P2P 스트리밍 시스템의 성능 평가를 위한 NS2 기반 시뮬레이터 개발

김 혜 선[†] · 황 기 태[††]

요 약

인터넷 스트리밍 시스템은 스트리밍 소스를 공급하는 미디어 서버와 이로부터 미디어 스트림을 받아 분배하는 스트리밍 서버, 그리고 스트리밍 단말기들로 구성되며, 기존에는 하나의 스트리밍 서버에 다수의 단말기들이 직접적으로 연결되는 방식을 취하고 있다. 이러한 클라이언트-서버 형태의 중앙 구조에서는 스트리밍 서버에 대한 트래픽 집중으로 병목 현상이 발생하며, 한 스트리밍 서버의 용량에 따라 스트리밍 단말기의 개수가 제한되는 등 확장성 및 수용 능력의 한계라는 단점을 근본적으로 가지고 있다. 이러한 문제를 극복하여 확장성을 제공하기 위해 P2P 분산 아키텍처를 이용하는 P2P 스트리밍 시스템에 대한 연구가 최근 들어 진행되고 있다. 그러나 P2P 방식을 이용한 인터넷 스트리밍 시스템을 설계, 구현, 테스트하기 위해서는 실제 많은 컴퓨터들이 필요하며, 네트워크의 다양한 구성이나 트래픽 변화에 따른 실험을 하기에는 현실적인 어려움이 있다. 그러므로 본 논문에서는 P2P 스트리밍 시스템에 대한 다양한 연구 및 실험을 지원하기 위해, P2P 스트리밍 시스템에 대한 구조적 모델 및 동작 모델, 시간 모델, 성능 지수들을 정의하고, 네트워크 시뮬레이터를 지원하는 NS2 시뮬레이션 라이브러리를 이용하여 P2P 스트리밍 시스템 시뮬레이터 *P2PStreamSim*을 설계 및 구현하였다. 또한 테스트 P2P 스트리밍 시스템을 사례로 적용하여 *P2PStreamSim*의 동작을 검증하고 성능을 평가하였다.

키워드 : P2P 스트리밍 시스템, 시뮬레이터, NS2, 성능 평가

NS2 based Simulator for Performance Evaluation of P2P Streaming Systems

Hye Sun Kim[†] · Kitae Hwang[††]

ABSTRACT

Internet streaming systems consist of a media server, a streaming server, and terminals. The media server delivers multimedia contents such as video and/or audio to the streaming server, which distributes the contents to terminals as well. Existing Internet streaming systems have a bottleneck problem in the streaming server because of the limit of the processing capacity of the streaming server and therefore a streaming server can not accomodate more terminals than the limit. As a solution to this problem, P2P streaming systems have been lately proposed and investigated, using P2P distributed architectures. Actually, however, there exist many difficulties in the design and implementation of P2P streaming systems, because it needs many real computers and various network constructions. In this paper, we have proposed and defined a P2P streaming system model such as the architectural model, the timing model, the behavior model, and the performance metrics. And also we have implemented an NS2 based P2P streaming system simulator called *P2PStreamSim*. Finally, we have verified it through test simulations and analyzed the results.

Key Words : P2P Streaming System, Simulator, Performance Evaluation

1. 서 론

네트워크 속도 및 멀티미디어에 대한 컴퓨팅 처리 능력이 향상됨에 따라 화상 채팅, 비디오 회의, VOD(Video On Demand), E-learning 등 실시간 스트리밍에 대한 요구 및 응용 분야들이 많이 제시되고 지난 10년 동안 많은 연구들이 진행되어 왔다[1, 2]. 특히 최근에는 인터넷 방송 등 인터넷 스트리밍 시스템들이 개발 및 활용되고 있다. 인터넷 스트리밍 시스템은 스트리밍 소스를 공급하는 미디어 서버와 이로부터 미디어 스트림을 받아 분배하는 스트리밍 서버, 그리고 스트리밍 단말기들로 구성되며, 하나의 스트리밍 서버에 다수의 단말기들이 직접적으로 연결되는 클라이언트-서버 형태의 구조에서는 스트리밍 서버에 대한 트래픽 집중으로 병목 현상이 발생하며, 한 스트리밍 서버의 용량에 따라 스트리밍 단말기의 개수가 제한되는 등 확장성 및 수용

※ 본 연구는 2007년도 한성대학교 교내연구비 지원과제임
† 정 회 원 : (주)코난테크놀로지 연구원
†† 정 회 원 : 한성대학교 컴퓨터공학과 교수 (교신저자)
논문접수 : 2007년 3월 21일, 심사완료 : 2007년 7월 11일

능력의 한계라는 단점을 근본적으로 가지고 있다.

최근 들어 인터넷 스트리밍 시스템에 P2P 네트워크를 이용함으로써 이러한 단점을 해결하기 위한 연구들이 다소 진행되고 있다[3, 4, 5]. P2P 스트리밍 시스템이란 스트리밍 소스가 존재하며 이를 소비하고자 피어들이 직접적으로 스트리밍 소스에 접속하여 스트리밍 데이터를 받아 재생하거나, 스트리밍이 진행 중인 다른 피어에 접속하여 스트림 데이터를 중계받아 재생하는 방식의 시스템을 말한다. P2P 방식의 인터넷 방송을 위해 개발된 오픈 소프트웨어인 PeerCast[6]나, AnySee[7] 등은 P2P 스트리밍 시스템의 한 사례이다. T. Hama[8] 등은 피어의 이탈에 따라 하위 피어들의 스트림 끊김 현상을 해결하기 위해 피어와 피어 사이에 이중 링크를 이용하는 방안을 제시하였으며 CoopNet[9]에서는 스트리밍 서버의 병목 현상을 완화시키기 위한 연구를 수행하였다. 그럼에도 불구하고 여전히 P2P 스트리밍 시스템에 대한 체계적인 설계, 구현, 성능 평가 등에 대한 연구 결과가 미비한 상태이며 대규모의 P2P 스트리밍 시스템의 경우나, 사용자 그룹에 따른 스트리밍 질의 제어 등 많은 연구 과제들이 남아있다.

이러한 연구를 진행함에 있어 실제 구현을 통한 실험을 위해 많은 컴퓨터들이 필요하며 다양한 형태의 네트워크, 네트워크 트래픽 변화, 멀티미디어 컨텐츠의 질, P2P 네트워크의 토폴로지 구성 알고리즘 등에 따른 성능 실험을 하기에는 현실적인 어려움이 있다. 그러므로 P2P 스트리밍 시스템에 대한 연구, 개발을 위해 필연적으로 시뮬레이터의 개발이 필요하다. 이에 본 논문에서는 P2P 스트리밍 시스템에 대한 시뮬레이터인 *P2PStreamSim*을 설계 구현하였다. P2P 스트리밍 시스템을 정의하고, 구조적, 시간적, 행동적, 성능적 관점에서 모델링하고 네트워크 시뮬레이션을 위해 많이 사용되는 NS2(Network Simulator 2)를 근간으로 시뮬레이터를 작성하였다. 또한 시뮬레이터의 동작 및 시뮬레이션을 정확성을 확인하기 위해 테스트 P2P 스트리밍 시스템을 정의하고 이를 시뮬레이션하고 성능을 분석하였다.

본 논문은 다음과 같이 구성된다. 2절에서는 NS2 시뮬레이션 도구에 대해 소개하고, 3절에서는 P2P 스트리밍 시스템을 모델링한다. 4절에서는 P2PStreamSim 시뮬레이터의 설계 구현에 대해 기술하며 5절에서는 테스트 P2P 스트리밍 시스템에 대한 시뮬레이션 결과를 보이고 6절에서 결론을 맺는다.

