韓國컴퓨터情報學會 論文誌
第15卷 第3號, 2010. 3.

2010-15-3-3-4

푸시-메시 구조 기반의 효율적인 피어투피어 스트리밍 기법

김 진 성*, 김 동 일**, 김 은 삼***, 배 성 일****

An Efficient Peer-to-Peer Streaming Scheme Based on a Push-Mesh Structure

Jin-Sung Kim*, Dong-il Kim**, Eunsam Kim***, Sung-il Pae****

요 약

피어투피어 방식의 스트리밍 기법은 크게 트리-푸시 구조와 메시-풀 구조로 연구가 진행되었다. 하지만 트리-푸시 구조는 트리를 재구성하는 지연 시간이 상당히 길어진다는 단점이 있고, 메시-풀 구조는 서버와의 재생 시간차가 크며, 초기 재생 시작이 지연되는 단점을 가진다. 본 논문에서는 트리-푸시 구조와 메시-풀 구조의 장점을 동시에 사용하는 새로운 방식의 푸시-메시 구조의 피어투피어 스트리밍 기법을 제안한다. 이 기법에서는 기본적으로 메시-풀 구조를 기반으로 높은 네트워크 업로드 성능을 가지는 피어를 최대한 활용한다. 또한, 소스 서버와 수퍼 피어 및 수퍼 피어와 특정 수의 일반 피어들 사이에 푸시 방식의 데이터 전송을 지원한다. 마지막으로 NS-2시뮬레이터를 이용한 실험을 통해 초기 재생 지연 시간이 메시-풀 구조보다 감소하였고, 소스 서버와의 시간차는 트리와 비슷하며, 재생 연속성은 가장 우수하다는 것을 보였다.

Abstract

The research on peer-to-peer streaming schemes has largely focused on tree-push and mesh-pull structures. However, the tree-push structure has a defect that the tree restructuring time is long, and the mesh-pull structure has long startup delay and lag time from source servers. In this paper, we propose a new peer-to-peer live streaming scheme based on a push-mesh structure that takes advantages of tree-push and mesh-pull structure simultaneously. This structure basically provides the mesh-pull mechanism for data transmission and utilizes peers with high network upload capacity. It also supports the push mechanism along with paths from a source server, super peers, and selected general peers. By NS-2 simulation experiments, we finally show that our proposed scheme can achieve shorter startup delay than the mesh-pull structure, similar lag time to tree-push structure and best playback continuity among the three schemes.

▸ Keyword : 피어투피어 스트리밍(Peer-to-Peer Streaming), 하이브리드 피어투피어 구조(Hybrid Peer-to-Peer Structure), 푸시-메시 구조(Push-Mesh Structure)

• 제1저자 : 김진성 교신저자 : 김은삼
• 투고일 : 2010. 02. 29, 심사일 : 2010. 03. 04, 게재확정일 : 2010. 03. 18.
*홍익대학교 컴퓨터공학과 박사과정 **홍익대학교 컴퓨터공학과 석사과정 ***홍익대학교 컴퓨터공학과 조교수
****홍익대학교 컴퓨터공학과 전임강사
※ 이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2009-0071912, 2009-0077288).

Ⅰ. 서 론

현재 컴퓨터 네트워크의 발전으로 초고속통신망이 일반화되고 방송 컨텐츠가 디지털화됨에 따라 기존의 방송서비스와 통신서비스를 융합하는 새로운 차세대 서비스들이 등장하고 있다. 기존에는 고품질의 멀티미디어 데이터를 실시간으로 전송하기 위해 CDN(Content Distribution Network) 등과 같은 서버-클라이언트 구조가 일반적이었다. 하지만 실시간 비디오 스트리밍을 사용하는 응용 프로그램과 사용자가 급증함에 따라 확장성이 높고 저비용으로 인프라를 구성할 수 있는 피어투피어 방식의 스트리밍 연구가 활발하게 진행되고 있다.

피어투피어 스트리밍 방식은 크게 트리-푸시(tree-push) 기반과 메시-풀(mesh-pull) 기반으로 발전되어 왔다. 트리 기반 구조에서는 네트워크 상에 가상의 오버레이 트리 구조를 생성한 후 부모 피어가 자식 피어들에게 데이터를 푸시(push)해서 전송하는 방식을 사용하였다[1,2,3,4,5]. 일단 트리가 생성되면 전송 지연 시간이 짧기 때문에 효과적이지만 각 피어가 서비스에 가입하거나 탈퇴할 때마다 트리를 새롭게 구성해야 하는 오버헤드가 존재한다. 특히 루트에 가까운 피어가 이탈하거나 오류로 인해 이탈할 경우 트리를 재구성하는 지연 시간이 상당히 길어지는 단점이 있다.

이러한 트리-푸시 구조의 문제점으로 인해 이 후 메시-풀 기반 구조에 대한 연구가 등장하였다. 이 구조는 각자 저장하고 있는 데이터를 버퍼맵(buffer map)에 표기하고 이웃 피어들에게 전송을 통해 서로의 데이터를 파악, 필요한 데이터를 요청하는 풀(pull) 방식을 사용한다[6,7,8,9]. 메시-풀 기반 구조는 메시 구조의 연결을 통해 트리기반 구조에 비해 피어들의 접속 및 이탈로 인한 피어 연결의 재구성에 따른 지연시간의 영향이 적다는 점과 참여한 피어들이 많을수록 성능이 더욱 향상된다는 장점이 있다. 하지만 풀 방식의 요청으로 소스 서버와의 재생 시간차(playback lag time)가 길어지고 데이터 도착 순서가 불규칙하여 끊김 없는 영상 재생을 위해 일정부분까지 전송 받은 후 재생하도록 버퍼링을 설정하여 초기 재생 시작이 지연되는 문제점이 있다.

