P2P 구조에 기반한 인터넷 방송 시스템 설계 및 구현

정회원 우 문 섭*, 김 남 윤**, 황 기 태***

Design and Implementation of Internet Broadcasting System based on P2P Architecture

Moonsup Woo*, Namyun Kim**, Kitae Hwang*** ***Regular Members***

요 약

클라이언트-서버 구조의 스트리밍 시스템은 서버의 가용 능력에 따라 클라이언트의 개수가 제한되는 단점을 가지고 있다. 본 논문에서는 인터넷 방송 시스템의 확장성과 안정성을 지원하기 위해 P2P에 기반한 모델을 제시한 후, 프로토타입 시스템을 구현한 사례를 소개한다. 본 논문에서 구현한 시스템인 OmniCast264는 실시간으로 H.264 비디오 스트림을 제공하는 인코딩 서버, 스트림을 네트워크에 분배하는 스트리밍 서버, 비디오의 재생 및 분배를 담당하는 피어 노드, 그리고 P2P 네트워크에 노드를 동적으로 배치시키는 프록시 서버로 구성된다. P2P에 기반한 OmniCast264은 스트리밍 부하의 분산화, 실시간 재생, 에러 발생에 따른 강건함, 계층의 모듈화 등의 개념을 가지고 있기 때문에 대용량의 인터넷 방송에 적합하다고 할 수 있다. 마지막으로 12 대의 PC들을 병렬 및 직렬 구조의 P2P로 구성한 후, OmniCast264의 성능을 평가하여 실시간 재생이 가능함을 검증하였다.

Key Words : P2P architecture, Live streaming, Real-time, Internet broadcasting system

ABSTRACT

IStreaming services with a client-server architecture have scalability problem because a server cannot accomodate clients more than its processing capability. This paper introduces a case study for implementing H.264 streaming system based on P2P architecture in order to provide scalable and stable broadcast streaming services over the internet. The prototype system called OmniCast264 consists of the H.264 encoding server, the streaming server, the proxy server, and peer nodes. The proxy server dynamically manages placement of the peer nodes on the P2P network. OmniCast264 has the concepts of distributed streaming loads, real-time playback, error-robustness and modularity. Thus, it can provide large-scale broadcast streaming services. Finally, we have built P2P streaming systems with 12 PCs connected serially or in parallel. The experiment shows that OmniCast264 can provide real-time playback.

Ⅰ. 서 론

유·무선 인터넷의 확산과 멀티미디어 처리 능력의 발전으로 VoD, IPTV와 같은 멀티미디어 스트리밍 연구가 활성화되었다[1,2]. 서버는 멀티미디어 스트림을 전송하고 클라이언트는 수신과 함께 재생하는 구조를 취하고 있다. 이러한 클라이언트-서버 구조의 스트리밍 시스템은 서버의 가용 능력에 따라 클라이언트 수가 제한되는 확장성 문제를 가지고 있다.

※ 본 연구는 2007학년도 한성대학교 교내 연구비 지원 과제임.
* 프로자이너 (teddywoo@prosigner.com), ** 한성대학교 정보시스템 공학과, *** 한성대학교 컴퓨터시스템 공학과
논문번호 : KICS2007-05-220, 접수일자 : 2007년 5월 22일, 최종논문접수일자 : 2007년 11월 16일

클라이언트-서버 구조의 문제점을 해결하기 위해서 P2P(peer to peer) 분산 아키텍처가 최근 주목을 받고 있다[3,4]. P2P 시스템은 분산 시스템의 한 종류로서 독립된 피어 노드(이하 노드) 사이에서 자원을 공유하는 시스템이다. 여기서 노드란 서버와 클라이언트의 기능을 동시에 가진 단말기이다. 이러한 P2P 시스템은 노드간에 컨텐츠를 공유함으로써 확장성이 중앙 집중 방식에 비하여 매우 높다.

최근 수년 동안 P2P에 대한 연구가 활발하게 진행되어 왔다. NICE[5]와 ZIGZAG[6]는 P2P의 네트워크 토폴로지를 트리 구조로 구성하여 노드 이탈시 하위 노드들의 스트림 끊김 현상을 최소화하려는 연구를 수행하였으며, Coolstreaming[7]은 네트워크 토폴로지를 메쉬(Mesh) 구조로 구성하여 트리 구조보다 좀 더 안정적 네트워크를 구축하고자 연구하였다. GnuStream[8]은 Gnutella[9] 프로토콜을 기반으로 하여 시스템을 구현한 사례들이다. 그러나 기존 연구들은 주로 토폴로지 구성을 바탕으로 노드의 참여와 이탈에 따른 프로토콜에 주안점을 두었으며 많은 경우 시뮬레이션에 의존한 연구들이다. P2P 스트리밍 시스템은 네트워크 토폴로지를 구성하는 방식, 응용의 요구 조건, 멀티미디어 데이터 포맷 등에 따라서 시스템의 구조가 달라진다. 따라서 대용량 인터넷 방송 시스템을 지원하기 위해서는 응용에 적합한 P2P 시스템 모델을 정의하고 구성 요소의 기능 및 구조에 대한 구체적인 제안이 필요하다.