2. NS2(Network Simulator version 2)

2.1 NS2 개요

콜롬비아 대학에서 개발된 시뮬레이션 테스트베드인 NEST를 기반으로 UC 버클리에서 1988년에 REAL 네트워크 시뮬레이터를 개발하였고, 이것을 기반으로 1989년 LBNL(Lawrence Berkely National Laboratory)이라는 네트워크 연구 그룹은 NS1이라는 네트워크 시뮬레이터를 발표하였다. 1995년 VINT (Virtual InterNetwork Testbed) 프로젝트의 일환으로 NS1 시뮬레이터가 완성되었다. 1996년 MIT에서 개발하여 발표한 OTcl(Object Tcl)을 사용하여 NS1의 기능을 더욱 향상시킨 것이 NS2 이다[10, 11].

NS2는 TCP, UDP, HTTP 등과 같은 TCP/IP 프로토콜 패밀리와 라우팅 프로토콜, 그리고 멀티캐스팅 프로토콜 등과 같은 다양한 인터넷 프로토콜을 시뮬레이션 할 수 있으며 Ad Hoc 네트워크, 이동 통신망의 기지국(Base Station) 모델, WLAN, Mobile-IP 관련 프로토콜, 위성 네트워크(Satellite Network) 등과 같은 무선 네트워크까지 지원할 수 있는 매우 적용 범위가 넓은 네트워크 시뮬레이터이다.

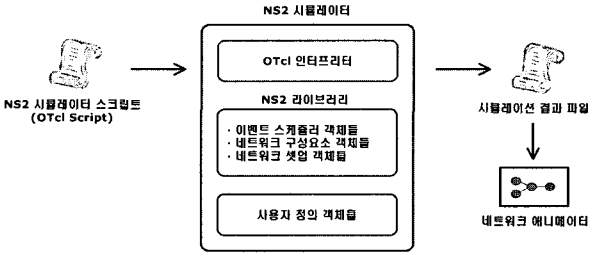
2.2 NS2 구조

(그림 1)은 사용자 관점에서 본 NS2 구조이다.

사용자는 OTcl 스크립트 언어를 사용하여 시뮬레이션 대상 시스템에 대한 네트워크 토폴로지 구성 및 작업 부하는 기술하는 시뮬레이션 스크립트 프로그램을 작성한다. OTcl 인터프리터는 시뮬레이션 스크립트 프로그램을 읽고 한 라인씩 시뮬레이션을 수행한다. 시뮬레이션은 이벤트 기반으로 작동되며 이벤트 처리의 핵심이 되는 이벤트 스케줄러는 단일 스레드로서, 한 번에 오직 한 가지 이벤트만을 순차적으로 처리한다.

NS2 라이브러리에서 지원되는 네트워크 컴포넌트는 노드, 링크, 에이전트 등과 같은 객체를 말한다. 노드란 라우터, 허브, 기지국 등과 같은 네트워크에서의 중계용 장비를 말하며, 이를 연결하는 객체가 링크이다. 에이전트는 TCP, UDP와 같은 전송 계층 프로토콜 역할을 담당하는 객체이다. 시뮬레이션 결과는 트레이스(trace) 파일로 저장되며 NAM(Network Animator)[11]을 이용하여 시뮬레이션 과정을 애니메이션을 재생할 수 있다.

NS2에서는 OTcl 스크립트 언어와 C++ 언어를 사용한다. 대부분의 네트워크 연구 분야는 다양하게 변하는 파라미터 및 네트워크 구조의 변화, 매우 복잡한 시나리오를 표현하고자 한다. OTcl 스크립트 언어로는 사용자 인터페이스와 네트워크 토폴로지에 대한 매개 변수 등을 정의하며, C++로는 주로 네트워크 구성 요소를 객체 형태로 정의한다. 그리고 OTcl과 C++ 언어를 서로 연동시키기 위해 인터프리터를 사용한다. 시뮬레이터의 작성자는 NS2가 가진 기본 객체들을 그대로 사용하거나 상속받아 자신의 특별한 객체들을 작성한다, 필요하면 OTcl 기반의 스크립트 프로그램을 작성하면 된다.



(그림 1) NS2 구조

3. P2P 스트리밍 시스템 모델링

3.1 P2P 스트리밍 시스템 정의

본 절에서는 P2P 시스템을 보다 명확히 정의한다. P2P 스트리밍 시스템은 구조적 측면에서 인코딩 서버(Encoding Server: ES), 스트리밍 서버(Streaming Server: SS), 프록시 서버(Proxy Sever : PS), 그리고 피어(Peer: PE)들로 구성된다. ES 는 라이브 혹은 저장 소스로부터 인코딩된 멀티미디어 스트림을 발생시키는 노드이며, SS는 ES로부터 인코딩된 스트림을 받아 자신에게 연결된 여러 PE들에게 실시간으로 분산 시키는 역할을 하며, PS는 PE 들의 도착 및 이탈 시에 P2P 네트워크의 토폴로지를 관리하는 노드로서 각 PE에 대해 서버(부모) PE를 결정하는 역할을 한다. PE는 멀티미디어 스트림의 소비자이면서 동시에 자신의 자식 PE들에게 실시간으로 멀티미디어 스트림을 릴레이하는 서버 노드의 역할을 수행한다.

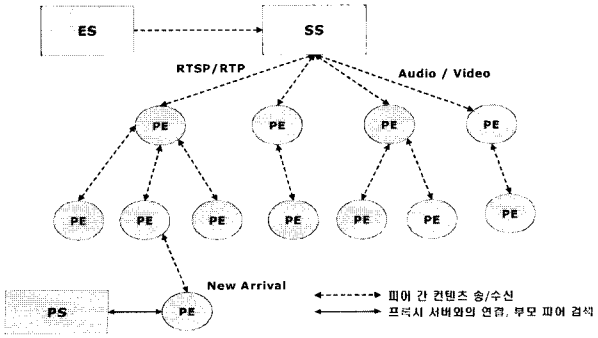
다음의 용어를 정의한다.

S_{es} : *ES의 집합*
S_{ss} : *SS의 집합*
S_{pe} : *PE의 집합*
P_{proxy} : *PS*
S_{pt} : *PE들 사이 또는 SS와 PE 사이의 통신프로토콜의 집합 (RTSP, RTP)*
$S_{streams}$: *멀티미디어 스트림들(비디어, 오디오)*
S_{link} : *PE 사이의 네트워크 링크의 속성*
S_{func} : *시스템 내 행동 모형의 집합(참여, 이탈, 재배치, 릴레이)*
S_{algo} : *시스템 내 알고리즘의 집합(부모 피어 선택 등)*

이때 P2P 스트리밍 시스템은 다음과 같이 정의한다.

"P2P 스트리밍 시스템은 { S_{es}, S_{ss}, S_{pe}, P_{proxy}, S_{pt}, $S_{streams}$, S_{link}, S_{func}, S_{algo}} 의 튜플로서 ES로부터 실시간으로 전송되는 멀티미디어 스트림을 일차적으로 SS가 받아 이를 실시간으로 PE 들에게 분산시키며 PE들은 자신이 이를 소비하면서 동시에 자신의 자식 PE 들에게 실시간으로 분산시키는 P2P 형식의 스트리밍 시스템이다."