트리-푸시 구조와 메시-풀 구조의 문제점의 발생으로 두 구조를 통합한 하이브리드형 구조인 푸시-풀(Push-Pull) 구조에 대한 연구가 등장하였다[10,11]. mTreebone의 경우 트리-푸시 구조 중심의 상태에서 트리의 재구성이 발생할 경우 분산 헤시 테이블을 사용하여 메시-풀 구조로 주위 피어들과 연결을 한다. 이후 버퍼맵을 통해 트리의 재구성이 되는 동안 데이터를 전송받는 구조이다. 하지만 mTreebone의 경우 트리-푸시 기반을 중점으로 구성되기 때문에 접속과 이탈이 자주 반복될 경우 트리가 새롭게 구성할 때 지연 시간이 발생한다.

따라서 본 논문에서는 트리-푸시 구조와 메시-풀 구조의 장점을 동시에 활용하여 푸시-메시 구조의 피어투피어 스트리밍 기법을 제안한다. 이 기법에서는 높은 네트워크 업로드 성능을 가지는 피어를 최대한 활용하여 초기 재생 지연 시간 및 소스 서버와의 시간차를 줄이고 연속 재생률을 증가시킨다. 기본적으로 메시-풀 구조를 기반으로 하며 소스 서버와 수퍼 피어 및 수퍼 피어와 특정수의 일반 피어들 사이에 푸시 방식의 데이터 전송을 지원한다.

NS-2를 이용한 실험을 통해 기존의 메시-풀 구조에 비해 초기 재생 지연 시간 및 소스 서버와의 시간차가 상당히 짧다는 것을 보인다. 또한, 트리-푸시 구조에 비해서도 재생의 연속성 비율이 높다는 것을 보인다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 피어투피어 스트리밍 기법에 대한 관련 연구를 기술한다. 3장에서는 본 논문에서 제시하는 푸시-메시 구조의 피어투피어 스트리밍 기법에 대해 설명한다. 4장에서는 푸시-메시 구조의 성능을 기존의 트리-푸시 및 메시-풀 방식과 비교 분석한다. 마지막으로 5장에서 결론을 서술한다.

Ⅱ. 관련 연구

실시간 방송, VOD 등을 포함하는 IPTV 서비스는 네트워크, 컴퓨팅 기술, 단말기술 등의 컴퓨터 전 분야의 기술을 포괄하는 응용으로서 최근 이 분야에 대한 연구가 활발히 진행되고 있다. 특히, 본 논문의 연구 분야로서 높은 확장성과 저비용 설치가 가능한 피어투피어 스트리밍 방식을 이용한 실시간 IPTV 방송에 대한 많은 연구 성과가 나오고 있다[12,13,14,15]. 피어투피어 스트리밍은 가상의 오버레이 네트워크를 이용하여 피어 간에 프로그램을 양방향으로 전송하는 방식으로서 트리-푸시 방식, 메시-풀 방식, 그리고 이 두 방식을 통합한 하이브리드 구조인 푸시-풀 구조 등이 제안되었다.

트리-푸시 기반의 피어투피어 스트리밍 기법은 네트워크 상에서 트리 기반의 가상의 오버레이를 생성한 후 루트인 미디어 소스 서버에서 마지막 리프 피어들까지 부모 피어가 하위 자식 피어들에게 푸시 형태로 데이터를 전송하는 방식이며 ZIGZAG[1], SplitStream[2], P2PR-Tree[3], PTree[4], TreeP[5] 등과 같은 기법들이 제안되었다. ZIGZAG는 상위 피어가 자신을 제외한 하위 클러스트 그룹 모두에게 데이터를 전송하는 구조를 제안하여 기존 트리 구조의 초기 지연 시간 및 소스 서버와의 시간차(lag time)를 감소시켰다. SplitStream은

각 피어당 피어끼리의 연결 경로를 4-5 개까지 유지를 하며 특정 피어가 이탈을 하더라도 다른 경로를 통해 데이터를 끊김 없이 전송받아서 트리 재구성하는데 필요한 시간을 감소시켰다. P2PR-Tree는 R-Tree를 트리 구조에 적용하여 자식을 분할하거나 합병하는 구조를 제시하여 기존 트리 구조보다 빠른 트리의 재구성 방법을 제시하였다. 이와 같은 트리-푸시 구조 기반의 피어투피어 스트리밍에 대한 많은 연구가 진행되었지만 피어 이탈시 트리구조가 재구성됨에 따라 발생하는 지연 시간 문제는 여전히 큰 오버헤드로 남아 있다.