본 논문에서는 인터넷 방송 시스템의 요구 조건을 (1) P2P 노드의 부하 분산화, (2) 실시간 재생, (3) 에러 발생에 따른 강건함, (4) 모듈의 계층화로 분류하고 혼합 P2P 토폴로지와 H.264 압축 기술에 기반한 시스템의 구조를 설계하였다. 그리고 프로토타입 시스템 OmniCast264를 구축한 후, 실험을 통해 시스템을 검증하였다. 본 논문은 인터넷 환경에서 확장가능한 스트리밍 시스템을 설계 구현한 사례를 소개함으로써 향후 대용량 스트리밍 시스템 개발의 초석을 제공하는데 목적이 있다.

본 논문의 구성은 다음과 같다. Ⅱ절에서는 인터넷 방송에 적합한 P2P 스트리밍 시스템 모델에 대해 설명한다. Ⅲ절에서는 P2P 기반 H.264 인터넷 방송 시스템 OmniCast264의 설계 및 구현 사항을 기술한다. 그리고 Ⅳ절에서는 실험을 통해 OmniCast264의 성능을 분석한다. 마지막으로 Ⅴ절에서는 결론 및 향후 연구 과제에 대해 기술한다.

Ⅱ. P2P 구조의 방송 스트리밍 시스템 모델

P2P 시스템은 디렉토리 서버의 존재 유·무에 따라 순수 P2P(Pure P2P), 혼합 P2P(Hybrid P2P), 수퍼 P2P(Super P2P) 방식으로 나뉜다[10]. 디렉토리 서버는 노드들의 위치와 컨텐츠 정보를 메타 데이터로 유지한다. 본 논문에서는 멀티미디어 스트림을 중계하던 노드의 이탈 시에 스트리밍 중단 시간을 최소화하기 위해 메타 정보를 중앙의 디렉토리 서버에서 관리하는 혼합 P2P 방식을 적용한다.

2.1 시스템 구성 요소

그림 1은 본 논문에서 정의한 P2P 스트리밍 시스템 모델로서 구성 요소는 다음과 같다.

- 인코딩 서버 : 라이브 비디오 소스를 실시간으로 압축하는 서버이다.
- 스트리밍 서버 : 멀티미디어 컨텐츠를 최초로 P2P 네트워크에 분배하는 서버이다.
- 프록시 서버 : 디렉토리 서버의 일종으로 네트워크 내의 메타 정보를(노드의 위치, 노드의 네트워크 대역폭, 컨텐츠 등) 유지하고 있는 서버로서 클라이언트의 참여(join)/이탈(leave)시 노드의 위치를 갱신한다.
- 노드 : 스트리밍 서버로부터 분배된 컨텐츠를 공유 및 재생하는 사용자 컴퓨터이다. 컨텐츠를 중계할 경우 '서버 노드'라고 정의하고 컨텐츠를 수신 시 '클라이언트 노드'라고 분류한다. 노드는 서버/클라이언트 역할을 동시에 수행할 수 있으며 단지 클라이언트 역할만을 수행할 수도 있다.

노드는 P2P 네트워크에 참여시, 프록시 서버에게 알린다. 프록시 서버는 토폴로지 알고리즘에 따라서 스트리밍이 가능한 서버 노드를 참여 노드에 알린

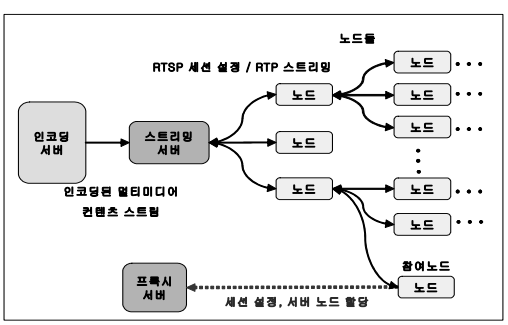

그림 1. P2P 스트리밍 시스템 모델

다. 참여 노드는 서버 노드와 RTSP 세션을 맺고 RTP를 통하여 방송 컨텐츠를 스트리밍 받는다.

2.2 개념적 모델

본 논문에서는 인터넷 환경에서 P2P 스트리밍을 위해 요구되는 사항을 파악하여 아래와 같은 개념적 모델을 정의하였다.

- 분산화 (Decentralization): 프록시 서버는 참여하는 노드들의 처리 능력을 기준으로 부하 균형(Load-Balancing)을 유지하도록 네트워크 토폴로지를 구성할 수 있어야 한다. 노드의 처리 능력이 높을수록 많은 노드에게 스트림을 중계할 수 있다. 따라서 노드는 자신의 처리 능력을 전달하여야 하고 프록시 서버는 효율적으로 부하를 분배할 수 있는 기능을 제공하여야 한다.
- 실시간 재생(Real-Time Playback): 노드에서 실시간 재생을 위해서는 서버 노드로부터 첫 프레임을 받기까지의 연결 지연 시간이 짧아야 하며, 인코딩 서버에서 전송한 재생율(초당 15~30 프레임)을 만족할 수 있도록 노드를 구성하여야 한다.
- 에러에 대한 강건함 (Error-Robustness): 노드의 예상치 못한 이탈은 전체 P2P 네트워크의 스트리밍 서비스를 중단시킨다. P2P 스트리밍 시스템은 예상치 못한 노드의 이탈 시에 서비스가 중단되는 시간을 최소화해야 한다. 이를 위해 노드는 프록시 서버에게 에러를 통보할 수 있는 기능을 제공해야 하며 서비스 복구 지연 시간이 작아야 한다.
- 계층의 모듈화(Modularity): 프록시 서버의 네트워크 토폴로지 알고리즘이나 인코더는 응용 분야에 따라 결정된다. 따라서 시스템에서는 모듈화를 통해 적응성과 확장성을 높일 수 있어야 한다.

