있다. P2P 오버레이 멀티캐스트 트리를 구성하는 대에 있어서 중요한 피어의 행동은 피어가 탈퇴 할 때 그것을 다른 피어들에게 알리는지에 대한 여부와 피어가 얼마나 오랫동안 세션에 참여했었는지에 대한 것이다. 따라서 본 논문에서는 이러한 정보를 바탕으로 피어의 과거 평판을 계산한다. 피어의 과거 행동은 미래의 행동과 매우 연관성이 높다는 가정[6] 하에 과거 평판을 기반으로 피어의 행동을 예측 할 수 있다.

Ⅲ. 본 론

제안하는 알고리즘의 목적은 인터넷상에서 안정적인 멀티미디어 서비스를 제공하기 위한 P2P 오버레이 멀티캐스트 트리를 구성하는 것이다. 피어의 안정성과 피어들 사이의 지연은 전체 P2P 네트워크 성능 측면에서 매우 중요한 역할을 한다. 앞서 언급했듯이 피어들은 세션 도중에 빈번히 참여와 탈퇴를 하기 때문에 피어의 안정성은 중요한 요소이다. 또한 피어들 사이의 지연은 멀티미디어 서비스의 지연 제약 조건을 만족시키기 위해서 중요한 요소이다.

본 논문에서 고려하고 있는 P2P 네트워크 구조는 그림 1과 같다. 부트스트랩 서버는 피어들이 멀티캐스트 세션에 참여할 수 있도록 도와주며, 피어들의 정보를 관리한다. 피어들은 루트 피어, 클러스터 리더 피어, 그리고 클러스터 멤버 피어와 같이 세 가지 그룹으로 분류된다. 루트 피어는 멀티미디어 콘텐츠를 저장하고 있으며, 그것을 다른 피어들에게 스트리밍을 제공하는 역할을 한다. 제안하는 시스템에서는 단 한명의 루트 피어만을 고려하고 있다. 그리고 각 클러스터에서는 지연과 피어의 안정성을 고려하여 한 명의 피어를 클러스터 리더로

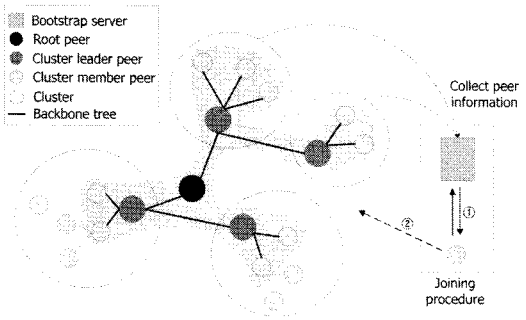


그림 1. 문제 접근을 위한 P2P 네트워크 구조

서 선정한다. 클러스터 리더 피어는 클러스터를 대표하는 피어로서 루트 피어 혹은 다른 클러스터 리더 피어로부터 스트리밍 서비스를 받게 된다. 반면에 클러스터 멤버 피어는 자신이 속한 클러스터의 클러스터 리더 피어 혹은 다른 클러스터 멤버 피어로부터 스트리밍 서비스를 받게 된다. 여기에서 피어들의 위치는 피어와 랜드마크(landmark)[7] 사이의 지연을 측정함으로써 이미 예측 되어 그것을 기반으로 클러스터링 되어 있다고 가정한다. 또한 피어들은 주기적으로 Hello 패킷을 주고받는다. 일정 시간동안 특정 피어로부터 Hello 패킷을 받지 못하는 경우 해당 피어가 떠난 것으로 간주한다. 본 논문에서는 멀티캐스트 트리를 어떻게 구성하고 유지보수 하는지에 초점을 맞춘다. 먼저 N_h-홉 백본 트리(N_h-hop backbone tree)를 정의한다. 여기에서 N_h는 0 이상의 정수이다. N_h-홉 백본 트리는 서브트리(sub-tree)로써 루트 피어, 모든 클러스터 리더 피어, 그리고 각 클러스터 리더 피어로부터 N_h-홉 이내에 위치한 클러스터 멤버 피어들로 구성된다. 그림 1에는 1-홉 백본 트리의 예가 있다. N_h가 0일 경우에 백본 트리는 오직 루트 피어와 클러스터 리더 피어만으로 구성된다. N_h-홉 백본 트리는 뒷 절에서 적응적 ω 제어 알고리즘에서 사용된다.