트리-푸시 기반 구조에서 피어 이탈시 트리를 재구성하는 시간이 길어지는 단점을 해결하기 위해 PPLive[6], AnySee[7], CoolStreaming[8], ContinuStreaming[9] 등과 같은 메시-풀 구조 기반의 피어투피어 스트리밍 기법들이 제안되었다. 메시 구조를 생성하기 위해 사용하는 방식으로 가십(gossip)과 트래커(tracker) 서버 방식으로 나눌 수 있다. CoolStreaming 등이 사용하는 가십 방식에서는 들어온 피어는 일단 연결이 가능한 피어에게 연결을 한다. 이후 각 피어가 원하는 데이터를 찾기 위해 이웃 피어들을 이용하여 전송 요청을 계속 전달하는 방식이다. 이 방식은 원하는 피어들을 찾기 위해 정보를 이웃 피어들을 통해 계속 전달해야 하므로 검색 시간이 길다는 단점이 있다. 그에 비해 트래커 서버의 경우 접속한 피어들이 가진 데이터 정보를 유지하고 접속한 피어에게 적절한 이웃 피어를 선별하여 연결시킴으로 최적의 구조를 유지시킨다. 또한 접속 피어들이 트래커 서버에게 일정시간 마다 정보를 전송함으로써 트래커 서버가 피어들의 접속 상태 및 저장 데이터 등을 직접 관리한다. 피어들은 새로운 정보를 얻기 위해 트래커 서버에게 검색을 요청하고 트래커 서버가 가지고 있는 정보에서 검색을 함으로써 검색시간을 줄이게 된다. 단점으로는 피어들의 정보 유지를 위해 트래커 서버를 계속적으로 유지함에 따른 오버헤드가 발생한다. 이후 가십 방식을 이용하지만 검색속도를 줄이기 위해 분산 헤시 테이블(DHT)을 적용한 CountinuStreaming 구조를 제안하였다. 메시 구조 기반의 피어투피어 스트리밍 역시 많은 연구가 진행되었지만 버퍼맵을 통해 데이터를 일정부분까지 전송 받은 후 재생을 시작할 수 있기 때문에 버퍼링에 따른 초기 지연이 커지는 단점이 존재한다.

트리-푸시 구조의 트리 재구성 지연 시간 문제와 메시-풀 기반의 초기 버퍼링으로 인한 재생 지연 문제를 해결하기 위해 두 구조를 다양한 형태로 통합하는 푸시-풀 기반의 피어투피어 스트리밍 기법이 등장하였지만 이 기법도 대부분의 경우 트리-푸시 기반으로 수행되므로 접속과 이탈이 빈번한 상황에서는 트리 재구성으로 인한 지연 시간이 발생하게 된다.

III. 푸시-메시 기반의 피어투피어 스트리밍

최근 네트워크 기술의 발전으로 10M~100Mbps의 높은 네트워크 업로드 성능을 보유한 피어들이 증가하고 있다. 본 논문에서 제안하는 푸시-메시 기반의 피어투피어 스트리밍 기법은 이와 같이 높은 네트워크 업로드 성능을 가지는 피어를 최대한 활용하여 초기 재생 지연 시간 및 소스 서버와의 시간차를 줄이고 재생 연속성을 증가시킨다.

푸시-메시 스트리밍 기법은 기존 트리-푸시 구조와 메시-풀 구조의 조합한 형태로 구성되는 반면 기존 구조에서는 제안되지 않은 구조로 네트워크 업로드 성능이 우수한 피어들을 수퍼 피어로 선정하여 일반 피어들에게 푸시 방식으로 데이터를 전송함으로써 성능을 향상시킨다.

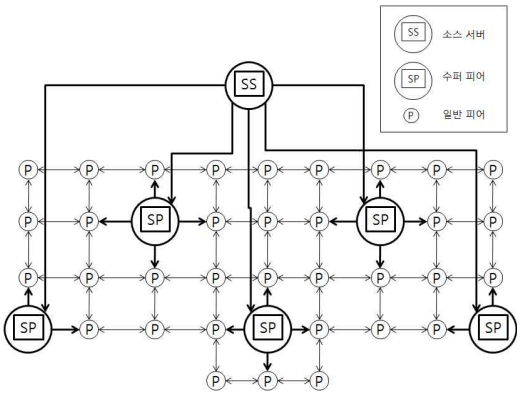


그림 1. 푸시-메시 구조 기반의 피어투피어 스트리밍
Fig 1. P2P Streaming based on Push-Mesh Structure

1. 시스템 구성요소

이 절에서는 <그림 1>에서 보이는 푸시-메시 기반의 피어투피어 스트리밍 기법을 구성하는 시스템 구성요소에 대해서 설명한다.

1.1 소스서버

소스 서버는 방송되는 스트림들을 각 채널별로 현재 접속되어 있는 수퍼 피어들에게 푸시 방식으로 데이터를 전송하는 역할을 수행한다. 이 방식은 트리-푸시 구조에서 루트 피어가 자신의 모든 자식 피어에게 전송하는 것과 유사하다. 하지만 푸시-메시 구조에서는 수퍼 피어와 수퍼 피어가 선정한 제한된 수의 일반 피어에게만 푸시 방식으로 데이터가 전송된다는 점에서 다르다. 또한, 메시-풀 구조에서는 소스 서버도 피어들

과 버퍼맵 교환을 통해 메시 구조를 통한 풀 방식으로 데이터를 전송한다는 점에서 푸시-메시 구조와 다르다.

1.2 트래커 서버

트래커 서버는 메시-풀 구조에서와 같이 피어들의 접속 및 이탈을 관리하는 역할을 수행한다. 피어들은 정상적인 이탈뿐만 아니라 하드웨어 장애 등으로 인해 비정상적으로 이탈하는 경우도 발생한다. 트래커 서버는 이런 경우를 감지하기 위해 각 피어로부터 일정시간마다 주기적으로 라이브 메시지를 전송받는다. 일정시간동안 특정 피어로부터 라이브 메시지를 전송받지 못하면 그 피어가 비정상적으로 이탈했다고 판단한 후 피어 이탈 알고리즘을 수행한다. 접근 및 이탈에 대한 상세한 알고리즘은 3.4절에서 설명한다.

또한, 접속하는 각 피어의 네트워크 업로드 성능에 따라 수퍼 피어와 일반 피어로 지정한다. 즉, 각 피어가 접속할 때 네트워크 업로드 성능을 파악하여 현재의 특정 수퍼 피어보다 업로드 성능이 우수하면 접속한 피어를 기존 수퍼 피어를 대체해서 수퍼 피어로 설정한다. 그렇지 않다면 일반 피어로 지정하여 기존 수퍼 피어 중 하나에 연결시킨다.