Ⅲ. OmniCast264의 설계 및 구현

Ⅱ절에서 정의한 P2P 스트리밍 시스템 모델을 기반으로 OmniCast264를 설계 및 구현한 내용에 대해 기술한다.

3.1 인코딩 서버

인코딩 서버는 오디오/비디오 캡처 장치에서 수신한 멀티미디어 컨텐츠를 실시간으로 인코딩하여 스트리밍 서버에게 전송하는 장치이다. 본 논문에서는 실시간으로 인코딩하기 위해 하드웨어 오디오/비디오 인코더를 사용하였다. 하드웨어 인코더는 SD급 H.264를 실시간으로 30 fps에 해당하는 데이터를 인코딩한다.

인코딩 서버는 인코딩된 오디오/비디오 데이터를 RTP/UDP 패킷으로 스트리밍 서버에게 전송한다. 그리고 오디오/비디오에 대한 정보를 담은 SDP(Session Description Protocol) 화일은 인코딩 서버에서 생성되며, 스트리밍 서버로 복사된다.

3.2 스트리밍 서버

스트리밍 서버는 인코딩 서버로부터 수신한 오디오/비디오 방송 컨텐츠를 P2P 네트워크에 분배하는 장치로서, Apple사의 오픈 소스인 다윈 스트리밍 서버를 사용하여 구축하였다. 스트리밍 서버의 기능은 다음과 같다.

- 인코딩 서버로부터 실시간으로 들어오는 H.264 인코딩 패킷들을 버퍼링한다.
- RTSP를 지원하여 노드와 세션을 설정하고, 세션 정보를 담은 SDP를 노드에게 전송한다.
- RTP를 지원하여 노드에게 비디오/오디오 패킷들을 실시간으로 전송한다.

3.3 프록시 서버

프록시 서버는 P2P 네트워크에 참여하는 노드를 인증하고, 노드의 참여 및 이탈에 따라 네트워크 토폴로지를 관리하는 서버이다. 프록시 서버의 기본 구조는 그림 2와 같다. 프록시 서버를 구성하는 세 가지 계층에 대한 설명은 다음과 같다.

3.3.1 인터페이스 계층(Interface Layer)

이 계층은 본 논문에서 제안한 USCP(User Session Control Protocol)로 구성되며 분산화, 에러

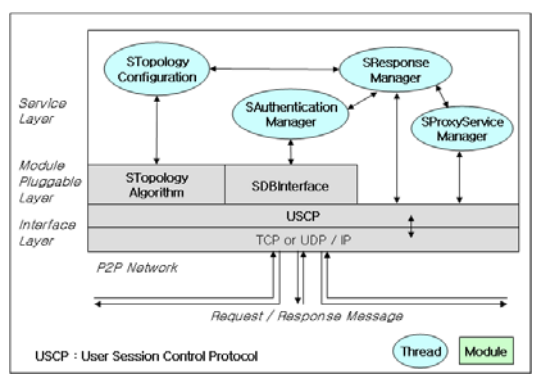

그림 2. 프록시 서버 소프트웨어 구조

표 1. USCP 요청 및 응답 메시지의 종류

메시지 종류	전송 방향	설명
Login	노드 → 프록시 서버	노드의 로그인 요청 메시지
Logout	노드 → 프록시 서버	노드의 로그아웃 요청 메시지
Subscription	노드 → 프록시 서버	노드의 가입 요청 메시지
ReConfiguration	노드 → 프록시 서버	서버 노드의 이탈시 새로운 서버를 할당받기 위한 요청 메시지
Node Information	노드 → 프록시 서버	노드의 서비스 능력을 프록시 서버에게 알리기 위한 메시지
ReStreaming Connection	프록시 서버→ 노드	Subscription, Login, ReConfiguration에 대한 응답 메시지, 서버 노드의 정보를 내포
OK	프록시 서버→ 노드	노드의 요청에 따른 정상적인 처리 응답
ERROR	프록시 서버→ 노드	노드의 요청에 대하여 처리가 불가능 응답, 에러의 내용을 담은 문자열 송신

에 대한 강건함을 제공하기 위해 다음과 같은 메시지들을 처리한다.

- 노드의 인증을 수행할 수 있는 메시지
- 노드로부터 서버 노드의 이탈을 보고 받을 수 있는 메시지
- 노드의 스트리밍 서비스 역량을 보고 받을 수 있는 메시지

USCP 모듈은 요청/응답 방식으로 동작하며, 표 1은 프록시 서버와 노드 사이에 전송되는 USCP 요청/응답 메시지의 종류이고 메시지들은 XML로 설계하였다.

