스트리밍 미디어 캐쉬 서버를 위한 RTSP/RTP 스트림 제어 기법

(A RTSP/RTP Stream Control Mechanism for Streaming Cache Server)

오 재 학 † 차 호 정 †† 최 영 근 †††
(Jaehak Oh) (Hojung Cha) (Youngkeun Choi)

요 약 본 논문은 효율적인 스트리밍 캐쉬 서버 개발에 필수적인 스트림 제어 기법의 설계와 구현에 대해 기술한다. 논문에서 사용한 스트리밍 프로토콜은 표준 프로토콜인 RTSP/RTP/RTCP이다. 스트림 제어 기법은 기존 스트리밍 환경에서 On-Demand 캐싱과 실시간 Splitting 기능등을 제공하며, 프로토콜 수준의 세션 관리와 캐쉬 관리 기능으로 구성된다. 프로토콜 수준의 세션 관리는 RTSP 세션 제어, RTP 송수신과 RTCP 세션 제어를 양방향으로 제공하고, 캐쉬 관리 기능은 RTP 패킷 수준의 캐싱과 패킷 미디어 일관성 유지를 제공한다. 스트리밍 캐쉬 서버는 Apple의 QTSS 서버와 QuickTime Player로 구성된 스트리밍 환경에서 리눅스 시스템을 기반으로 구현하였다. 구현된 시스템은 On-Demand 캐싱과 스트림 Splitting 서비스를 원활하게 제공하였다.

키워드 : 스트림 제어, RTSP, RTP, 캐쉬 서버

Abstract This paper presents the design and implementation of stream control mechanisms which are necessary for the development of an efficient streaming cache server. The streaming protocols used in our implementation are the RTSP/RTP/RTCP standards. The mechanisms support both the on-demand media caching and real-time media splitting applications. The core of the stream control includes the session management, which handles the RTSP/RTCP control session and the RTP transport session, and the cache block management which efficiently manages the RTP-based cache blocks stored in the cache server. The streaming cache server with the proposed stream control mechanism has successfully been implemented on a Linux platform and it works well with the Apple's QTSS server and the QuickTime player for both on-demand and splitting media services.

Key words : Stream Control, RTSP, RTP, Cache Server

1. 서 론

최근 동영상 스트리밍 서비스는 유무선 개인용 통신 장비의 발달과 다양한 콘텐츠 증가로 보편화되고 있으나, 이에 비례하여 증가하는 네트워크 트래픽은 사용자 QoS를 저해하는 주요인이 되고 있다. 이러한 문제를 해결하기 위하여 사용자 QoS 제공을 위한 스트리밍 미디어 캐싱 기술에 대한 관심이 집중되고 있다[1]. 특히, 기존 스트리밍 환경과의 호환성을 고려한 스트리밍 미디어 캐싱 기술이 캐쉬 서버의 기본 요소로서 연구되고 있다[2].

스트리밍 미디어 캐싱은 지역 네트워크 캐싱 시스템을 기반으로 사용자가 선호하는 콘텐츠의 일부를 저장하고 관리하는 기술이다. 사용자 요구가 발생했을 때 요청된 콘텐츠와 캐싱된 콘텐츠가 일치하면, 해당 콘텐츠를 지역 네트워크의 캐싱 시스템에서 전송받기 때문에 사용자 대기 시간과 원격 스트리밍 서버의 콘텐츠 전송량을 감소시키는 효과가 있다. 또한, 기간 네트워크 트래픽을 지역망으로 이전시키는 효과가 있어 전체 네트워크 트랙픽의 균일화를 이룰 수 있다. 스트리밍

· 본 연구는 정보통신기초기술연구지원사업(2001-07603)과 과학재단 특정기초연구사업(R01-2002-000-00141-0)에 의해 지원되었음.
† 비 회 원 : 코어세스
ojh@corecess.com
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
hjcha@cs.yonsei.ac.kr
††† 종신회원 : 광운대학교 컴퓨터과학과 교수
ygchoi@cs.kwangwoon.ac.kr
논문접수 : 2002년 3월 15일
심사완료 : 2003년 3월 10일