1.3 수퍼 피어

수퍼 피어는 소스 서버로부터 푸시 방식으로 데이터를 전송받는다. 또한, 네트워크 업로드 성능이 일반 피어에 비해 상대적으로 높기 때문에 높은 업로드 성능을 이용하여 제한된 수의 일반 피어들에게 푸시 방식으로 데이터를 전송한다.

1.4 일반 피어

모든 일반 피어들은 이웃 일반 피어들과 버퍼맵 교환을 통해 원하는 데이터를 풀 방식으로 전송받는다. 또한, 초기 재생 지연 시간과 소스 서버와의 재생 시간차를 줄이기 위해 해당 수퍼 피어의 네트워크 업로드 성능에 따라 특정수의 일반 피어는 푸시 방식으로도 데이터를 전송받는다. 일반 피어는 기본적으로 메시-풀 구조이기 때문에 피어 이탈시 구조 재구성에 따른 영향이 적다.

2. 푸시-메시 구조

본 논문에서 제안하는 푸시-메시 기반의 피어투피어 스트리밍 기법에서는 피어의 대부분이 기존의 메시-풀 구조를 기반으로 하는 일반 피어들이다. 이 구조는 피어의 이탈시 메시 구조를 재구성 하는 동안에도 연결된 다른 피어에게 데이터를 받을 수 있는 장점이 있다. 이 구조를 바탕으로 소스 서버와 수퍼 피어 및 수퍼 피어와 특정수의 일반 피어들 사이에 푸시 방식의 데이터 전송을 활용함으로써 초기 재생 지연 시간과 소스 서버와의 재생 시간차를 줄인다.

<그림 1>에서 보듯이 소스 서버는 연결된 모든 수퍼 피어들에게 데이터를 푸시 기반으로 전송한다. 소스 서버의 전송 능력이 우수할수록 많은 수퍼 피어들과 연결할 수 있다. 따라서 각 수퍼 피어를 통해서 더 많은 일반 피어에게 푸시 기반으로 데이터를 전송할 수 있기 때문에 성능 향상이 가능하다. 이때 피어의 능력을 측정하여 수퍼 피어와 일반 피어로의 구분이 이루어진다. 수퍼 피어는 접속 중인 피어 중 네트워크 업로드 성능이 가장 좋은 피어를 선택하게 되며, 수퍼 피어의 개수는 소스 서버의 성능에 의해 결정된다.

수퍼 피어는 소스 서버에게서 푸시 방식으로 전송받은 데이터를 특정수의 일반 피어에게 푸시 기반으로 전송한다. 일반 피어는 기본적으로 이웃 일반 피어들과 메시-풀 방식으로 데이터를 전송받으며 자신의 속한 수퍼 피어의 전송 능력에 따라 푸시 기반으로 데이터를 전송받을 수도 있다.

이러한 구조는 푸시 방식으로 데이터 전송을 중계하는 수퍼 피어가 이탈하더라도 구조를 재구성할 때 발생하는 전송 지연을 최소화한다. 이것은 수퍼 피어에 속한 다른 일반 피어들이 메시-풀 구조로 구성되어 있기 때문이다. 다시 말해서, 푸시-메시 구조는 트리-푸시 구조의 문제점인 피어 이탈시 구조 재구성에 따른 지연 시간 문제를 메시-풀 구조로 극복하면서 트리-푸시 구조의 장점인 짧은 초기 재생 지연 시간과 작은 소스 서버와의 재생 시간차를 제공하는 효율적인 구조이다.

3. 피어 접속 알고리즘

이 절에서는 새로운 피어의 접속시 푸시-메시 구조에 추가되는 알고리즘을 설명한다.

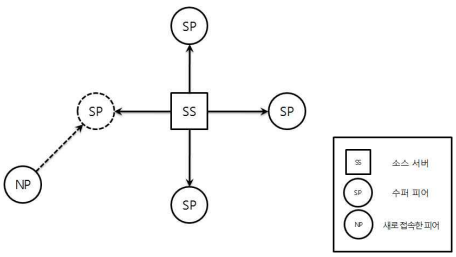


그림 2. 접속 피어를 수퍼 피어로 설정하는 경우
Fig 2. A Case Where a Joining Peer Becomes a Super Peer

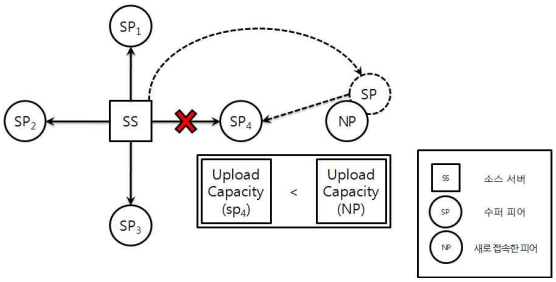


그림 3. 접속 피어를 성능 낮은 기존 수퍼 피어를 대신해서 수퍼 피어로 설정하는 경우
Fig 3. A Case Where a Joining Peer Becomes a Super Peer by Replacing the Existing Super Peer with Low Performance

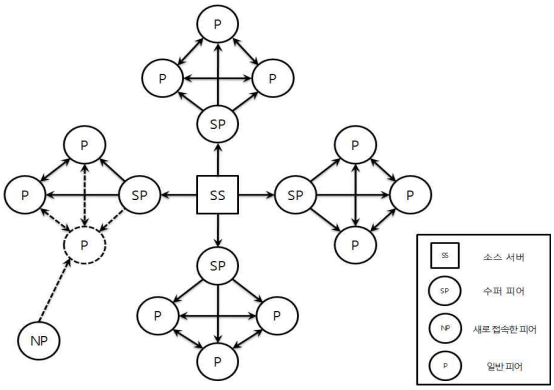


그림 4. 접속 피어를 수퍼 피어로부터 푸시 방식으로 데이터를 전송받는 일반피어로 설정하는 경우
Fig 4. A Case Where a Joining Peer Becomes a Normal Peer that Receives Data From its Super Peer with Push Mechanism