3.3.2 모듈 연결형 계층(Module Pluggable Layer)

모듈 연결형 계층은 토폴로지 알고리즘(STopology Algorithm) 모듈과 데이터베이스 연결(SDBInterface) 모듈로 구성되어 있다.

토폴로지 알고리즘 모듈은 에러 발생에 따른 강건함을 지향하기 위해 다중 트리나 메쉬 구조 등의 P2P 네트워크 토폴로지를 구성하는 알고리즘 모듈을 제공하며 기본적으로 다음과 같은 기능을 지원한다.

- 노드 참여시 P2P 네트워크에 추가하는 기능
- 노드 이탈시 P2P 네트워크에서 노드를 제거하는 기능
- 노드에게 서버 노드를 할당하기 위한 알고리즘 수행 기능

데이터베이스 연결 모듈은 프록시 서버가 노드의 인증과 정보를 유지하기 위해 사용된다. 이 모듈은 기본적으로 다음과 같은 기능을 지원한다.

- 노드의 시스템 가입 시 저장소에 추가하는 기능
- 노드의 시스템 탈퇴 시 삭제하는 기능
- 노드의 인증 요청 시 저장소에서 노드를 검색하는 기능

3.3.3 서비스 계층(Service Layer)

서비스 계층은 인터페이스 계층과 모듈 연결형 계층을 기반으로 프록시 서버의 기능을 정의한다. 서비스 계층은 노드와 소켓 연결을 위한 연결 서비스 스레드(SProxyServiceManager), 노드와의 요청 및 응답 메시지 통신을 위한 응답 스레드(SResponseManager), 노드 인증을 위한 인증 스레드(SAuthentication Manager), 네트워크 토폴로지 구성을 위한 배치 스레드(STopologyConfiguration)로 구성된다.

① 연결 서비스 스레드

노드의 소켓 접속 요청을 처리하며 주요 기능은 다음과 같다.

- 노드의 소켓 연결 요청에 따른 수락
- 수락된 소켓 핸들을 응답 스레드로 전달

연결 서비스 스레드는 노드와 통신하기 위한 스레드로서 본 논문에서 제안한 XML 기반 USCP 모듈 계층을 이용한다.

② 응답 스레드

USCP 요청 메시지 및 응답 메시지 처리를 담당하며 주요 기능은 다음과 같다.

- 노드들의 세션 리스트 관리
- USCP 요청 메시지 분석을 통해 인증은 인증 스레드, 네트워크 토폴로지 구성은 배치 스레드에게 처리를 요청
- 각 스레드에서의 처리 결과를 USCP 응답 메시지로 작성하여 노드에게 전송

③ 배치 스레드

배치 스레드는 STopologyAlgorithm을 이용하여 P2P 네트워크 토폴로지를 구성한다.

STopologyAlgorithm은 여러 가지 네트워크 토폴로지 알고리즘 모듈을 적용할 수 있도록 추상 클래스로서 정의되었으며, 본 논문에서는 네트워크 토폴로지 구성 알고리즘의 하나인 "홉 수 기반 순차적 검색 알고리즘"을 적용한다. 즉, 홉 수가 작은 노드를 선정한 후, 서비스 가용 능력을 고려하여 선택한다. 배치 스레드는 네트워크 토폴로지를 동적으로 구성하고 그 결과를 응답 스레드에게 알린다.

④ 인증 스레드

노드를 인증하며 주요 기능은 다음과 같다.

- SDBInterface를 이용한 노드의 가입/탈퇴, 인증
- 처리 결과를 응답 스레드에게 전달

3.4 노드

노드는 프록시 서버에 접속하여 P2P 네트워크에 참여하고, 서버 노드로부터 멀티미디어 컨텐츠를 실시간으로 수신 및 재생하며, 이를 다른 클라이언트 노드에게 중계하기도 한다. 노드의 기본 구조는 그림 3과 같다. 노드의 주요 기능은 다음과 같다.

- 프록시 서버에 접속하여 인증 및 서버 노드를 할당받는다.
- 프록시 서버에게 자신의 방송 중계 능력을 측정하여 알린다.
- 서버 노드와 RTSP 세션을 맺고 RTP를 통하여 오디오/비디오 스트림을 수신한다.

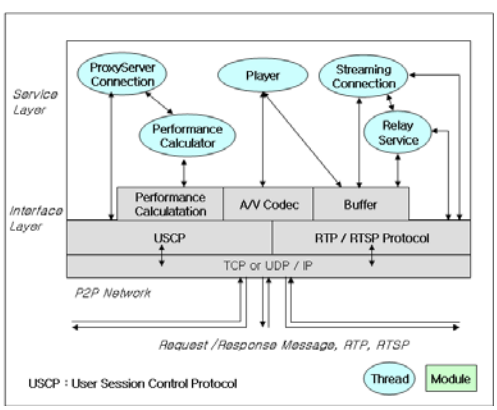

그림 3. 노드의 소프트웨어 구조

- 수신한 스트림을 버퍼링 및 재생한다.
- 클라이언트 노드의 RTSP 세션 연결 시에, 버퍼링한 스트림을 중계한다.