미디어 캐쉬 서버는 클라이언트의 전방위에 위치하며, 스트리밍 서버로부터 전송될 콘텐츠에 대한 캐싱과 재전송 서비스를 위한 스트림 제어 기술이 주요 개발 요건이다. 즉, 콘텐츠 캐싱 방식, 캐쉬 블럭 저장 방식, 미디어 전송 프로토콜등이 기본적인 고려 사항이다.

캐쉬 서버의 콘텐츠 캐싱은 정적 캐싱과 동적 캐싱으로 구분한다. 정적 캐싱은 스트리밍 서버로부터 완전한 콘텐츠를 내려받아 캐싱하여 클라이언트의 요구시 스트리밍 서버에 상관없이 자체적으로 서비스 할 수 있다. 정적 캐싱 구조는 개발이 용이하나 제한된 캐쉬 저장 공간에서 동영상과 같은 대용량 콘텐츠에 대해 효율적인 캐싱 정책의 적용이 어렵고, 캐쉬 미스(miss)인 경우에 콘텐츠를 내려받는 동안 서비스 지연으로 사용자 대기 시간이 길어지는 단점이 있다. 동적 캐싱은 콘텐츠의 일부분을 캐싱하고 요구가 발생했을 때 캐싱이 안된 부분을 서버로부터 전송받으며 동시에 클라이언트로 전송한다. 동적 캐싱은 캐쉬 저장 공간의 효율적인 활용과 다양한 캐싱 정책 적용이 용이하지만, 양방향 전송 구조를 고려한 캐쉬 서버의 설계가 필요하다. 캐쉬의 저장 미디어는 원본 미디어와 패킷 미디어가 있다. 원본 미디어는 서버로부터 콘텐츠를 복사하거나 스트리밍 프로토콜에 의해 전송받은 패킷을 복원한 것이다. 캐쉬 서버에서 캐싱된 콘텐츠의 서비스는 스트리밍 서버의 서비스 구조와 동일하게 구성되어야 한다. 반면에, 패킷 미디어는 스트리밍 프로토콜 수준에서 전송된 패킷이 저장 단위이고 패킷들의 일정량을 캐쉬의 입출력 단위로 구성한다. 패킷 미디어 방식은 미디어 재구성이 필요없지만 프로토콜 정보를 입출력 정보로 재구성할 필요가 있다[3]. 캐쉬 서버의 전송 프로토콜은 기존 멀티미디어 스트리밍 프로토콜을 활용하는 방법과 캐쉬 전용 프로토콜의 개발을 고려할 수 있다. 멀티미디어 스트리밍 프로토콜의 활용은 기존 스트리밍 환경의 변화없이 캐쉬 서버의 구현을 용이하게 하지만 캐쉬 서버 내부적으로 프로토콜 정보 변환 기능을 추가해야 한다. 캐쉬 전용 프로토콜의 개발은 효율적인 콘텐츠 전송 구조를 구성할 수 있지만 기존 스트리밍 환경과의 호환성을 유지할 수 없다.

이러한 캐쉬 서버 구성 요건은 스트리밍 환경의 재구성이나 재활용 문제로 요약할 수 있다. 스트리밍 환경의 재구성은 스트리밍 미디어 캐싱을 포함한 새로운 표준을 구성하여 효율적인 구조를 제시할 수 있지만 표준화되기까지 기술적인 문제와 기존 스트리밍 환경의 이해 관계에 따라 많은 시간이 걸릴 수 있다. 반면에, 기존 스트리밍 환경의 재활용은 표준화된 멀티미디어 프로토콜에 기반한 스트림 제어 기법을 개발하여 스트리밍 환경과의 호환성, 코덱에 독립적인 미디어 캐싱과 단기간 개발성을 제공할 수 있다. 즉, 기존 스트리밍 환경에 대한 독립성과 호환성을 동시에 만족시킬 수 있다.