3.1 문제 정의

문제 정의하기에 앞서 피어의 안정성 지수를 정의한다. 피어의 안정성 지수는 과거 피어 평판과 현재 세션에서의 예측된 피어 행동에 의해서 정의된다. 과거 피어 평판은 피어가 이전 세션에서 자신이 떠남을 알린 횟수와 머무른 시간 정보를 기반으로 한다. 그리고 현재 세션에서의 예측된 피어 행동은 확률적으로 현재 세션에서 머무를 시간을 예측하여 정의한다.

과거 피어 평판 정의: 과거 피어 평판 R_{hist}는 다음과 같이 정의된다.

$$R_{hist}=\gamma_{st}\cdot\frac{\sum_{i=1}^{n_s}T_{s_i}^{stay}}{\sum_{i=1}^{n_s}T_{s_i}}+(1-\gamma_{st})\cdot\frac{n_{gr}}{n_s}, \quad (1)$$

위 수식에서 γ_{st}는 상수이며$(0\le\gamma_{st}\le 1)$, n_s는 피어가 과거에 참여한 세션의 횟수이고, $T_{s_i}^{stay}$는 피어가 i번째 세션에 머무른 시간을 나타내고, T_{s_i}는 i번째 세션의 총 시간을 나타내고, n_{gr}은 피어가 자신이 떠남을 알리고 떠난 횟수이다. 이러한 과거 정보 $T_{s_i}^{stay}$, n_{gr}는 부트스트랩 서버에서 관리되며, 피어가 떠날 때 그 피어가 세션에 머무른 시간은 자신 혹은 그 피어의 자식 피어로부터 부트스트랩 서버에게 보고된다.

현재 세션에서 예측된 피어의 행동 정의: 현재 세션의 시작 시간은 0이고, 피어가 앞으로 머무를 시간은 파레토 분포(Pareto distribution)를 따른다고 가정한다[3]. 피어가 시간 t_{join}에 참여했을 때, 시간 t에 그 피어의 예측된 피어의 행동 $B_{curr}(t,t_{join})$는 다음과 같이 정의된다.

$$B_{curr}(t,t_{join})=\begin{cases}\dfrac{E_{rt}(t,t_{join})}{T_s-t} & \text{if } E_{rt}(t,t_{join})\le T_s-t,\\ 1 & otherwise,\end{cases} \quad (2)$$

$$E_{rt}(t,t_{join})=\frac{\beta_B+(t-t_{join})}{\alpha_B}, \quad (3)$$

위 수식에서 T_s는 현재 세션의 총 시간이고, (T_s-t)는 현재 세션의 남은 시간을 나타내고, α_B와 β_B는 파레토 분포의 파라미터이다. $E_{rt}(t,t_{join})$는 현재 세션에서 피어가 앞으로 얼마 동안 머무를지 확률적으로 예측한 값이다[3]. $B_{curr}(t,t_{join})$는 1에 가까울수록 피어가 세션에 더 오래 머물 것임을 의미한다. $E_{rt}(t,t_{join})$이 (T_s-t) 보다 클 경우에 $B_{curr}(t,t_{join})$는 1로 결정된다.

피어의 안정성 지수 정의: 피어의 안정성 지수 $S(t,t_{join})$는 다음과 같이 정의된다.

$$S(t,t_{join})=\rho \cdot R_{hist}+(1-\rho)\cdot B_{curr}(t,t_{join}), \quad (4)$$

$$\rho=\left(1+\left(\frac{n_s}{N_{trust}}\right)^{-k}\right)^{-1}, \quad (5)$$

위 식에서 ρ는 과거 평판의 신뢰성이고, k와 N_{trust}는 1보다 큰 정수이다. 과거 정보가 더 누적될 수 록 과거 평판의 신뢰성 높아진다. 다시 말해서 과거 정보의 누적된 양이 충분하지 않을 때 $(n_s < N_{trust})$, $B_{curr}(t,t_{join})$는 피어의 안정성 지수를 계산하는데 있어서 더 중요한 요소로 고려된다.

멀티캐스트 트리 상에서 피어의 안정성 지수는 자신의 조상(ancestor) 피어들과 연관성을 가진다. 예를 들어 어떠한 피어가 세션을 탈퇴했을 때 그 피어의 손자(descendant) 피어들은 트리와 연결이 끊어짐으로써 서비스를 받지 못하게 된다. 따라서 멀티캐스트 트리 상에서 상대적인 피어의 안정성 지수는 반드시 고려되어야 한다.