3.4.1 인터페이스 계층

인터페이스 계층은 프록시 서버와 통신하는 USCP 프로토콜과 멀티미디어 세션 설정 및 멀티미디어 스트리밍을 위한 RTSP/RTP로 구성되어 있다.

① USCP 클라이언트 모듈

프록시 서버에게 전송할 요청 메시지를 작성하고, 프록시 서버로부터 수신한 응답 메시지를 해석하는 기능을 제공한다. 이 모듈은 다음과 같은 메시지들을 지원한다.

- 프록시 서버와의 인증을 수행할 수 있는 메시지
- 서버 노드 이탈시 프록시 서버에게 보고 할 수 있는 메시지
- 프록시 서버에게 스트리밍 서비스 역량을 보고할 수 있는 메시지
- 프록시 서버로부터 서버 노드를 할당 받을 수 있는 메시지

② RTSP/RTP 모듈

RTSP는 스트리밍 서버와 클라이언트 사이에 세션 설정 및 제어를 수행하며, RTP는 스트림내의 순서화 및 동기화 기능을 수행한다. 본 논문에서는 표준 RTSP 버전 1.0을 기반으로 모듈을 구현하였으며, RTP 모듈은 오픈 소스인 JRTPLIB 3.6.0 버전을 수정하여 구현하였다[11].

3.4.2 모듈 연결형 계층

모듈 연결형 계층은 노드의 스트리밍 서비스 역량 측정 모듈(Performance Calculation)과 컨텐츠 스트림을 버퍼링하는 버퍼 관리 모듈(Buffer Manager) 그리고 컨텐츠 스트림을 디코딩하는 모듈(A/V CODEC)로 구성되어 있다.

스트리밍 서비스 역량 측정 모듈은 분산화 특성을 고려하여 프록시 서버가 네트워크 토폴로지를 안정적으로 유지할 수 있도록 노드의 역량을 측정한다. 즉, 노드의 네트워크 대역폭 가용율, 노드의 PC 성능(CPU, 메모리 등) 가용율 등을 측정하는 기능을 제공한다. 버퍼 관리 모듈은 서버 노드로부터 전송된 데이터를 저장한다. 음성 및 영상 코덱

모듈은 복호화를 수행하여 재생 가능한 데이터로 변환한다.

3.4.3 서비스 계층

서비스 계층은 인터페이스 계층과 모듈 연결형 계층에서 구성하는 모듈들을 사용한다. 주요 구성 요소는 다음과 같다.

- 프록시 연결 스레드(ProxyServer Connection): 프록시 서버와 통신하기 위한 스레드로서, 사용자의 요청에 따라 USCP 클라이언트 모듈을 사용하여 프록시 서버에 접속하고 프록시 서버로부터 인증 및 서버 노드의 정보를 수신한다.
- 스트림 연결 스레드(Streaming Connection): 프록시 연결 스레드로부터 서버 노드에 대한 정보를 수신한 후, 서버 노드와 RTSP 세션을 설정한다. 그리고 서버 노드로부터 스트림을 수신하여 버퍼에 저장한 후, 재생(Player) 및 릴레이 스레드에게 알린다.
- 릴레이 스레드(Relay Service): 클라이언트 노드와의 RTSP 세션 설정 및 중계 서비스를 제공한다.
- 역량 측정 스레드(Performance Calculator): 실시간으로 노드의 CPU, 네트워크, 메모리 이용률을 측정하여 스트리밍을 제공할 수 있는 클라이언트 수를 결정한다.
- 재생 스레드(Player): 수신한 컨텐츠의 디코딩 및 재생 처리를 담당한다.

그림 4는 서버 노드로부터 스트림을 전송 받은 후 클라이언트 노드로 중계하는 과정과 플레이어에서 재생하는 과정을 보여주고 있다. 스트림 연결 스레드는 생산자로서 스트림을 저장하고 재생 스레드, 릴레이 스레드는 소비자로서 스트림을 읽는다.

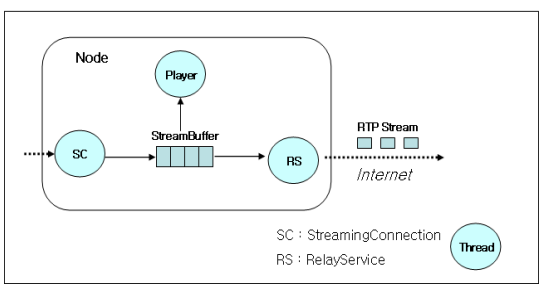

그림 4. 스트림 전달 과정

Ⅳ. 실험 및 성능 평가

4.1 실험 환경

OmniCast264는 Visual Studio 6.0을 이용하여 C++로 구축되어 Windows 환경에서 실험을 수행하였다. 본 논문에서 구현한 노드 소프트웨어의 구성요소 중 재생 스레드(Player)는 아직 개발단계 이므로 구현한 노드가 H.264 비디오 스트리밍을 올바르게 수신하여 중계하는 지를 검증하기 위하여 VideoLAN Project의 VLC Player를 이용하여 재생하였다. 표 2는 실험에 사용된 각 구성요소의 하드웨어 사양을 보여주고 있다. 노드의 처리 역량에 따라 프레임의 재생율을 실험하기 위해 두 종류의 노드(펜티엄 4의 1.8/3.0GHz)를 선택하였다.