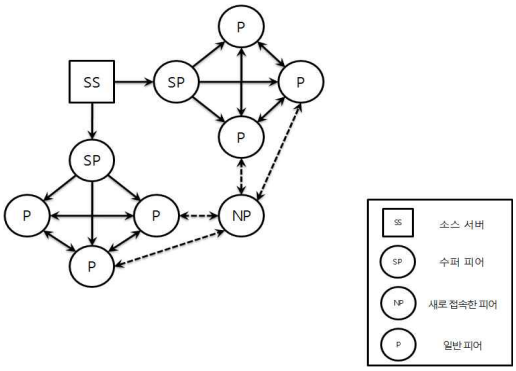


그림 5. 접속 피어를 일반 피어로 설정하는 경우
Fig 5. A Case Where a Joining Peer Becomes a Normal Peer

표 1. 피어 접속 알고리즘
Table 1. Peer Joining Algorithm

```
if(소스 서버가 연결 가능한 최대 수퍼 피어 수> 현재
  수퍼 피어수)
{
  수퍼 피어로 설정 후 소스 서버에 연결;
}
else
{
  접속 피어의 성능(네트워크 업로드 성능) 측정;
  if(접속 피어의 성능 > 가장 성능이 낮은 수퍼 피어의
   성능)
  {
    접속 피어는 기존 수퍼 피어를 대신 수퍼 피어로 설
     정 후 소스 서버에 연결;
    기존 수퍼 피어는 일반 피어로 설정 후 새로운 수퍼
     피어에 푸시 방식으로 연결됨;
    접속 피어는 기존 수퍼 피어에서 푸시 방식으로 연결
     된 일반 피어의 주소 정보를 넘겨받아서 연결;
  }
  else
  {
    if(수퍼 피어가 푸시방식으로 연결 가능한 최대 피어
     수 > 현재 푸시방식으로 연결된 피어 수)
    {
      연결 가능한 피어수가 가장 많은 수퍼 피어를 선
       택;
      일반 피어로 설정 후 선택된 수퍼 피어와 푸시 방
       식으로 연결;
      이웃 일반 피어들과 메시 방식으로 연결;
    }
    else
    {
      일반 피어로 설정 후 이웃 일반 피어들과 메시 방
       식으로 연결;
    }
  }
}
```

트래커 서버는 피어 접속 시 소스 서버가 연결 가능한 최대 수퍼 피어수와 현재 수퍼 피어수를 비교한다. 소스 서버가 연결 가능한 최대 수퍼 피어 수가 더 많다면 <그림 2>에서 보듯이 접속 피어는 수퍼 피어로 설정 후 소스 서버에 연결하게 된다. 만약 소스 서버가 연결 가능한 최대 수퍼 피어 수가 적거나 같다면 접속 피어의 네트워크 업로드 성능을 측정한다. 그리고 접속 피어의 성능이 현재 수퍼 피어중 가장 성능이 낮은 수퍼 피어의 성능보다 높다면 <그림 3>에서 보듯이 접속 피어는 기존 수퍼 피어를 대신하여 수퍼 피어로 설정 후 소스 서버에 연결한다. 이때 기존 수퍼 피어는 일반 피어로 설정 후 새로운 수퍼 피어에 푸시 방식으로 연결된다. 또한, 새로운 수퍼 피어는 기존 수퍼 피어에서 푸시방식으로 연결된 일반 피어의 주소 정보를 받아서 연결한다. 만약 접속 피어의 성능이

현재 수퍼 피어 중 가장 성능이 낮은 수퍼 피어의 성능보다 낮거나 같다면 현재 수퍼 피어가 푸시 방식으로 연결 가능한 최대 일반 피어 수와 현재 푸시 방식으로 연결된 일반 피어 수를 비교한다. 수퍼 피어가 푸시방식으로 연결 가능한 최대 일반 피어 수가 더 많다면 <그림 4>에서 보듯이 현재 수퍼 피어 중 푸시 방식으로 연결 가능한 피어 수가 가장 많은 수퍼 피어를 선택한다. 이때 접속 피어를 일반 피어로 설정 후 선택된 수퍼 피어와 푸시 방식으로 연결하고 이웃 일반 피어들과 메시 방식으로 연결한다. 만약 현재 수퍼 피어가 푸시 방식으로 연결 가능한 최대 일반 피어 수가 적거나 같다면 <그림 5>에서 보듯이 접속 피어는 일반 피어로 설정 후 이웃 일반 피어들과 연결한다.

접속한 피어는 수퍼 피어와 일반 피어 중 하나로 설정되며 수퍼 피어일 경우 소스 서버와 일반 피어에게 푸시 방식으로 연결하게 된다. 일반 피어일 경우 수퍼 피어에게 접속이 가능할 경우는 푸시 방식으로 연결하여 데이터를 받는 동시에 이웃 일반 피어들과 메시 방식으로 연결하여 버퍼맵을 통해서도 데이터를 전송 및 데이터를 받는다.

4. 피어 이탈 알고리즘

접속된 피어는 이탈시 정상적인 이탈과 라이브 메시지가 일정시간 동안 도착하지 않을 경우의 이탈이 발생된다. 모든 이탈시 이탈 피어가 수퍼 피어일 경우와 일반 피어일 경우에 대해 동작하는 알고리즘을 설명한다.