멀티캐스트 트리 상에서의 상대적인 피어의 안정성 지수 정의: 멀티캐스트 트리 상에서 상대적인 피어의 안정성 지수 $RS_p(t)$는 다음과 같이 정의된다.

$$RS_p(t)=\prod_{n_i\in \vec{P}_p^{anc}} S_{n_i}\left(t,t_{join_{n_i}}\right), \quad (6)$$

위 식에서 $\vec{P}_p^{anc}$는 피어 p의 조상 피어 벡터를 나타내고, $S_{n_i}\left(t,t_{join_{n_i}}\right)$는 $\vec{P}_p^{anc}$의 i번째 피어의 안정성 지수이고, $t_{join_{n_i}}$는 $\vec{P}_p^{anc}$의 i번째 피어의 참여 시각을 나타낸다. 상대적인 피어의 안정성 지수의 물리적인 의미는 멀티캐스트 트리 상에서 루트로부터 해당 피어까지의 경로의 안정성이다. 이 상대적인 피어의 안정성 지수는 멀티캐스트 트리 상에서 주기적으로 갱신된다. 어떤 피어가 자신의 자식들에게 자신의 상대적인 피어를 전송하면, 그 자식 피어들은 자신의 상대적인 안정성을 갱신한다. 모든 피어들은 이러한 과정을 통해 분산적으로 상대적인 안정성을 갱신한다.

문제정의: 다음 효용함수 $U(t)$를 최대화 하는 P2P 오버레이 멀티캐스트 트리를 구성한다.

$$U(t)=\omega\cdot\frac{\sum_{i=1}^{N_P} RS_{p_i}(t)}{N_P}+(1-\omega)\cdot\frac{1}{\max_{1\le i\le N_P}(d_i)}, \quad (7)$$

$$\text{subject to } g_i \le G_i^{\max} \text{ for } 1\le i\le N_P, \quad (8)$$

위 식에서 ω는 상수이며$(0\le\omega\le 1)$, N_P는 세션에 참여하는 모든 피어의 수이고, $RS_{p_i}(t)$는 i번째 피어의 상대적인 안정성이고, d_i는 멀티캐스트 트리 상에서 루트 피어로부터 i번째 피어까지의 지연이며, g_i는 i번째 피어의 차수, 그리고 $G_i^{\max}$는 i번째 피어의 최대 차수를 나타낸다. 여기서 차수란 피어에게 연결된 링크의 수를 의미한다. 따라서 효용함수 $U(t)$는 상대적인 피어의 안정성 지수가 크고, 최대 지연이 작을수록 더 큰 값을 가지게 된다.

3.2 제안하는 트리 구성 알고리즘

위 문제의 최적해를 구하는 것은 NP-완전하다. 따라서 제안하는 알고리즘에서는 계산 복잡도를 줄이기 위해 유전자 알고리즘[8]을 사용하여 근사해를 구한다. 제안하는 멀티캐스트 트리 구성 알고리즘은 세 부분으로 구성된다. 첫 번째는 세션이 시작하기 전에 초기 트리 구성하는 부분이고, 두 번째는 피어가 참여 혹은 탈퇴함에 따라 부모 피어를 선택하는 부분이다. 그리고 마지막으로 적응적 ω 제어 메커니즘이 있다. 제안하는 알고리즘은 초기에 트리를 구성하기 위해서 단 한번만 중앙 집중적으로 동작하고, 세션이 시작된 이후에는 분산적으로 동작한다.

3.2.1 초기 트리 구성 알고리즘

세션이 시작하기 이전에 부트스트랩 서버는 유전자 알고리즘을 사용하여 참여 피어들 사이의 초기 트리를 구성하고, 각 피어들에게 자신의 부모를 알려준다. 유전자 알고리즘은 근사해에 수렴하는 속도가 매우 빠르기 때문에 부트스트랩 서버에 큰 부담이 되지 않는다[8]. 그러나 유전자 알고리즘을 사용하더라도 P2P 네트워크의 규모가 커지게 되면 계산 복잡도는 증가 할 수밖에 없다. 따라서 two-tier hierarchical 접근 방법을 사용한다. 먼저 각 클러스터마다 안정적인 피어들을 클러스터 리더 후보로 선정한다. 그리고 클러스터 리더들 사이의 트리를 구성한 다음 각 클러스터 내의 트리를 구성한다. 자세한 트리 구성 과정은 다음과 같다.

가정: 모든 피어는 서로 연결되어 있고(full-mesh), 그래프 상의 n개의 링크들은 각각 1부터 n까지 넘버링 되어있다고 가정한다. 이때 멀티캐스트 트리는 다음과 같이 벡터 형태로 표현 될 수 있다.