표 2. 시스템 구성 요소와 사양

구성요소	하드웨어 사양
인코딩 서버	Dual Core Xeon Pro 5140 2.33GHz CPU, 2G RAM, 100Mbps Network Card, Ateme Kompressor A/V Encoder, SATISEC SAT-10N CCD Camera
스트리밍 서버	Dual Zeon 3.0GHz CPU, 1G RAM, 100Mbps Network Card
프록시 서버	P4 2.8GHz CPU, 1G RAM, 100Mbps Network Card
노드	P4 3.0GHz CPU, 512RAM, 100Mbps Network Card 1대 P4 1.8GHz CPU, 512RAM, 100Mbps Network Card 8대

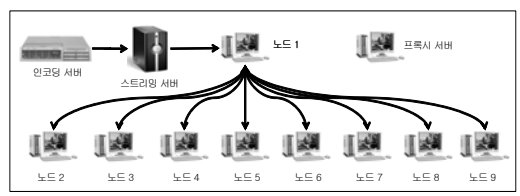

그림 5. 실험 구성 1(병렬)

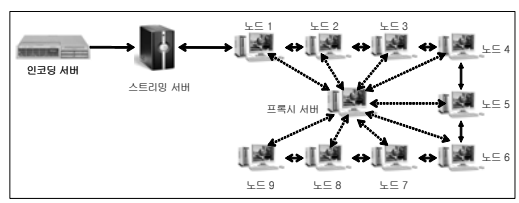

그림 6. 실험 구성 2(직렬)

실험을 위한 네트워크 구성은 가용한 PC 12 대를 이용하여 그림 5, 6과 같이 병렬 및 직렬로 구성하였다.

4.2 실험 결과

4.2.1 프레임 재생율

서버 노드의 CPU 성능에 따른 프레임 재생율을 측정하여 실시간 재생이 가능한지를 검증하였다. 네트워크 구성은 그림 5와 같이 노드 1에서 8개의 다른 노드로 중계하였다. 노드 1은 1.8/3.0 Ghz에서 각각 실험하였으며 그 결과는 그림 7과 같다.

실험 결과 CPU의 성능에 관계없이 노드들은 1초 이내에 H.264 비디오 30프레임을 수신하였으며, 이는 프로토타입 시스템의 구조의 정확성 및 효율성을 보여주고 있다.

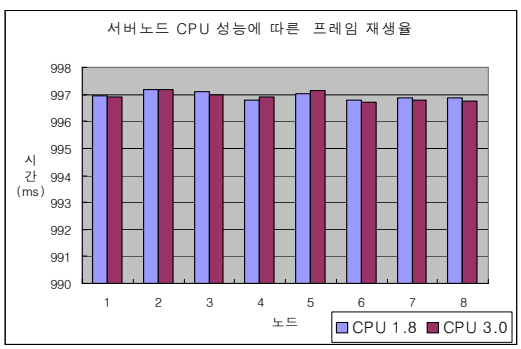

그림 7. 실험 구성 1에서의 서버 노드 CPU 성능에 따른 프레임 재생율

4.2.2 노드의 연결 지연시간

연결 지연시간($T_{connection}$)은 노드가 프록시 서버에게 접속하는 시점부터 비디오 첫 프레임을 수신하는 시점까지의 시간으로 정의된다. 그림 8과 같이 연결 지연 시간은 프록시 서버와의 USCP 세션을 설정하고 서버 노드를 할당받는 시간 $T_{proxyconnection}$, 서버 노드와 RTSP 세션 설정 시간 $T_{rtspconnection}$, 그리고 RTP를 통하여 비디오 첫 프레임을 받는 시간 $T_{initialframe}$으로 구분된다. 네트워크 구성은 그림 6과 같이 직렬로 노드를 구성한 후, 홉 수에 따른 연결 지연 시간을 측정하였다.

실험 결과는 그림 9와 같고, 노드의 연결 지연시간은 총 100~120ms가 소요되었다. 예를 들어 노드 2는 USCP 세션 설정 및 서버 노드 할당 시간이 5ms, RTSP 세션 설정 시간이 60 ms, 프레임 손실