(그림 2)는 본 논문에서 정의한 P2P 스트리밍 시스템의 한 사례로서 하나의 ES, SS, PS, 그리고 n 개의 PE 들로 구성된다. ES는 비디오 또는 오디오 데이터를 인코딩하여 SS로 RTP(Real-Time Transport Protocol)[12]를 이용하여 전송한다. 부모 피어와 자식 피어는 RTSP(Real Time Streaming Protocol)[13]로 연결되며 실시간 패킷 전송은 RTP를 이용한다. SS는 최상위 부모 피어의 역할로서, 자식 PE 들에게 수신한 패킷을 실시간으로 전달한다. PS는 피어의 도착과 이탈 시 필요에 따라 피어들에게 부모 피어를 결정하여 알려주며, 이에 따라 네트워크 토폴로지를 재구성한다.



(그림 2) P2P 스트리밍 시스템 모델 사례

3.2 행동 모델

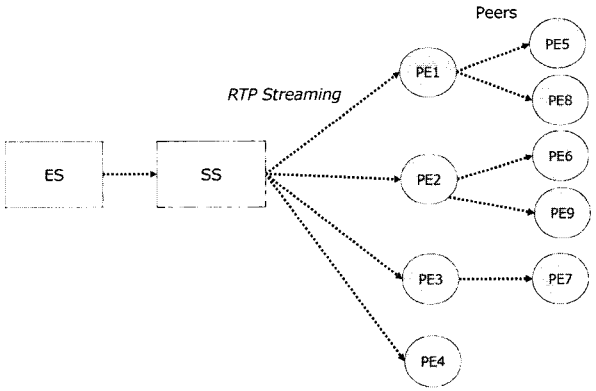
3.2.1 실시간 릴레이

ES로부터 말단 PE까지 스트리밍 서비스가 이루어지는 과정으로서 (그림 3)과 같이 묘사된다. ES는 일정한 시간간격으로 RTP 패킷을 생성하여 SS로 전달하며, SS는 이를 연결된 PE1, PE2, PE3, PE4에게 분배한다. 각 PE들은 SS로부터 수신한 RTP 데이터를 버퍼링한 후, 한 프레임씩 디코딩하여 재생한다. 또한 PE는 연결된 자식 PE들에게 각각 RTP 데이터를 릴레이 해주는 역할을 한다.

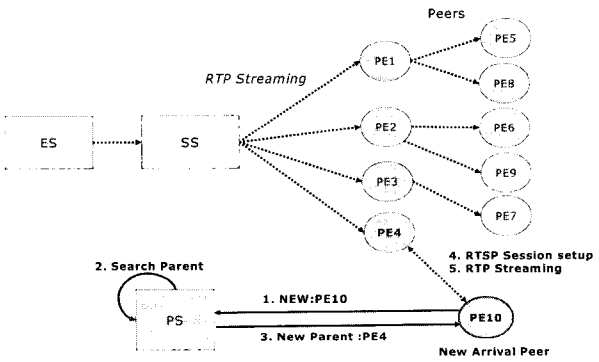
3.2.2 피어 참여

새로운 PE가 시스템 내에 도착하여 임의의 부모 피어를 할당받고 접속되는 과정으로서 (그림 4)와 같이 묘사된다.

현재 9개의 PE 들로 구성된 시스템 상황에서 새로 도착



(그림 3) 스트리밍 및 릴레이

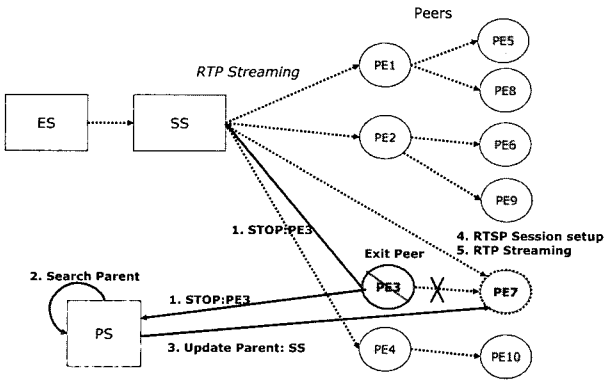


(그림 4) 피어 참여

한 PE10은 PS에게 자신의 존재를 알린다. PS는 적당한 부모 피어(PE4)를 탐색하여 PE10에게 알려준다. PE10은 PE4로 접속하여 RTSP 세션을 설정한 후 RTP 패킷을 수신한다. PE10은 수신한 패킷을 버퍼에 일시 저장하고 즉시 디코딩 후 재생한다.

3.2.3 피어 이탈 및 재배치

피어가 네트워크에서 이탈할 시, 연결된 자식 피어들이 새로운 부모 피어로 접속되는 과정으로서 (그림 5)와 같이 묘사된다. PE3은 이탈 시, PS와 부모 피어(SS)에게 알린다. PS는 적당한 부모 피어(SS)를 검색하여 PE7에게 새로운 부모 피어(SS)를 알려준다. PE7은 SS로 접속하여 RTSP 세션을 설정한다. PE7은 SS로부터 RTP 패킷을 수신하여 디코딩 후 재생한다.



(그림 5) 피어 이탈 및 재배치

3.3 시간 모델

3.3.1 시간 요소

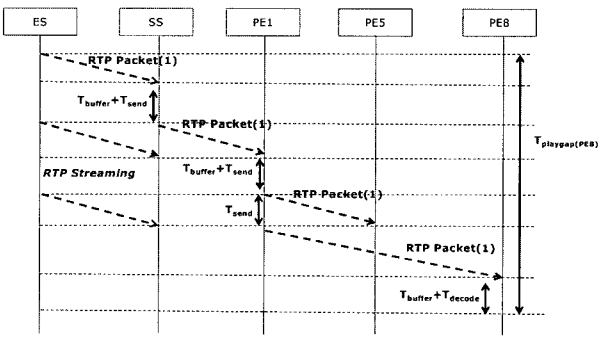
P2P 스트리밍 시스템에서 다음의 시간 요소들이 존재한다. 아래에서 부모 PE는 SS가 될 수도 있다.

- $T_{getserver}$: 임의의 PE가 PS에게 부모 PE를 알려줄 것을 요청한 시점으로부터 부모 PE를 통보받을 때까지의 시간 경과.
- $T_{p2pconnection}$: 임의의 PE가 자신의 부모 PE의 주소를 알고 있는 상황에서 부모 PE에 접속하여 RTSP 세션 설정을 완료할 때까지의 시간.
- $T_{initialframe}$: PE가 자신의 부모 PE에 접속된 후에 처음으로 하나의 프레임을 성공적으로 받을 때까지의 경과 시간.
- $T_{connection}$: 임의의 PE가 접속을 시작하여 부모 피어가 결정되고 부모와 접속한 후 첫 번째 프레임을 전송받을 때까지 지연 시간으로 $T_{getserver}$ + $T_{p2pconnection}$ + $T_{initialframe}$와 동일.
- T_{proxyq} : PE가 PS에게 부모 PE의 탐색을 요청하였을 경우, PS의 큐에서 대기하는 시간.
- T_{search} : PS가 하나의 PE에 대해 부모 PE를 결정하는데 걸리는 탐색 시간.
- T_{buffer} : PE가 자신의 릴레이 버퍼에서 읽거나 쓰는 시간.
- T_{decode} : PE에서 하나의 프레임이 디코딩되는데 소요되는 시간.
- T_{rtsp} : RTSP 스택을 통과하는데 소요하는 시간.
- $T_{proxyconnection}$: PE가 PS에게 요청 또는 응답하는 데 소요하는 전체 시간.
- T_{rtp} : 한 패킷이 RTP 프로토콜 스택을 통과하는데 걸리는 시간.
- T_{send} : PE가 하나의 패킷을 한 자식 PE에게 전송하는데 걸리는 시간.
- $T_{waitexit}$: 임의의 PE가 이탈하고자 하는 시점에서 P2P 시스템으로부터 완전히 분리될 때까지 기다릴 수 있다고 정의한 최소 시간. 이 시간은 사용자가 시스템에서 의해서 정의 가능함
- T_{exit} : 임의의 PE의 사용자가 작업을 종료하고자 한 시점에서 실제 완전한 종료가 이루어지는데 걸리는 시간.
- $T_{playgap(p)}$: PE p와 ES에 있어서의 재생 시간에 대한 시간 차이. 즉, 동일한 타임스탬프를 가진 프레임이 ES에서 재생될 때와 PE p에 의해 재생될 때의 시간 차이.
- $T_{relocate}$: 임의의 PE가 자신의 부모 PE의 이탈에 의해 PS로부터 새로운 부모 PE가 할당된 후, 새로운 부모 PE로 접속을 시작한 때부터 첫 번째 프레임을 성공적으로 받을 때까지의 시간.