Multicast tree vector $T=(e_1,e_2,...,e_i,...,e_n)$,

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

Step 0: 각 클러스터마다 상대적으로 높은 과거 평판을 가진 N_C명의 피어를 클러스터 리더 후보로 선정한다. N_{CS} 개의 후보 집합 $(cs_1,cs_2,...,cs_{N_{CS}})$을 만든다. 여기에서 cs_i는 각 클러스터로부터 한명의 클러스터 리더 후보를 선택하여 만든 i번째 후보 집합이다.

Step 1: 순차적으로 한 개의 후보 집합 cs_i를 선택한다.

Step 2: 후보 집합 cs_i에 속한 피어들로 구성된 N_{TREE}개의 멀티캐스트 트리 $(T_1,T_2,...,T_{N_{TREE}})$를 랜덤하게 생성한다. 생성된 멀티캐스트 트리는 초기 모집단으로 사용된다.

Step 3: 생성된 멀티캐스트 트리는 완벽한 트리 형태가 아닐 수 있다. 따라서 check-and-repair 과정을 통해서 트리로 만든다. 세부적인 과정은 그림 2와 같다.

Step 4: 멀티캐스트 트리를 수식 (7)에 정의 된 효용함수에 따라서 정렬한다. 먼저 높은 효용함수 값을 가지는 $N_{TREE}/2$개의 멀티캐스트 트리를 선택한다. 그리고 선택된 멀티캐스트 트리에 cross-over와 mutation 연산[8]을 적용하여 $N_{TREE}/2$개의 새로운 멀티캐스트 트리를 생성한다. 결과적으로 선택된 N_{TREE}개의 멀티캐스트 트리가 다음 세대 모집단으로 사용된다.

Step 5: Steps 3~4를 N_{GEN}번 반복한다. 그리고 Step 1에서 선택된 cs_i에 대해서 적합도가 가장 큰 멀티 캐스트 트리를 결정한다.

Step 6: Steps 1~5를 N_{CS}번 반복한다. 결과적으로 후보 집합 $(cs_1,cs_2,...,cs_{N_{CS}})$에 대해서 멀티캐스트 트리가 결정된다. 이 트리 중에서 가장 적합도가 높은 트리를 최종 트리로 선택한다.

위의 과정을 통해 클러스터 리더 사이의 멀티캐스트 트리를 구성하고, 같은 방법으로 클러스터 내의 멀티캐스트 트리를 구성한다.

3.2.2 피어의 참여/탈퇴에 따른 부모 선택 알고리즘

멀티캐스트 세션에 새로운 피어가 참여 했을 경우 그 피어는 PCL (parent candidate list)를 부트스트랩 서버로부터 받게 된다. 부트스트랩 서버는 참여한 피어들의 정보를 관리하며, 높은 안정성을 가지며 차수가 남아있는 피어들을 포함하여 PCL을 만든다. 새로 참여한 피어가 PCL에 속한 피어 p_{par}을 부모로 선택했을 경우 PMI (parent matching index)는 다음과 같이 정의 된다.

$$PMI_{p_{par},new}\left(t,t_{join_{p_{par}}}\right)=\omega\cdot RS_{p_{par}}(t)\cdot S_{p_{par}}\left(t,t_{join_{p_{par}}}\right)+(1-\omega)\cdot\frac{1}{d_{p_{par}}+d_{p_{par},new}},\quad(9)$$

위 식에서 $RS_{p_{par}}(t)$는 루트로부터 선택된 부모사이 경로의 상대적인 안정성이고, $S_{p_{par}}\left(t,t_{join_{p_{par}}}\right)$는 선택된 부모 피어의 안정성 지수이고, $d_{p_{par}}$는 클러스터 리더로부터 선택된 부모 피어까지의 지연이며, $d_{p_{par},new}$는 선택된 부모 피어와 새로운 피어사이의 지연이다. 피어는 PCL 중에서 가장 큰 PMI를 가지는 피어를 부모 피어로써 선택하게 된다.

세션 도중 피어가 탈퇴 했을 때, 그 피어의 자손들은 멀티캐스트 트리로부터 단절되게 된다. 이 경

Initialization:
Peer_set={All peers in multicast tree}
Tree_set={Root peer}
Queue={Root peer}

Check and repair:
WHILE({Peer_set - Tree_set} != ∅)
  WHILE (Queue != Empty)
    1. Dequeue a peer (A) from Queue.
    2. Randomly select peers (B) connected with peer (A) as many as possible under the following conditions.
      a. Peers (B) must not be included in Tree_set.
      b. The number of peers (B) are less than maximum degree of peer (A).
    3. Enqueue peers (B) to Queue and add to peers (B) to Tree_set.
  ENDWHILE
  IF ({Peer_set - Tree_set} != ∅) THEN
    1. Randomly select a peer (C) from {Peer_set - Tree_set}.
    2. Connect a peer (C) to the peer with the largest PMI (Eq. 9) among Tree_set.
    3. Enqueue peer (C) to Queue and add to peer (C) to Tree_set.
  ENDIF
ENDWHILE