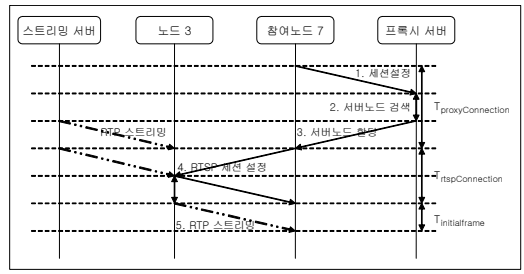

그림 8. 노드의 참여에 따른 연결 지연 시간

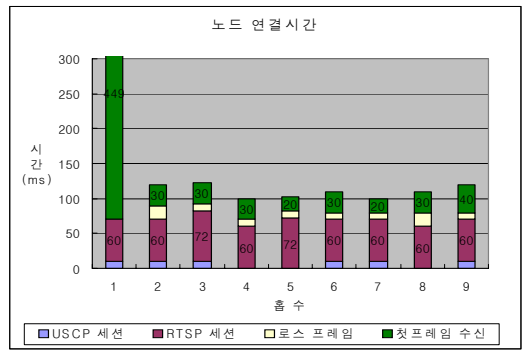

그림 9. 실험 구성 2에서의 노드의 연결 지연시간

로 인한 지연시간이 15 ms, 첫 프레임 받기까지의 지연 시간이 30 ms 이다. 첫 프레임 수신하기 전에 프레임 손실이 일어나는 이유는 하나의 프레임이 여러 개의 RTP 패킷으로 구성되어 있는 경우 프레임 중간 부분 수신시 본 논문에서 손실로 처리하였기 때문이다. 실험 결과에서 노드 1은 RTSP 세션 설정 후 첫 프레임을 수신할 때까지 449ms가 소요되었다. 다른 노드들보다 크게 나타난 이유는 노드 1이 스트리밍 서버에 직접 접속하여 스트리밍을 받기 때문이다.

4.2.3 스트리밍 서비스 복구 지연 시간

스트리밍 서비스 복구 지연시간($T_{recovery}$)은 서버 노드가 이탈함에 따라 새로운 서버 노드로부터 스트리밍 서비스를 재개하는데 소요되는 총 시간이다. 그림 10은 P2P 네트워크에서 노드 2의 이탈에 따른 노드 5의 재배치 시간의 흐름을 묘사한 것이다. 노드 5는 프록시 서버로부터 새로운 서버 노드인 스트리밍 서버를 할당받고 이 서버에 세션을 연결하고 RTP 패킷을 수신한다. $T_{reallocation}$ 은 이탈 노드의 클라이언트 노드가 새로운 서버 노드를 할당받는 시간을 의미한다.

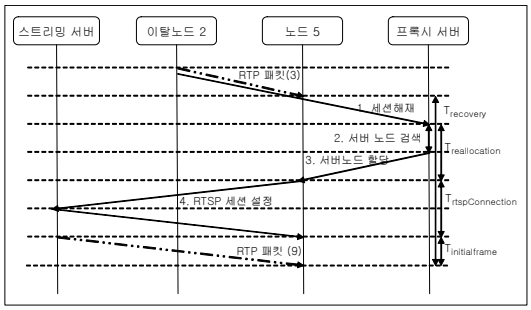

그림 10. 노드의 이탈에 따른 재배치 시간 모델

네트워크 구성은 그림 6과 같이 노드를 직렬로 연결하였다. 노드 2가 이탈하였을 때 노드 3의 스트리밍 서비스 복구 지연 시간 $T_{recovery}$를 측정하였다. 노드 1부터 이탈하지 않은 이유는 노드 1이 이탈할 시에는 노드 2가 다원 스트리밍 서버로 접속하기 때문이다. $T_{recovery}$는 이탈하기 전 최종으로 보낸 프레임을 수신한 시각과 새로운 서버 노드로부터 프레임을 수신한 시각의 차를 이용하였다. 실험 결과 모든 노드에서 연결 지연 시간($T_{connection}$)과 비슷한 100~120ms의 시간이 소요되었다. 노드의 연결 지연 시간과 큰 차이가 없기 때문에 빠르게 에러 복구가 되며 사용자에게 시각적으로 불편함을 주지 않을 것으로 판단된다.

4.3 성능 분석

4.3.1 분산화

중앙 집중식 스트리밍 서버는 가용 능력과 네트워크 대역폭에 의해 클라이언트의 수가 제한되는 반면 P2P 모델은 많은 수의 클라이언트를 서비스할 수 있다. 특히 서버를 분산된 위치에 설치함으로써 네트워크 대역폭을 효율적으로 사용할 수 있는 장점이 존재한다.

한 명의 클라이언트를 서비스하기 위해 필요한 네트워크 대역폭은 (Video Bitrate + Audio Bitrate) * 1.05 로 표현할 수 있다. 여기서 프로토콜 헤더 첨가에 따른 오버헤드를 5%로 계산하였다. 즉, H.264 영상과 AAC 오디오를 RTP를 통해 전송할 경우, Codec/RTP/UDP/IP 헤더가 필요한데, 페이로드에 비해 약 5% 된다고 가정하였다. 만약 전송하는 동영상의 영상이 300kbps, 오디오가 64kbps일 경우, (300+64) * 1.05 = 382.2 Kbps 이다. 한편, 멀티미디어 스트림은 시간에 따라 가변적인 비트율을 가지므로 피크시의 혼잡을 줄이기 위해서는 대략 25~30% 여유분을 두어야 한다. 결국 382.2 Kbps * 1.25 = 477.7 Kbps가 필요하다고 할 수 있다. 따라서 100 Mbps 네트워크 대역폭을 가진 서버는 최대 21명을 서비스할 수 있다고 할 수 있다.

P2P 토폴로지를 가진 10대의 노드가 물리적으로 독립된 곳에 위치한다면 이론적으로 210명을 수용할 수 있는 장점이 존재한다.