표 2. 피어 이탈 알고리즘
Table 2. Peer Leaving Algorithm

```
피어 이탈 감지 (정상적인 이탈 혹은 라이브 메시지가 일
 정시간 동안 도착하지 않은 경우);
if(이탈한 피어가 수퍼 피어라면)
{
  모든 일반 피어 중 가장 성능(네트워크 업로드 성능)이
   높은 피어를 선정하여 새로운 수퍼 피어로 설정;
  새로운 수퍼 피어는 이탈한 수퍼 피어의 연결 정보를
   통해 소스 서버와 일반 피어에게 연결;
  새로운 수퍼 피어와 푸시 방식으로 연결 되어있던 수퍼
   피어와 메시 방식으로 연결 되어있던 일반 피어들은
   새로운 수퍼 피어로 인해 발생한 빈 연결공간에 새로
   운 연결 가능한 피어를 검색하고 연결;
}
else // 이탈한 피어가 일반 피어라면
{
  푸시 방식으로 연결 되어있던 수퍼 피어와 메시 방식으
   로 연결 되어있던 일반 피어들은 이탈한 피어로 인해
   발생한 빈 연결 공간에 새로운 연결 가능한 피어를 검
   색하고 연결;
}
```

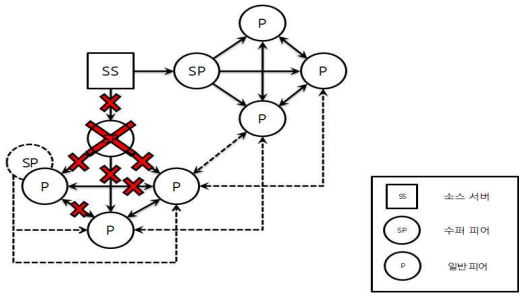


그림 6. 수퍼 피어가 이탈하는 경우
Fig 6. A Case Where a Super Peer Leaves its Group

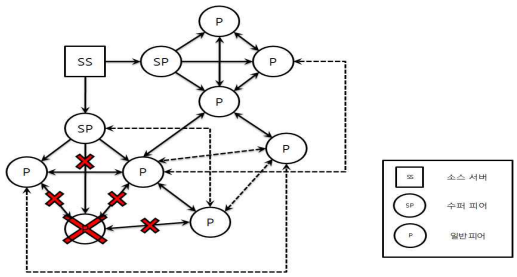


그림 7. 일반 피어가 이탈하는 경우
Fig 7. A Case Where a Normal Peer Leaves its Group

트래커 서버는 모든 피어들에게 일정 시간마다 라이브 메시지를 전달받는다. 만약 피어가 정상적인 이탈 메시지를 전송하거나 피어의 라이브 메시지가 일정시간 동안 도착하지 않으면 피어가 이탈했다고 판단한 후 새로운 연결 설정을 한다. 연결 설정 시 이탈한 피어가 수퍼 피어인지 일반 피어인지를 확인한다. 수퍼 피어일 경우 <그림 6>에서 보듯이 현재 접속된 모든 일반 피어 중 네트워크 업로드 성능이 가장 높은 피어를 선정하여 새로운 수퍼 피어로 설정한다. 그리고 이탈한 수퍼 피어의 연결 정보를 통해 소스 서버와 푸시 방식으로 일반 피어에 연결한다. 또한 선정된 피어와 푸시 방식으로 연결되었던 수퍼 피어나 메시 방식으로 연결되었던 일반 피어들은 선정된 피어가 수퍼 피어로 설정되므로 대신할 피어를 검색하여 연결시킨다. 만약 이탈한 피어가 일반 피어일 경우 <그림 7>에서 보듯이 푸시 방식으로 연결되었던 수퍼 피어나 메시 방식으로 연결되었던 일반 피어들은 이탈한 일반 피어를 대신할 피어를 검색하여 연결시킨다.

IV. 성능 평가

이 장에서는 본 논문에서 제안한 푸시-메시 구조의 피어투피어 스트리밍 기법과 기존의 트리-푸시 및 메시-풀 구조와의

성능을 비교 분석한다. 먼저 실험 환경 구성에 대해 기술한 후 초기 재생 지연 시간(startup delay), 소스 서버와의 재생 시간차(playout lag time), 그리고 재생 연속성(playback continuity index)에 대한 실험 결과를 설명한다.

1. 실험평가

가상 네트워크 토폴로지를 제공하는 NS-2 네트워크 시뮬레이터를 사용하여 트리-푸시, 메시-풀, 푸시-메시 구조의 피어투피어 스트리밍 기법을 구현하였다. NS-2에서 1000개의 피어와 100개의 라우터를 생성하였고 하나의 라우터에 10개의 피어와 2~4개의 라우터가 연결되도록 구성하였다. 또한, 10Mbps 및 100Mbps 네트워크 대역폭을 가지는 피어와 라우터 간 연결은 각각 900개 및 100개로 설정했고 라우터와 라우터 간 모든 연결은 100Mbps로 설정하였다.

먼저 접속 피어 수를 100개에서 1000개까지 증가시키면서 초기 재생 지연 시간 및 소스 서버와의 재생 시간차를 측정하였다. 또한, 동적으로 피어들의 접속 및 이탈을 반복하면서 재생의 연속성을 측정하였다.