그림 2. check-and-repair 수도 코드 (pseudo code)

우에 끊김 없는 멀티미디어 서비스를 위해서는 가능한 한 빨리 새로운 부모 피어를 찾을 필요가 있다. 그러나 자손 모두가 새로운 부모 피어를 찾게 된다면 네트워크에 혼잡을 발생 시킬 수 있다. 따라서 자손 모두가 새로운 부모 피어를 찾는 대신 단절된 자손 서브트리의 루트 피어들만 다시 새로운 부모 피어를 찾는 과정을 수행하게 된다.

3.2.3 적응적 ω 제어 알고리즘

제안하는 알고리즘에서 가중치 ω는 안정성과 지연 사이에 trade-off 관계를 조절하는 매우 중요한 역할을 한다. 일반적으로 시간이 지남에 따라 불안정한 피어들은 탈퇴하게 되고, 상대적으로 안정적인 피어들만 남기 때문에 전체 멀티캐스트 트리의 안정성은 향상되게 된다. 따라서 멀티캐스트 트리의 상대적인 안정성의 평균값이 특정 수준에 도달하게 되었을 때, 지연이 더욱 중요한 요소로 고려되어야만 한다. 그러므로 효율적인 멀티캐스트 트리를 구성하기 위해서는 가중치 ω를 동적으로 트리 상태에 따라서 조절할 필요가 있다. 그림 1에서 언급한 바와 같이 트리의 안정성을 판단하기 위해서 N_h-홉 백본 트리의 안정성을 사용한다. N_h-홉 백본 트리의 안정성의 평균값이 클 경우 ω는 작게 설정되고, 반대의 경우에는 ω는 크게 설정된다. N_h-홉 백본 트리의 안정성을 기반으로 ω는 다음과 같이 제어된다.

$$\omega(\vec{P}_b)=\begin{cases}1-\dfrac{BS(\vec{P}_b)}{S_{target}} & \text{if } BS(\vec{P}_b)\le S_{target},\\ 0 & otherwise,\end{cases} \quad (10)$$

$$BS(\vec{P}_b)=\frac{\sum_{p_i\in\vec{P}_b} RS_{p_i}(t)}{|\vec{P}_b|}, \quad (11)$$

위 식에서 $\vec{P}_b$는 N_h-홉 백본 트리에 속하는 피어의 벡터이고, S_{target}은 멀티캐스트 트리 안정성의 목표 값이고, $BS(\vec{P}_b)$는 N_h-홉 백본 트리의 평균 안정성이다. N_h-홉 백본 트리에 속하는 피어들은 자신의 상대적인 안정성을 부모 피어에게 주기적으로 전송하고, 부모 피어들은 그 값에 자신의 상대적인 안정성을 더함으로써 같은 과정을 반복하게 된다. 따라서 루트 피어는 N_h-홉 백본 트리의 모든 피어의 상대적인 안정성의 합을 구할 수 있고, 이 값은 부트스트랩 서버에게 전송된다. 결과적으로 부트스트랩 서버는 ω를 분산적으로 갱신할 수 있다.

Ⅳ. 실험결과

본 실험은 NS-2[9] 시뮬레이터를 사용하여 진행하였다. 전체 피어의 수는 6000명이고, 600명의 피어가 하나의 클러스터로 구성되며 총 10개의 클러스터가 있다. 피어의 차수는 1~5 사이의 값으로 균일 분포를 따른다. 전체 피어 중 약 30%는 세션이 시작하기 전에 참여하며, 나머지 피어들은 세션이 시작한 후에 푸아송 프로세스에 따라 참여하게 된다. 피어의 참여 시간은 파레토 분포를 따르며, 파레토 분포의 파라미터 α_B, β_B는 각각 2.1과 430으로 설정하였다. 수식 (1)에서 γ_{st}는 0.5로 설정하였고, N_{trust}와 k는 모두 3으로 설정하였다. 상대적인 피어의 안정성 지수를 갱신하기 위한 주기는 5초로 설정 되었다. 갱신 패킷의 크기는 8 bytes의 매우 작은 크기로써 큰 오버헤드가 되지 않는다. 실험에 참여하는 피어의 성향과 분포는 표 1과 같다. 표 1에서 Graceful Leave Probability는 피어가 자신이 떠남을 부모와 자식 피어에게 알리고 떠나는 확률이다. 피어의 과거 평판은 다른 시나리오로 5번의 실험을 반복하여 얻었다. 모든 실험 결과는 각기 다른 시나리오의 실험 10번을 반복하여 얻은 평균값이다.