4.3.2 실시간 재생

클라이언트 노드에서 실시간 재생은 노드의 연결 지연 시간 및 재생율과 밀접한 관련이 있다. 실험 결과 노드의 연결 지연 시간은 약 100~120 ms이고 재생율은 인코딩 서버의 재생율과 동일하므로 실시간 재생이 가능하다고 볼 수 있다.

4.3.3 에러에 대한 강건함

P2P 시스템에서 노드의 이탈로 인한 끊김없는 재생을 위해서는 스트리밍 서비스 복구 시간이 짧아야 한다. 복구 시간은 실험 환경에 따라 다르게 나타날 수 있지만, 본 논문에서와 같은 LAN 환경에서는 약 100~120ms로 빠르게 복구가 됨을 알 수 있다. 만약 인터넷 환경에서는 네트워크 전송 지연으로 인해 복구 시간이 다소 증가할 수 있다.

4.3.4 모듈화

프록시 서버와 노드에 각각 모듈 연결형 계층을 두고 표준 인터페이스를 정의함으로써 서비스 계층은 동일한 인터페이스를 통해 모듈을 이용할 수 있는 장점이 있다. 예를 들어 토폴로지 구성 알고리즘이나 코덱은 서비스 계층에 영향을 주지 않고 교체할 수 있기 때문에 향후 확장성 및 호환성을 높일 수 있다.

Ⅴ. 결론 및 향후연구

본 논문에서는 높은 확장성과 안정성을 가진 인터넷 방송 시스템을 개발하기 위해 혼합 P2P 네트워크 구조에 기반한 방송 시스템 OmniCast264를 설계 및 구현하였다. 그리고 PC 들을 병렬/직렬로 구성한 후 재생율, 연결 지연 시간, 노드 이탈에 따른 복구 지연 시간을 측정하여 시스템의 성능을 분석하였다. 실험 결과, 인터넷 방송의 초기 프로토타입으로 충분한 성능을 나타냄을 알 수 있었다. 본 논문의 성능 결과는 네트워크 환경이 양호한 곳에서 실험한 것으로서, 향후 다양한 인터넷 환경에서

성능을 측정할 필요가 있다. 또한 노드의 처리 능력에 따라 P2P 토폴로지를 효율적으로 구성하는 알고리즘의 개발 등은 향후 과제로 남겨둔다.

참 고 문 헌

〔1〕 J. Liu, B. Li, and Y.-Q Zhang, "Adaptive Video Multicast Over the Internet," *IEEE Multimedia*, 2003.

〔2〕 B. Alfonsi, "I Want My IPTV: Internet Protocol Television Predicted a Winner," *IEEE Distributed Systems Online*, 2005.

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

〔4〕 박호진, 박광로, "P2P 기술 동향 및 홈 네트워크 응용", 2006.

〔5〕 NICE Project, http://www.cs.umd.edu/projects/nice.

〔6〕 Duc A. Tran, Kien A. Hua, and Tai Do, "ZIGZAG: An efficient peer-to-peer scheme for media streaming," *Proceedings of IEEE INFOCOM*. 1283-1292, 2003.

〔7〕 X. Zhang, J. Liu, etc, "Coolstreaming/DONet: A Data-Driven Overlay Network for Efficent Live Media Streaming," *Proceedings of IEEE. INFOCOM*, 2005.

〔8〕 Xuxian Jiang; Yu Dong; Dongyan Xu; Bhargava, "GnuStream: a P2P media streaming system prototype," *Proceedings of the International Conference on Multimedia and Expo*, Volume 2, July 2003.

〔9〕 Gnutella, http://www.gnutella.com.

〔10〕 R.Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of P2P Architectures and Applications," *Proceedings of the First International Conference on Peer-to-Peer Computing*, 2001.

〔11〕 JRTPLIB, http://research.edm.uhasselt.be/jori/page.

우 문 섭 (Moonsup Woo) 정회원

2002년 2월 한성대학교 컴퓨터시스템 공학과 졸업
2007년 2월 한성대학교 컴퓨터시스템 공학과(석사)
2007년 3월~현재 프로자이너 연구원
<관심 분야> 유비쿼터스 컴퓨팅, 멀티미디어 통신

김 남 윤 (Namyun Kim) 정회원

1992년 2월 서울대학교 컴퓨터공학과 졸업
1994년 2월 서울대학교 컴퓨터공학과(석사)
2000년 2월 서울대학교 컴퓨터공학과(박사)
1999년~2002년 2월 삼성전자 무선사업부 책임 연구원
2002년~현재 한성대학교 정보시스템공학과 조교수
<관심 분야> 멀티미디어 통신, 정보 보안

황 기 태 (Kitae Hwang) 정회원

1986년 2월 서울 대학교 컴퓨터공학과 졸업
1988년 2월 서울 대학교 컴퓨터공학과(석사)
1994년 2월 서울 대학교 컴퓨터공학과(박사)
2000~2001년 UC Irvine 방문 교수
1994년~현재 한성대학교 컴퓨터시스템 공학과 교수
<관심 분야> 유비쿼터스 컴퓨팅, 인터넷 시스템, 모바일 보안 등