2. 실험결과

2.1 초기 재생 지연시간

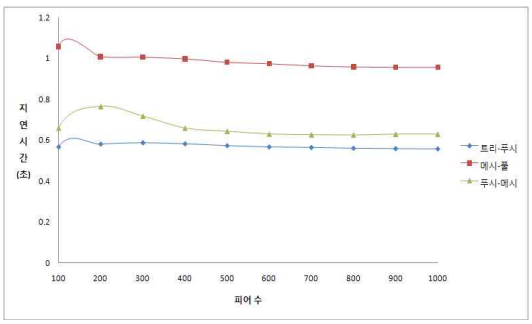


그림 8. 초기 재생 지연 시간
Fig 8. Startup Delay

<그림 8>은 접속 피어 수를 100개에서 1000개까지 증가시킬 때 각 피어투피어 스트리밍 기법의 초기 재생 지연시간을 나타낸다. 예상대로 전체 피어가 푸시 방식으로 데이터를 전송하는 트리-푸시 구조가 피어 이탈이 없는 경우 다른 두 구조에 보다 초기 재생 지연 시간이 가장 짧았다. 예를 들어 피어수가 600개인 경우 트리-푸시, 푸시-메시 및 메시-풀은 각각 0.57초, 0.63초, 0.97초의 초기 재생 지연시간이 걸렸다.

2.2 소스 서버와의 재생 시간차

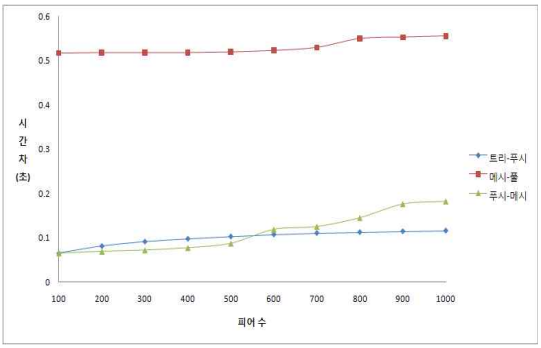


그림 9. 소스 서버와의 재생 시간차
Fig 9. Playback Lag Time

<그림 9>은 접속 피어 수를 100개에서 1000개까지 증가시킬 때 세 구조에서의 소스 서버와의 재생 시간차를 보여준다. 실험 결과 푸시-메시와 메시-풀이 비슷한 성능을 보였고 메시-풀의 약 5배 긴 소스 서버와의 재생 시간차를 보였다. 메시-풀 기반의 피어투피어 스트리밍은 버퍼맵을 사용하여 데이터를 서로 주고받기 때문에 홉 수가 높아질수록 소스 서버와의 시간차가 길어지는 것을 알 수 있다. 트리-푸시는 소스 서버에서부터 데이터를 전송받아 연결된 피어에게 푸시하기 때문에 소스 서버와의 재생 시간차가 상대적으로 짧았다. 푸시-메시는 소스 서버와 수퍼 피어가 푸시 방식으로 빠르게 데이터를 전송하기 때문에 메시-풀보다 21% 짧은 소스 서버와의 재생 시간차를 보이며 트리-푸시와 거의 비슷한 시간차를 보였다.

또한, <그림 9>에서 보듯이 피어 수가 증가함에 따라 푸시-메시가 트리-푸시보다 소스 서버와의 재생 시간차가 길어지는 것을 알 수 있다. 이것은 푸시-메시의 수퍼 피어들이 푸시 방식으로 데이터를 전송할 수 있는 일반 피어의 수가 제한되어 있어서 피어 수가 증가함에 따라 풀 방식으로 데이터를 전송받는 피어가 수가 증가하기 때문이다

2.3 재생 연속성

<그림 10>은 접속 피어 수를 1000개로 설정 후 각 피어가 8번의 이탈과 접속을 반복할 때 각 구조의 재생 연속성을 보여준다. 이때 연속성의 값은 마감 시간 전에 도착한 데이터의 비율이다. 실험 결과 푸시-메시 구조가 메시-풀 및 트리-푸시에 비해 평균적으로 12% 및 45% 높은 재생 연속성을 보였다. 트리-푸시는 피어 이탈시 피어의 하위 트리를 재구성하는 지연 시간이 길다는 문제가 있기 때문에 접속과 이탈의 반복에 따라 재생의 연속성이 상대적으로 낮았다. 메시-풀은 피어 이

탈시 피어와 연결되었던 다른 피어들이 메시를 재구성 하는 동안에도 다른 연결된 피어에게 데이터를 받을 수 있기 때문에 재생의 연속성이 상대적으로 높았다. 푸시-메시는 일반 피어들이 기본적으로 메시 기반으로 구성되기 때문에 메시-풀이 가진 장점을 유지할 수 있었다. 또한, 수퍼 피어를 사용하여 데이터 전송의 속도를 증가시켰기 때문에 마감 시간 내에 도착하는 데이터 비율을 더욱 높일 수 있었다.