3.3.2 스트리밍 및 릴레이의 시간 모델

(그림 6)은 (그림 3)의 모델에서 스트리밍이 이루어지는 동안의 시간 모델을 보여주고 있다.

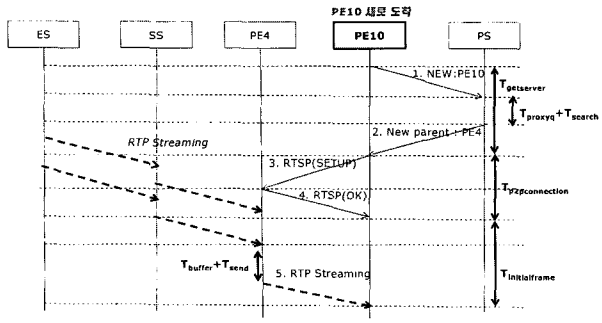
ES는 오디오 또는 비디오 데이터를 인코딩하고 RTP 패킷을 생성하여 SS로 전달한다. SS는 RTP 패킷을 버퍼링한 후, 자식 피어 PE1에게 이를 전달한다. 다시 PE1은 RTP 패킷을 수신하여 버퍼링 한 후, 자식 피어 PE5와 PE8에게 각각 릴레이한다. PE8의 $T_{playgap(p8)}$은 ES로부터 중간 PE들의 프로세싱 시간(버퍼링, 릴레이)과 네트워크 전달 지연 시간 등의 의해 누적된 시간의 합이 된다.



(그림 6) 스트리밍 및 릴레이

3.3.3 피어 참여의 시간 모델

(그림 7)은 (그림 4)의 모델에서 피어 참여에 따른 시간 흐름을 보여주고 있다.



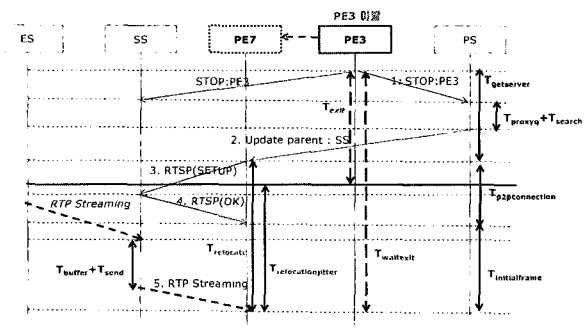
(그림 7) 피어 참여

새로운 피어 PE10은 PS로 접속하여 부모(PE3)를 할당받고 이로 접속하여 스트리밍 서비스를 시작한다. PE10의 연결 지연 시간은 $T_{getserver\ (p10)} + T_{p2pconnection\ (p10)} + T_{initialframe\ (p10)}$와 같다. 이 시간은 새로 도착하는 PE의 사용자가 감당하여야하는 지연 시간이며, P2P 시스템의 복잡도 및 성능과 관련 있다.

3.3.4 피어 이탈 및 재배치의 시간 모델

(그림 8)은 (그림 5)의 모델에서 피어 이탈에 대한 시간 모델을 보여주고 있다. 예를 들어 PE3이 P2P 네트워크에서 이탈 할 경우 자식 피어인 PE7은 PE3의 이탈로 인해 일시적으로 스트림 수신이 중단된다. PE7은 PS로부터 새로운 부모를 할당받고 다시 재접속하여 스트리밍 서비스를 시작한다.

피어 이탈과 재배치에 있어서 패킷 손실로 인한 지터가 발생할 소지가 다분하다. 여기서 새로운 시간 변수인 $T_{relocationjitter}$을 정의한다. $T_{relocationjitter}$은 부모 PE의 종료로 인해 더 이상 프레임을 전달받지 못해 생기는 시간 간격이며 그림에 표시하였다. 그림 중간의 가로 실선은 PE3이 완전히 종료한 시점으로 표시한다. 따라서 PE7의 $T_{relocationjitter}$을 0으로 만들기 위해서는 PE3의 T_{exit}가 $T_{waitexit}$ 이상이면 된다. 여기서 $T_{waitexit}$은 시스템에 설정하는 외부 상수 인자로 주어지며 사용자 혹은 프로그램이 기다릴 수 있는 최소 시간이다. $T_{waitexit}$와 지터 발생의 상관관계를 분석하여 $T_{waitexit}$을 조절하면 지터를 줄이거나 제거 할 수 있다.



(그림 8) 피어 이탈 및 재배치

3.4 성능 모델

P2P 스트리밍 시스템의 성능 지수를 다음과 같이 정의한다.

3.4.1 연결 지연 시간(P_{delay})

P_{delay}는 피어가 시스템 내에 도착하는 시점부터 첫 번째 프레임을 받는 때까지의 걸리는 시간으로, 사용자가 스트리밍 서비스를 받기 위해 기다리는 초기 연결 지연 시간을 의미한다. 이 시간은 $T_{connection}$과 동일한 값으로서, 이는 P2P 네트워크의 과부하 정도에 대한 지수로 이용할 수 있으며 사용자의 만족도를 평가하는 지수로 이용될 수 있다.

3.4.2 재생 시간 갭($P_{playgap}$)

$P_{playgap}$은 동일한 프레임이 ES에서 재생 될 때와 임의의 PE에서 재생되는 시간 사이의 차이 값으로 $T_{playgap}$과 동일하다. 이 시간은 패킷들이 ES로부터 임의의 PE까지 전송되는 동안 중간 PE들의 처리 시간(버퍼링, 릴레이)과 네트워크 전송 지연으로 인해 발생하게 된다. 사용자가 증가할수록 이 시간이 커져서 서비스의 질이 떨어질 수 있으며 동일한 조건하에서 버퍼링 알고리즘이나 PS의 배치 알고리즘의 효율성에 따라 달라질 수 있는 성능 요소이다.

3.4.3 지터율($P_{jitterrate}$)

$P_{jitterrate}$은 스트리밍 시간동안 총 패킷 중 손실 패킷에 의한 지터의 비율로서, $T_{relocationjitter}$동안 주로 발생한다. 패킷 손실은 P2P 시스템에서 PE들의 자유로운 이탈로 인하여, 스트리밍의 일시적 중단, 혹은 네트워크 트래픽의 급격한 변화로 인해 발생한다. 이로 인해 비디오 또는 오디오의 재생 질이 떨어지게 된다.