표 1. 피어의 성향과 분포

Level	Duration	Graceful Leave Probability	Proportion
Best	100%	1.0	12%
Good	50~100%	0.9	8%
Normal	25~50%	0.7	10%
Bad	10~25%	0.4	30%
Worst	0~10%	0.3	40%

4.1 ω 값에 따른 성능 검증

이 절에서는 ω 값에 따른 멀티캐스트 트리의 변화와 성능을 실험하였다. 수식 (7)에서 ω가 증가 할수록 안정성의 가중치가 증가하며, 반대의 경우는 지연에 대한 가중치가 증가한다.

그림 3 (a), (b) (c)에서는 세션이 진행됨에 따라 스트리밍 서비스를 받는 피어의 수의 변화를 보인다. 그림 3 (a), (b)에서는 대부분의 피어들이 성공적으로 서비스를 받고 있는 것을 확인 할 수 있다. 하지만 그림 3 (c)에서 보이듯이 $\omega=0$ 일 경우에

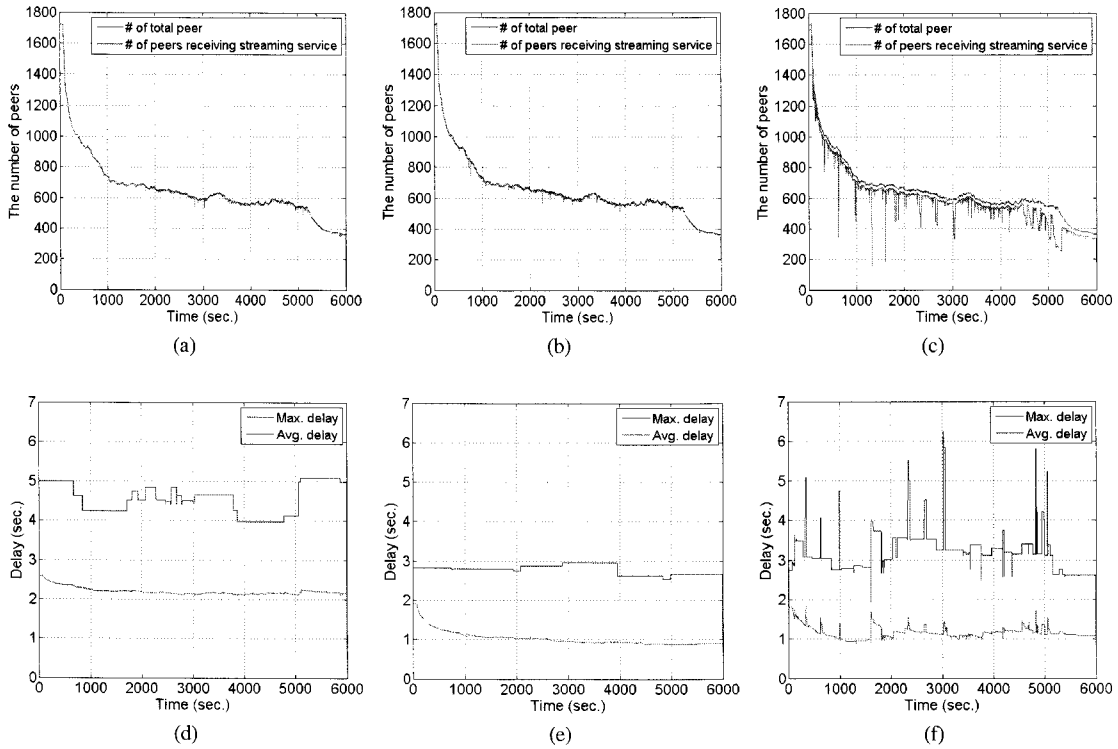


그림 3. ω 값에 따른 스트리밍 서비스를 받는 피어의 수의 변화: (a) $\omega=1$, (b) $\omega=0.5$, and (c) $\omega=0$ 그리고 ω 값에 따른 지연의 변화: (d) $\omega=1$, (e) $\omega=0.5$, and (f) $\omega=0$.

상당히 많은 수의 피어들이 서비스 받지 못하는 것을 볼 수 있다. 이것은 멀티캐스트 트리의 상단에 위치한 많은 수의 불안정한 피어들이 짧은 시간 안에 탈퇴함으로써 발생되는 결과이다.