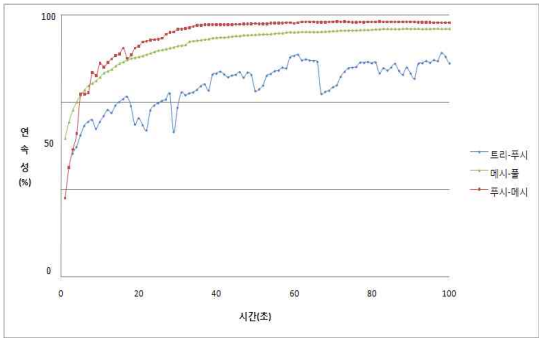


그림 10. 재생 연속성
Fig 10. Playback Continuity Index

V. 결론

피어투피어 기술은 확장성과 저비용으로 인해 실시간 스트리밍 기술에 활발히 연구되고 있다. 본 논문에서는 기존의 트리-푸시 구조에서 피어 이탈시 트리를 재구성할 때 발생하는 오버헤드 문제와 메시-풀 구조에서 초기 재생 지연 시간과 소스 서버와의 재생 시간차가 크다는 단점을 극복하기위해 새로운 구조인 푸시-메시 기반의 피어투피어 스트리밍 기법을 제안하였다. 이 기법은 네트워크 업로드 성능이 우수한 수퍼 피어를 통해 푸시와 풀 구조의 장점을 동시에 활용하였다. 또한, NS-2시뮬레이터를 이용한 실험을 통해 푸시-메시 구조가 기존 구조들에 비해 성능이 향상되었음을 보였다. 초기 재생 지연시간은 메시-풀보다 약 35% 감소하였다. 소스 서버와의 재생 시간차는 메시-풀보다 21% 짧아 졌고, 트리-푸시와 비슷한 시간차를 보였다. 또한, 영상 재생의 연속성에서는 푸시-메시 구조가 메시-풀 및 트리-푸시에 비해 평균적으로 12% 및 45% 높아졌다는 것을 보였다.

이와 같은 연구 성과를 바탕으로 향후에는 피어 그룹핑시 채널에 따라 각 피어의 지역성을 고려하며 실시간 방송뿐만 아니라 일시정지, 재시작, 빨리 감기, 되감기 등의 VCR 기능을 지원하는 연구를 진행할 계획이다.

참고문헌

[1] D. A. Tran, K. A. Hua, and T. Do, “ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming,” Proc. IEEE INFOCOM, pp.1283-1292, 2003.

[2] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," Proc. ACM SOSP, pp.298-313, 2003.

[3] A. Mondal, Y. Lifu, and M. Kitsuregawa. "P2PR-Tree: an R-Tree-based Spatial Index for Peer-to-Peer Environments," Proc. EDBT Workshops, pp.516-525, 2004.

[4] A. Crainiceanu, P. Linga, J. Gehrke, and J. Shanmugasundaram, "P-Tree: a P2P Index for Resource Discovery Applications," Proc. World Wide Web, pp.390-391, 2004.

[5] B. Hudzia, M. T. Kechadi, and A. Ottewill, "TreeP: a Tree based P2P Network Architecture," Proc. IEEE Cluster Computing, pp.1-15, 2005

[6] X. Hei, C. Liang, J. Liang, Y. Liu and K. . Ross, “a Measurement Study of a Large-Scale P2P IPTV System,” Proc. IEEE Multimedia, vol. 9, no. 8, pp.1672-1687, 2007.

[7] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Scalable Live Streaming Service based on Inter-Overlay Optimization," Proc. IEEE INFOCOM, pp.1663-1674, 2006.

[8] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: a Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," Proc. IEEE INFOCOM, pp.2102-2111, 2005.

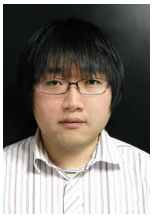
[9] Z. Li, J. Cao, G. Chen, "ContinuStreaming: Achieving High Playback Continuity of Gossip-based P2P Streaming," Proc. IEEE IPDPS, pp.1-12, 2008

[10] F. Wang, Y. Xiong, J. Liu, "mTreebone: a Hybrid Tree/Mesh Overlay for Applicationlayer Live Video Multicast," Proc. IEEE ICDCS, pp.49-49, 2007.

[11] M. Zhang, J. G. Luo, L. Zhao, and S. Q. Yang, "a Peer-to-Peer Network for Live Media Streaming Using a Push-Pull Approach," Proc. ACM Multimedia, pp.287-290, 2005.

[12] Y. Liu, L. Guo, F. Li and S. Chen, “A Case Study of Traffic Locality in Internet P2P Live Streaming Systems“, Proc. IEEE ICDCS, pp.423-432, 2009.
[13] C. Wu, B. Li and S. Zhao, "Diagnosing Network-Wide P2P Live Streaming Inefficiencies", Proc. IEEE INFCOM, pp.2731-2735, 2009
[14] 김종경, 최성욱, “P2P 환경에서 복합 스트림 서비스를 위한 효율적인 오버레이 멀티캐스트 스케줄링”, 한국컴퓨터정보학회 논문지, Vol. 13, No. 6, 233-241쪽, 2008년 11월.
[15] 정의현, “다중 피어 결합을 이용한 P2P 멀티미디어 스트리밍 프로토콜”, 한국컴퓨터정보학회 논문지, 제 11권, 제 2호, 253-261쪽, 2006년 5월.

저 자 소 개



김 진 성

2008 : 홍익대학교 컴퓨터공하과 학사
2010 : 홍익대학교 컴퓨터공학과 석사
현재 : 홍익대학교 컴퓨터공학과 박사 과정
관심분야 : P2P 비디오 스트리밍 시스템, 멀티미디어 네트워킹

김 동 일

2009 : 홍익대학교 컴퓨터공하과 학사
현재 : 홍익대학교 컴퓨터공학과 석사 과정
관심분야 : P2P 비디오 스트리밍 시스템, 멀티미디어 네트워킹

김 은 삼

1994 : 서울대학교 컴퓨터공하과 학사
1996 : 서울대학교 컴퓨터공학과 석사
1996-2002 : LG전자 선임연구원
2006 : Univ. of Florida 컴퓨터공학과 박사
현재 : 홍익대학교 컴퓨터공학과 조교수
관심분야 : 분산 멀티미디어 시스템, IPTV 시스템, 컴퓨터 저장시스템

배 성 일

1993 : 서울대학교 수학과 학사
1997 : Univ. of Illinois at Urbana-Champaign 수학과 석사
2005 : Univ. of Illinois at Urbana-Champaign 전산학과 박사
현재 : 홍익대학교 컴퓨터공학과 전임 강사
관심분야 : 알고리즘, 계산이론