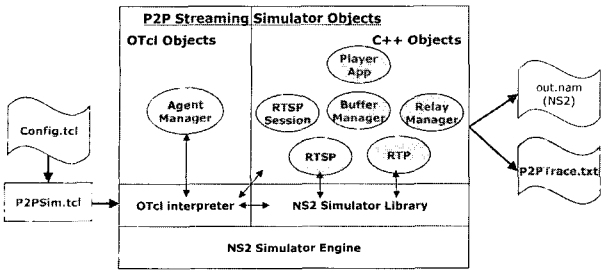
4. P2PStreamSim : P2P 스트리밍 시스템 시뮬레이터

4.1 시뮬레이터 구조

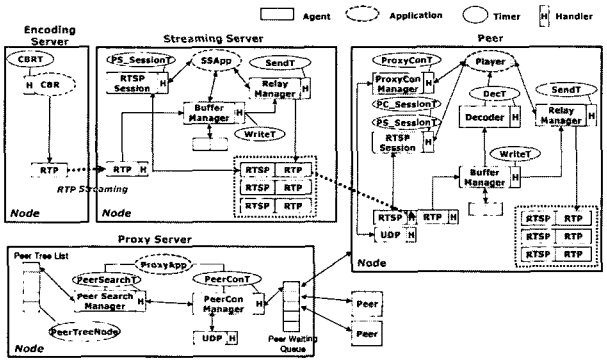
본 논문에서 구현한 *P2PStreamSim*의 시뮬레이션 모델은 3장에서 정의한 구조, 행동, 시간 모델을 기반으로 하며 구조는 (그림 9)와 같다.

시뮬레이션의 실행은 본 논문에서 작성한 P2PSim.tcl 코드에서 시작되며, 이는 시뮬레이션 파라미터를 설정하는 Config.tcl 파일을 입력으로 받는다. *P2PStreamSim* 객체들은 NS2 라이브러리를 이용하여 C++로 작성하였다. 시뮬레이션 결과는 out.nam과 P2PTrace.txt 파일로 저장한다. out.nam은 NS2에서 지원하는 모든 노드에 대한 트레이스를 저장한 결과이며, P2PTrace.txt는 성능 평가를 위해 성능 지수 값들을 따로 저장하는 파일이다.

현재 NS2에서는 P2P와 같은 노드들이 동적으로 생성, 삭제가 자유롭게 일어나는 네트워크 토폴로지를 시뮬레이션 하기에 다소 부적합한 구조로 되어있다. 따라서 동적인 피어의 도착과 이탈을 시뮬레이션 하기 위해 노드 간 링크는 메쉬 구조로 구성하고, 동적인 연결을 위해 RTSP와 RTP 에이전트를 런타임 중에 생성하여 연결하도록 구현하였다.



(그림 9) P2PStreamSim 구조



(그림 10) P2PStreamSim 내부 설계

(그림 9)에서 AgentManager는 동적으로 RTSP와 RTP를 생성하여 노드에 attach 하거나 detach 하기 위한 OTcl 클래스로, 동적인 피어의 참여와 이탈을 시뮬레이션 하기 위해 필요하다.

ES, SS, PE, PS는 NS2에서 정의한 노드(Node)를 상속받아 구현하였으며, RTSP, RTP와 같은 전송 계층 프로토콜과 스트리밍 서비스를 위한 모듈들은 NS2가 지원하는 에이전트(Agent) 타입으로 구현하였다. 또한 각 에이전트는 수행 시간의 흐름을 나타내기 위한 타이머를 각각 가지고 있다. (그림 10)은 *P2PStreamSim*의 전체 내부 구조이다.

4.2 P2PStreamSim 노드 구현

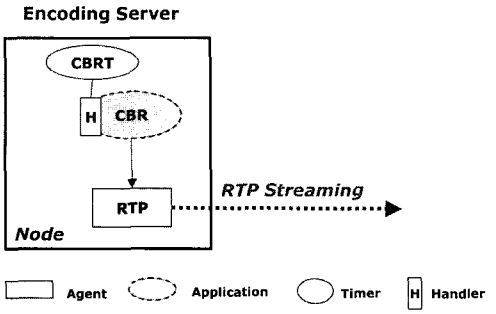
4.2.1 ES(Encoding Server) 구현

ES는 비디오 또는 오디오 스트림을 CBR(Constant Bit Rate) 또는 VBR(Variable Bit Rate) 형태로 압축하여 이를 스트리밍 서버에게 전송하는 서버로서 (그림 11)과 같은 구조를 가진다.

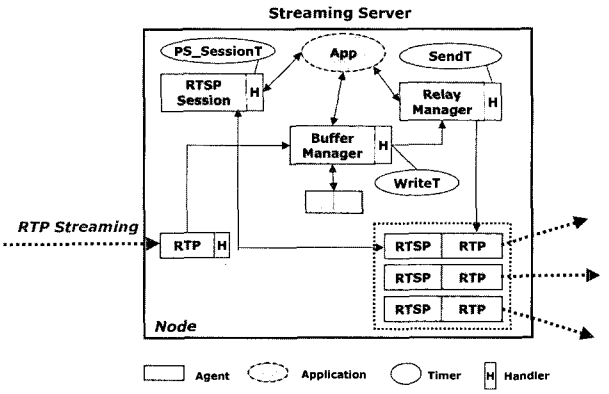
ES는 CBR 또는 VBR 형태의 트래픽을 발생시키는 어플리케이션과 RTP 패킷을 전송하는 RTP 에이전트로 구성된다. CBR 어플리케이션은 타이머인 CBRT 가지고 있어 일정한 간격으로 RTP 데이터를 생성한다. RTP 에이전트는 어플리케이션이 생성한 RTP 데이터를 패킷으로 나누어 전송한다.

4.2.2 SS(Streaming Server) 구현

SS는 스트림을 분배하는 노드로서 부모 피어의 역할을 수행한다. ES로부터 수신한 RTP 패킷을 자신의 자식 PE들에게 전달한다.



(그림 11) ES 구조



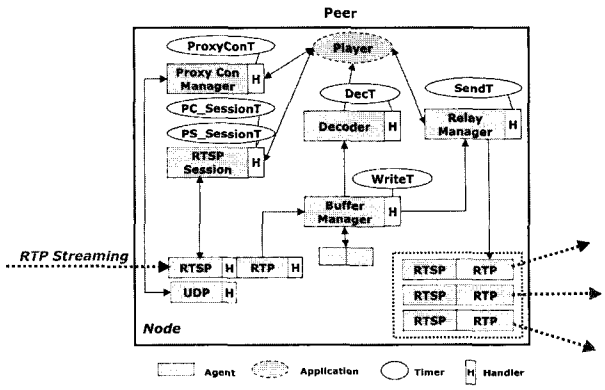
(그림 12) SS 구조

(그림 12)와 같이 SS는 RTSP Session, Buffer Manager, Relay Manager, RTSP, RTP 에이전트들로 구성되며, 각 에이전트는 실행할 시간의 흐름을 나타내기 위한 개별적 타이머를 가지고 있다. 임의의 PE가 RTSP 세션 연결을 요청할 경우 RTSP Session 에이전트는 응답 메시지를 생성하여 RTSP 에이전트로 보낸다. 세션 설정 후, Buffer Manager는 RTP 에이전트로부터 수신한 RTP 패킷을 버퍼에 저장한다. Relay Manager 에이전트는 버퍼에서 한 패킷 씩 읽어 RTP 에이전트를 통해 자식 PE 들에게 전송한다.