그림 3 (d), (e), (f)에서는 세션이 진행됨에 따라 지연의 변화를 보여주고 있다. 그림 3 (d), (e)에서 볼 수 있듯이 ω가 1에서 0.5로 감소함에 따라 평균 지연과 최대 지연이 감소하는 것을 알 수 있다. 그러나 그림 3 (f)에서는 ω를 0으로 설정하여 지연만 고려했음에도 불구하고 오히려 지연이 증가한 것을 확인 할 수 있다. 이것은 지연만 고려할 경우 멀티캐스트 트리는 매우 불안정하게 되어 잦은 피어들의 탈퇴로 인하여 많은 피어들이 새로 참여하는 과정 중에서 지연이 발생하기 때문이다. 멀티캐스트 트리가 극도로 불안정 할 경우 부모를 찾는 시간이 증가하기 때문이다. 이 결과는 P2P 오버레이 네트워크상에서 지연을 줄이기 위해 피어의 안정성도 중요한 요소임을 나타내고 있다.

4.2 적응적 ω 제어 알고리즘의 성능 검증

이 절에서는 적응적 ω 제어 알고리즘의 성능을 평가한다. ω 값을 0.2, 0.3, 0.5로 고정했을 때와 비교 실험을 진행하였다. N_h와 S_{target}은 각각 5와 0.5로 설정하였다. 실험 결과는 그림 4에 정리되어있다. 실험 결과에서 알 수 있듯이 제안하는 알고리즘은 서비스 받지 못하는 피어의 수와 지연 측면에서 모두 가장 좋은 성능을 보여준다. 그 이유는 고정 ω의 경우에는 세션 초기에 불안정한 피어들이 매우 많이 존재하더라도, ω 값은 변경되지 않기 때문이다. 그러나 적응적 ω의 경우에는 멀티캐스트 트리를 안정적으로 만들기 위해서 ω를 증가시킨다. 같은 방식으로 세션 초기에 안정적인 피어들이 매우 많은 경우에는 지연을 줄이기 위해서 ω를 감소시킨다.

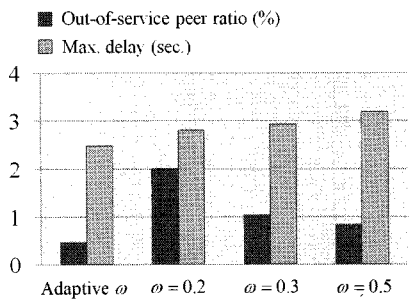


그림 4. 적응적 ω와 고정 ω의 성능 비교

4.3 기존 알고리즘과의 성능 비교

이 절에서는 제안하는 알고리즘과 기존 알고리즘[3]과의 성능 비교를 한다. 비교하는 기존 알고리즘은 Random algorithm, Oracle, Oldest, Minimum depth, 그리고 Behavior Probability Prediction 이다. 이 알고리즘들의 자세한 설명은 다음에 요약 되어 있다.

Random: 초기 트리는 랜덤하게 구성되고, 부모 역시 랜덤하게 선택되는 알고리즘이다.

Oracle: 이 알고리즘은 모든 피어들의 참여/탈퇴 시간을 알고 있다고 가정한다. 실제적으로는 구현 불가능한 알고리즘이다. 초기 트리는 가장 오래 남아있는 피어를 트리 상단에 위치하는 방식으로 구성된다. 부모 역시 가장 오랫동안 연결을 유지 할 수 있는 피어를 선택한다.

Oldest: 이 알고리즘에서는 가장 오랫동안 머문 피어를 부모로 선택한다. 초기 트리는 랜덤하게 구성된다. 그 이유는 세션 초기에는 피어가 머문 시간 정보가 없기 때문이다.

Minimum Depth: 이 알고리즘은 트리의 깊이(depth)를 최소화하여 초기 트리를 구성한다. 부모 역시 깊이가 가장 작은 피어를 선택한다.

Behavior Probability Prediction (BPP): 이 알고리즘에서는 수식 (2)의 예측된 피어의 행동만을 이용하여 안정성을 정의한다. 그 외에는 제안하는 알고리즘과 동일하게 동작한다. 그러나 세션 초기에 피어의 과거 평판 정보가 없기 때문에 초기 트리는 랜덤하게 구성된다.