- RTSP Session 에이전트: RTSP 세션을 설정하기 위해 요청 또는 응답 메시지를 생성하고 전송하는 에이전트이다. RTSP Session은 피어의 RTSP 요청에 대한 응답 소요 시간을 소모하기 위해 PS_SessionT를 생성하여 사용한다.
- Buffer Manager 에이전트: 수신한 RTP 패킷을 버퍼에 저장하기 위한 에이전트로서 더블 버퍼링을 사용한다.
- Relay Manager 에이전트: 버퍼에 저장된 RTP 패킷을 하나씩 읽어 연결된 자식 PE 들에게 전송하는 에이전트로서 버퍼에서 한 패킷을 읽어 하나의 피어에게 전송하는 시간을 소요하는 타이머인 SendT를 가지고 있다.

4.2.3 PE(Peers) 구현

PE의 구조는 SS와 유사하며 SS 구조에 Proxy Con Manager와 Decoder 에이전트가 추가된 구조로서 (그림 13)



(그림 13) PE 구조

과 같다. PE는 Proxy Con Manager, RTSP Session, Buffer Manager, Decoder, Relay Manager, RTSP, RTP, UDP 에이전트로 구성되며, 각 에이전트는 시간의 흐름을 나타내기 위한 자신의 타이머를 가지고 있다. 여기서 UDP 에이전트는 NS2에서 지원하는 UDP 에이전트를 그대로 이용하였다.

임의의 PE가 네트워크에 참여할 때, Proxy Con Manager는 요청 메시지를 생성하여 UDP를 통해 PS에게 새로운 부모 피어를 요청한다. PE는 PS로부터 할당받은 부모 피어로 RTSP Session 에이전트를 이용하여 요청 메시지를 생성하고 RTSP 에이전트를 통해 부모 피어로 전송한다. 부모 피어와 세션을 설정한 후, Buffer Manager는 RTP 에이전트로부터 수신한 데이터를 버퍼에 저장한다. Decoder 에이전트는 버퍼에서 한 프레임씩 읽어 디코딩한 후 재생한다. 연결된 자식 피어가 있을 경우 Relay Manager 에이전트는 버퍼에서 한 패킷씩 읽어 RTP를 통해 연결된 피어에게 데이터를 전송한다.

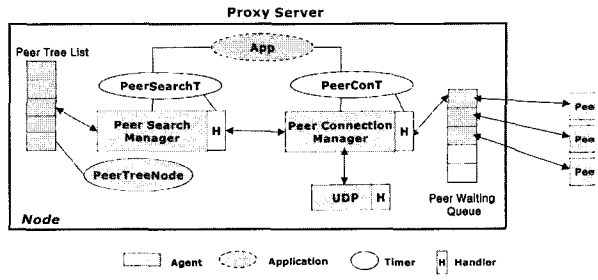
- Proxy Con Manager : PE가 네트워크에 참여 또는 이탈시, PS에게 자신의 상황을 알리기 위한 에이전트로 요청 메시지를 생성하는 에이전트이다.
- Decoder : 수신한 패킷을 한 프레임 단위로 디코딩하여 재생하는 에이전트로 DecT를 사용하여 디코딩 시간을 소요한다.

4.2.4 PS(Proxy Server) 구현

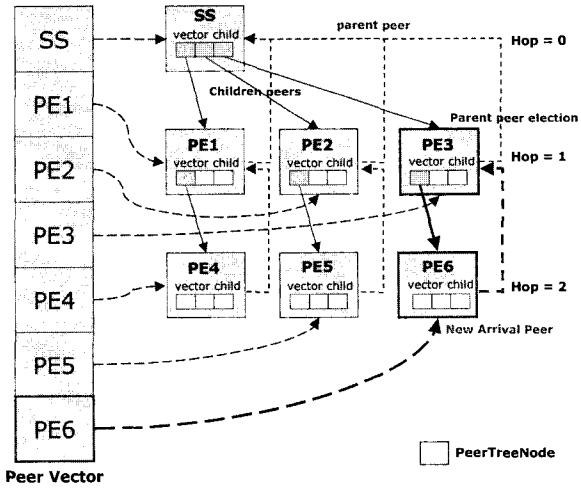
PS는 P2P 네트워크에 존재하는 피어에 대한 정보를 관리하는 서버로서, PE가 네트워크에 참여하거나 이탈 시 부모 피어를 탐색하여 PE에게 알려주며 동적으로 네트워크 토폴로지를 재구성한다.

PS는 (그림 14)와 같이 Peer Connection Manager와 Peer Search Manager, UDP 에이전트로 구성된다.

PE가 부모 피어의 탐색을 요청하였을 경우 PE는 PS의 Peer Waiting Queue에서 대기하게 된다. Peer Search Manager는 Peer Tree List에서 적당한 부모 피어를 선택한 결과, 부모 피어 IP를 Peer Connection Manager에게 알려주며, Peer Connection Manager는 응답 메시지에 부모 피어 IP를 삽입하여 큐에 대기하고 있는 PE에게 알려준다.



(그림 14) PS 구조



(그림 15) Peer Tree List 구조

〈표 1〉 PeerTreeNode 객체 필드

필드	설명
PeerID	피어를 식별하기 위한 ID
ParentPeerID	부모 피어 ID
PeerTreeNode *parent	부모 피어에 대한 포인터
vector<PeerTreeNode *> child	자식 피어들에 대한 벡터
childPeerCount	자식 피어의 수
hopCount	스트리밍 서버를 기준으로 한 자신의 홉 수

- Peer Connection Manager : PE의 참여 또는 이탈에 대한 요청 메시지를 처리하기 위한 에이전트로서, 응답 메시지를 생성하는 시간을 소요하기 위한 PeerConT를 가지고 있다.
- Peer Search Manager : PE의 부모 피어를 탐색하는 에이전트로서 탐색 시간을 소요하는 PeerSearchT를 가지고 있다. Peer Search Manager는 현재 네트워크에 참여한 PE에 대한 정보(PeerTreeNode)를 Peer Tree List에 트리 구조로 유지하며 (그림 15)와 같고 PeerTreeNode는 <표 1>과 같은 정보를 유지한다.

(그림 15)와 같이 Peer Tree List는 홉 수에 따른 순차적

인 검색을 하기 위해 현재 네트워크에 참여한 모든 PE에 대한 정보(PeerTreeNode)를 일차원 벡터로 유지한다. 각 PE는 자식 PE들에 대한 벡터와 부모 PE에 대한 포인터를 유지한다. 예를 들어 새로운 PE이 네트워크에 참여할 시, PS는 PeerTreeNode 중 부모 PE가 가능한 PE들 중에서 홉 수가 작고, 연결된 자식의 개수가 적은 PE3을 부모 PE로 선정한다. 여기서 홉 수란 SS를 기준으로 한 논리적인 홉 수를 의미한다. 그 후 Peer Tree List에 새로운 피어를 추가하여 Peer Tree List를 갱신한다.

4.3 시뮬레이션 파라미터

*P2PStreamSim*은 P2P 시뮬레이션 시스템 모델과 작업 부하는 위한 파라미터로 분리하여 정의하였으며 구체적으로 이들은 <표 2, 3>과 같다.

〈표 2〉 P2P 스트리밍 시스템 모델과 관련된 파라미터

요소	파라미터	설명
PE	bufferCount	버퍼 수
	bufferAccessDelay	버퍼 읽기 또는 쓰기 시간
	RTSPStackTime	RTSP 요청 또는 응답 시간
	decodingTime	프레임당 디코딩 시간
PS	protocolStackTime	PS 요청/응답 시간
	algorithmTime	부모 피어의 탐색 시간

〈표 3〉 작업 부하

요소	파라미터	설명
네트워크 환경	bandwidth	네트워크 대역폭
	transmissionDelay	네트워크 전송지연시간
ES	streamingTime	총 스트리밍 시간
	encodingType	인코딩 데이터 타입(CBR/VBR)
	dataType	오디오 / 비디오
	bitRate	비트율
	frameRate	프레임율
	VBR_variationRate	비트율에 대한 VBR의 비율
	VBR_duration	초당 VBR의 연속 수
PE	arrivalPeerCount	참여한 PE의 총 수
	meanInterarrivalTime	PE의 평균 도착 시간간격
	interarrivalDistribution	PE의 도착 분포
	exitPeerPercentage	PE 이탈 비율
	meanExitIntervalTime	PE의 평균 이탈 간격
	exitIntervalDistribution	PE의 이탈 분포
	exitStartTime	PE의 이탈 시작 시점
	capability	PE가 가질 수 있는 최 대 자식 피어 수
	waitExit	PE가 이탈하고자 한 시점에서 기다릴 수 있는 최소 대기 시간

5. 테스트 시뮬레이션

테스트 P2P 스트리밍 시스템을 정의하고 이를 사례로 적용하여 *P2PStreamSim*의 동작을 검증하고 성능 평가를 수행한다.

5.1 테스트 P2P 스트리밍 시스템 모델 및 성능 지수

테스트 P2P 스트리밍 시스템은 한 대의 ES, SS, PS와 n (현재는 100)개의 PE로 구성되며 성능 지수는 연결 지연 시간(P_{delay}), 재생 시간 갭($P_{playgap}$), 지터율($P_{jitterrate}$)로 설정한다.

5.2 시뮬레이션 실행

시뮬레이션을 위해 <표 4, 5>와 같이 파라미터를 설정하였다. PE ID는 피어를 구별하는 식별자이고, P2P 네트워크에 참여하는 순서대로 ID가 하나씩 증가한다. 100개의 PE가 평균 1초 간격으로 지수 분포로 도착한다. ES의 bitRate은 Ateme 사의 H.264/AAC Kompressor 하드웨어 인코더[14]를 기준으로 해상도가 720×480일 때 실제 측정한 값으로 설정하였다.

〈표 4〉 P2P 스트리밍 시스템 모델 파라미터

요소	파라미터	값
PE	bufferCount	2
	bufferAccessDelay	1us
	RTSPStackTime	1ms
	decodingTime	1ms
PS	protocolStackTime	1ms
	algorithmTime	1ms

〈표 5〉 작업 부하

요소	파라미터	값
네트워크 환경	bandwidth	100Mbps
	transmissionDelay	10ms
ES	streamingTime	500s
	encodingType	CBR
	dataType	Video
	bitRate	71Kbps
	frameRate	30fps
PE	arrivalPeerCount	100
	meanInterarrivalTime	1.0s
	interarrivalDistribution	Exponential
	exitPeerPercentage	30%
	meanExitIntervalTime	1.0s
	exitIntervalDistribution	Exponential
	exitStartTime	200s
	capability	5
	waitExit	0ms, 30ms, 70ms

모든 PE 들은 시뮬레이션 초기에 모두 PS와 양방향 링크로 설정되며, 또한 모든 PE 들이 메쉬 구조로 양방향 링크를 상호 설정한다. 실험 결과는 총 20번의 시뮬레이션 실행 결과의 평균값이다.

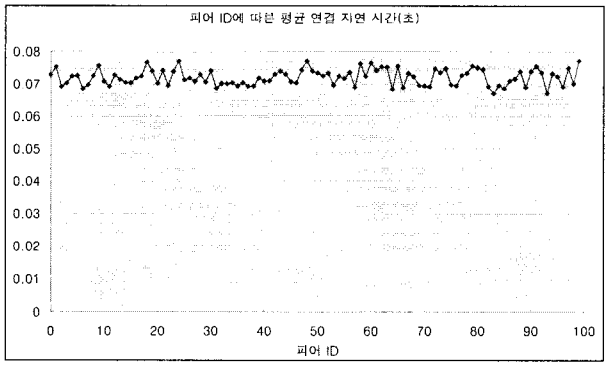
5.3 시뮬레이션 결과

5.3.1 실험 결과 1 : 피어 참여에 따른 실험 결과

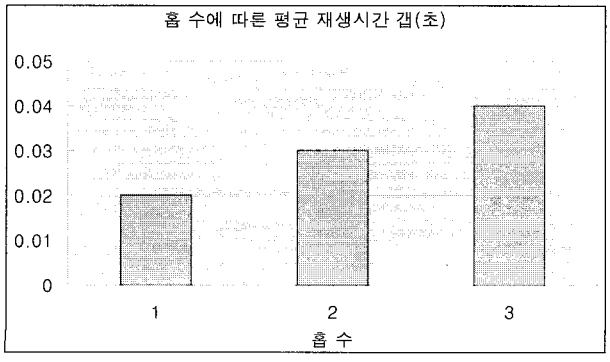
PE ID 별로 평균 연결 지연 시간(P_{delay})을 측정한 결과는 (그림 16)과 같다. 실험 결과 총 100개의 피어에 대한 평균 연결 지연시간은 약 70ms 소요됨을 알 수 있다. P_{delay}는 $T_{getserver} + T_{p2pconnection} + T_{initialframe}$과 동일하며 여기서 $T_{getserver}$ PE가 PS와 자신의 피어 서버와 통신하는 시간으로서 네트워크 지연 시간에 가장 영향을 많이 받는 시간이며 대략 40ms 가 조금 넘는 시간이 될 것이다. 인코딩 서버의 프레임율이 30fps이므로 프레임 간 지연 시간은 약 33ms 정도이다. 각 PE 들이 자신의 피어 서버에 접속한 후 첫 번째 프레임을 받기 위해서 소요되는 시간인 $T_{initialframe}$은 네트워크 지연 시간 10ms에다 평균 16.5ms (=33ms/2)의 첫 번째 프레임 전송 지연 시간을 합치고 기타 소요되는 시간을 예측하면 약 70ms의 평균 지연 시간이 예측된다.

(그림 17)은 홉수에 따른 평균 재생 시간 갭($P_{playgap}$)을 측정한 결과이며, 테스트 P2P 시스템에서 PE들의 최대 홉수는 3이다.

실험 결과 홉 수가 증가함에 따라 피어의 재생 시간 갭이 증가함을 알 수 있다. 네트워크에 참여한 피어의 가용 능력, 즉 연결 가능한 자식 피어의 수가 모두 동일하며 피어의 프로세싱 시간(버퍼링, 릴레이)이 매우 짧기 때문에 재생 시간 갭에 영향을 미치지 않는다. 따라서 홉 수에 따른 평균 재생 시간 갭은 네트워크 전송 지연 시간에 비례하여 증가함을 알 수 있다. 즉 SS에 직접 연결된 홉수 1인 PE들은 원 소스와 0.02ms의 시간차를 보이며 홉수 3인 PE들은 0.04초의 시간 차이 밖에 나지 않기 때문에 각 PE의 사용자들은 거의 시간 차이를 인식하지 않는다고 볼 수 있다. 이런 결과는 비교적 통신 속도가 빠르기 때문이며 통신 속도가 느린 네트워크에서는 이 시간이 커질 가능성이 높다.