실험 결과는 그림 5에 정리되어 있다. 제안하는 알고리즘은 서비스 받지 못하는 피어의 수 측면에서 Oracle을 제외한 나머지 알고리즘보다 좋은 성능을 보였다. Oracle은 실제로 구현 불가능한 알고리즘으로써 안정성 측면에서 최적의 결과를 보여준다. 제안하는 알고리즘이 Oracle에 가장 근사한 것을 알 수 있다. 지연 측면에서는 제안하는 알고리즘이 가장 좋은 성능을 보였다. Oracle, Oldest, 그리고 BPP는 오직 안정성만 고려하기 때문에 매우 큰 지연을 보인다. Minimum depth는 트리의 깊이를 최소화 하였지만 실제 링크 지연을 고려하지 않았기 때문에 지연이 실제적으로 매우 큰 것을 확인할 수 있다.

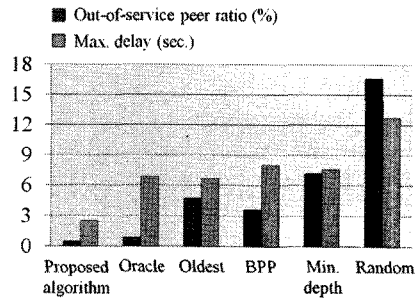


그림 5. 기존 알고리즘과의 성능 비교

V. 결 론

본 논문에서는 피어의 안정성과 지연을 동시에 고려하여 효율적인 P2P 오버레이 멀티캐스트 트리 구성 알고리즘을 제안하였다. 피어의 안정성 지수를 정의하기 위하여 피어의 과거 평판과 확률적으로 예측한 피어가 세션에 머무른 시간을 사용하였다. 그리고 피어의 안정성과 지연 사이의 효율적인 trade-off 관계를 유지하기 위하여 가중치 제어 알고리즘을 제안하였다. 제안하는 알고리즘은 유전자 알고리즘을 이용하여 계산 복잡도가 낮으며, 초기 트리 구성을 제외하고는 분산적으로 동작한다. 그리고 실험결과에서는 기존 알고리즘에 비하여 안정성뿐만 아니라 지연 측면에서도 우수한 성능을 보였다.

참 고 문 헌

[1] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," Journal of Peer-to-Peer Networking and Applications, Vol.1, No.1, pp.18-28, March 2008.

[2] H. Schulze and K. Mochalski, "Internet Study 2008/2009," http://www.ipoque.com/resources/internet-studies/, Tech. report, 2009.

[3] Y. Tian, D. Wu, G. Sun and KW. Ng, "Improving stability for peer-to-peer multicast overlays by active measurements," Journal of Systems Architecture, Vol.54, No.1-2, pp.305-323, January 2008.

[4] F. Wang, J. Liu and Y. Xiong, "Stable Peers: Existence, Importance, and Application in Peer-to-Peer Live Video Streaming," IEEE Conference on Computer Communications (INFOCOM), April 2008.

[5] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," IEEE Transactions

Knowledge and Data Engineering, Vol.16, No.7, pp. 843-857, 2004.

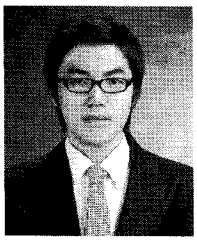
[6] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, October 2006.

[7] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for scalable distributed location," 2nd International Workshop on Peer-to-Peer Systems, February 2003.

[8] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," IEEE Computer, Vol.27, No.6, pp.17-26, June 1994.

[9] S. McCanne and S. Floyd, The Network Simulator, ns-2 http://www.isi.edu/nsnam/ns.

권 오 찬 (Oh Chan Kwon) 준회원



2008년 2월 홍익대학교 컴퓨터공학과 (학사)

2008년 3월~현재 포항공과대학교 컴퓨터공학과 (통합과정)

<관심분야> P2P 네트워크, 오버레이 멀티캐스트

윤 장 우 (Changwoo Yoon) 정회원

1990년 2월 서강대학교 학사

1992년 8월 포항공과대학교 석사

2005년 8월 Univ. of Florida, 컴퓨터공학 박사

2008년 8월~현재 UST(과학기술연합대학원대학교) 겸임교수 (부교수)

1992년 8월~현재 한국전자통신연구원 책임연구원

<관심분야> IPTV, Smart TV, 통방융합, 서비스 생성/제어, 미래인터넷, 정보검색

송 황 준 (HwangJun Song) 정회원

1990년 2월 서울대학교 제어계측 공학과 (학사)

1992년 2월 서울대학교 제어계측 공학과 (석사)

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

2000년 9월~2005년 2월 홍익대학교 (조교수)

2005년 2월~현재 포항공과대학교 (부교수)

<관심분야> 멀티미디어 네트워킹, 영상압축, 통방융합기술