(그림 16) 피어 ID에 따른 평균 연결 지연 시간(P_{delay})



(그림 17) 홉 수에 따른 평균 재생 시간 갭($P_{playgap}$)

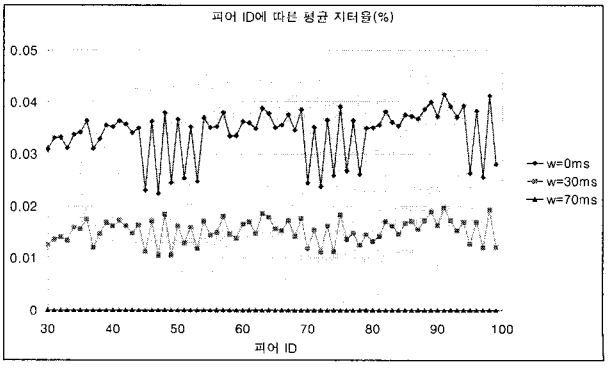
5.3.2 실험 결과 2 : 피어 이탈에 따른 실험 결과

지터의 발생은 부모 피어의 이탈로 인해 자식 피어들의 스트리밍 서비스가 일시적으로 중단되기 때문이다. 즉, 자식 피어들은 새로운 부모 피어로 접속하여 첫 번째 프레임을 수신하기 전까지 패킷 손실로 인한 지터가 발생하게 된다.

임의의 피어가 스트리밍 서비스를 중단하고 이탈할 경우 이탈하고자 한 시점에서 실제 사용자가 완전한 종료가 이루어질 때까지 기다릴 수 있다고 정의한 최대 대기 시간을 $T_{waitexit}$로 정의하며, 이는 시스템에 설정하는 외부 상수이다. 따라서 이탈하는 피어가 자식 피어들이 다시 재배치되어 스트리밍이 정상적으로 이루어질 때까지 $T_{waitexit}$을 유지한다면 지터를 줄일 수 있다.

총 500초의 스트리밍 시간동안 100개의 피어 중 30%의 피어가 평균 1초 간격으로 이탈할 경우를 시뮬레이션 하였다. (그림 18)은 $T_{waitexit}$시간이 0ms부터 70ms까지 증가할 경우 PE ID에 대한 평균 지터율($P_{jitterrate}$)을 측정한 결과이다.

실험 결과 $T_{waitexit}$가 증가함에 따라 PE ID에 따른 평균 지터율은 감소하며, $T_{waitexit}$가 70ms일 때는 지터가 발생하지 않는다. 이것은 피어의 평균 연결 지연 시간이 약 70ms이므로 이 시간만큼 이탈 피어가 $T_{waitexit}$을 유지한다면 자식 피어들은 스트림 끊김 현상 없이 원활한 서비스를 받을 수 있다. 즉, 사용자나 피어에 설치된 프로그램이 종료해야 한다고 결정하고 기다리는 시간이 약 70m로서, 사용자 측면에서 아주 짧은 시간이므로 이 정도의 시간은 참을 수 있다고 판단된다.



(그림 18) 피어 ID에 따른 평균 지터율($P_{jitterrate}$)

6. 결 론

기존 인터넷 스트리밍 시스템은 클라이언트-서버 형태의 중앙 구조로 서버에 대한 트래픽 집중으로 인한 병목 현상이 발생하며, 한 스트리밍 서버의 가용 능력에 따라 스트리밍 단말기의 개수가 제한되는 단점이 있다. 이러한 한계점을 극복하기 위해 인터넷 스트리밍 시스템에 P2P 네트워크를 이용하는 구조에 대한 연구들이 전 세계적으로 진행되고 있다.

그러나 P2P 기반 스트리밍 시스템은 설계 및 연구 개발에 있어 많은 컴퓨터들이 필요하며 성능 평가 시 많은 시간과 경비가 소모된다. 따라서 P2P 스트리밍 시스템을 설계 개발하기에 앞서 설계의 정확성을 검증하고 다양한 네트워크 토폴로지를 구성하여 실험하기 위한 시뮬레이터의 개발이 절실히 요구된다.

본 논문에서는 P2P 스트리밍 시스템에 대한 모델과 성능 지수를 정의하고 네트워크 기본 프로토콜과 체계를 제공하는 NS2라는 시뮬레이션 툴을 이용하여 P2P 스트리밍 시뮬레이터 *P2PStreamSim*을 설계 및 구현하였다. 또한 테스트 P2P 스트리밍 시스템에 대한 성능 평가를 수행하였다.

결론적으로 본 논문 연구는 P2P 스트리밍 시스템을 개발하기 전 단계에서 설계의 정확성을 검증하고 그 성능을 평가할 수 있는 도구를 제공한다는 면에서 큰 의미를 지닌다. 향후 P2P 스트리밍 시스템에 대한 실제 파라미터 값들을 적용한 실험을 수행하며 다양한 P2P 스트리밍 시스템 모델에 대한 성능 평가를 수행할 수 있도록 *P2PStreamSim*을 확장 보완하고자 한다.

참 고 문 헌

[1] K. Kikuma, Y. Morita, and H. Sunage, "A Study of a P2P Community on a P2P Communication Platform," *Proceeding of ICCT*, April 2003

[2] Hye Sun Kim, Nam Yun Kim, and Kitae Hwang, "Performance Parameters for Network Accessibility in MPEG-4 FGS Video Streaming," *5th Asia Pacific International Symposium on Information Technology*, pp.11-14, Jan., 2006.

[3] Napster, http://opennap.sourceforge.net/napster.txt.

[4] M.Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella Network," *In Proceedings of IEEE 1st Int'l Conf. on Peer-to-Peer Computing*, 2001.

[5] D.A. Tran, K. A. Hua, and T. T. Do, "ZIGZAG : An Efficient Peer-to-Peer scheme for Media Streaming," *In Proceedings of IEEE INFOCOM 2003*, April, 2003

[6] PeerCast, http://www.peercast.org.

[7] X. Liao, H. Jin and Y. Liu, "AnySee : Peer-to-Peer Live streaming," *In Proc. of IEEE INFOCOM*, pages 372-379, 2006.

[8] T. Hama, K. Asatani and H. Nakazato, "P2P Live Streaming System with Low Signal Interruption," *In Proceedings of the 18th AINA IEEE*, pages 605-610, 2004.

[9] V.N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distribution Streaming Media Content using Cooperative Networking," *In Proceedings of ACM NOSSDAV'02*, May, 2002.

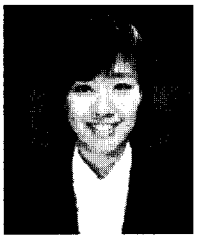
[10] NS2tutorial, http://www.isi.edu/nsnam/ns/tutorial/index.html.

[11] NAM, http://www.isi.edu/nsnam/nam/.

[12] RTP, H. Schulzrinne, S. Casner and R. Frederick, RFC 3550.

[13] RTSP, H. Schulzrinne, A. Rao and R. Lanphier, RFC 2326.

[14] H.264/AAC Kompressor, http://www.ateme.com/BB_kompressor AVC.php.



김 혜 선

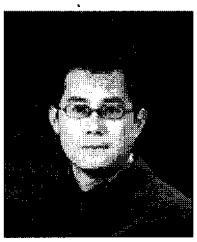
e-mail : hyesun.kim@konantech.com

2004년 한성대학교 컴퓨터공학과 (학사)

2007년 한성대학교 대학원 컴퓨터공학과 (공학석사)

2007년~현 재 (주)코난테크놀로지 연구원

관심분야 : 멀티미디어 스트리밍, 분산시스템, 유비쿼터스 시스템 등



황 기 태

e-mail : calafk@hansung.ac.kr

1986년 서울대학교 전자계산 학과 (학사)

1988년 서울대학교 대학원 전자계산학과 (공학석사)

1994년 서울대학교 대학원 전자계산학과 (공학박사)

1994년~현 재 한성대학교 컴퓨터공학과 교수

관심분야 : 분산 시스템, 모바일.유비쿼터스 시스템, 멀티미디어 스트리밍 등