본 논문에서는 표준 RTSP[4]/RTP[5]/RTCP[5] 프로토콜을 바탕으로, 미디어 캐싱 및 스트리밍 기능을 제공하는 스트리밍 캐쉬 서버를 위한 스트림 제어 기법을 구현하였다. 캐쉬 서버는 스트리밍 서버의 콘텐츠를 선행 미디어와 후행 미디어로 구분하여 콘텐츠 선호도에 따라 선행 미디어 크기를 조절할 수 있는 동적인 캐싱 구조를 유지한다. 세션 제어와 미디어 전송은 RTSP/RTP 스트리밍 프로토콜을 사용하고, 캐쉬 내부적으로 프락시 기능과 캐싱 기능을 제공하여 기존 스트리밍 환경과의 호환성을 유지하며, 전송 패킷 단위의 캐싱 및 입출력 구조를 제공한다. 패킷 수준의 스트리밍 구조는 On-Demand 캐싱 서비스를 가능하게 하며, 스트림 Splitting 기능은 방송 스트림에 대한 세션 제어와 패킷 중계 전송을 제공한다.

본 논문과 관련된 기존 연구는 웹 캐싱, 버퍼를 이용한 캐싱 구조와 미디어 캐싱 프락시 서버 분야등이 있다. 웹 캐싱은 CERN http가 출발된 이후에 종단 시스템과 종단 시스템 간에 초기 재생 시간을 줄이고 네트워크 트래픽을 줄이고자 연구되어 왔다. Harvest[6]와 Squid[7]의 주된 연구 내용은 계층적 캐싱(hierarchical caching)과 상호 연관성 캐싱(cooperative caching)이다. 웹 캐싱은 텍스트, 그림 등의 정적 콘텐츠가 주 대상이며, 캐싱 정책은 독립적인 객체를 단위로 서버와 캐쉬 서버 사이에 일대일로 전송하고 동영상과 같은 대용량 콘텐츠들도 정적 콘텐츠들과 동일하게 적용한다. 버퍼를 이용한 미디어 캐싱 구조의 연구는 디스크나 네트워크를 경유해 버퍼로 전송된 미디어의 일부분을 동일한 요구에 대해서 효율적으로 재활용할 것을 제안하고 있다. 이들 구조는 서버의 시스템 내부의 전송 버퍼 풀과 이웃한 클라이언트의 버퍼의 공유 방식으로 구분한다. 서버 내부의 버퍼 공유는 동일한 미디어의 요구가 있을 경우, 선행 요구로 인해 디스크로부터 버퍼로 읽어 들인 미디어의 일부가 후행 요구의 유효 범위내에 있다면 이를 재활용하는 것이다. 대표적인 예로 간격(Interval) 캐싱이 있다[8]. 클라이언트 버퍼의 공유는 선택한 미디어를 서버에 요구하기 전에 이웃한 클라이언트에 관련 미디어가 있는지 확인하는 정책이다. 유효 범위 내에 있는 미디어가 이웃한 클라이언트에 존재하는 경우 서버를 거치지 않고 일부 혹

은 전체를 빠르게 전송 받을 수 있다[9]. 미디어 캐싱 프락시 서버는 AT&T 연구소에서 RTSP와 RTP 프로토콜 환경을 기반으로 Real Network사의 Real Server와 Real player를 활용한 사례가 있다[10]. 상용 시스템으로는 Inktomi[11], Network Appliance[12], CacheFlow[13] 등의 업체들이 전용 캐싱 시스템을 개발하고 상품화였다. 그러나 대부분의 업체들이 시스템 구성 및 기술에 대한 비공개 정책을 고수함으로 인하여 연구 성과는 알려지지 않고 있다.

논문의 구성은 다음과 같다. 2장에서 스트리밍 미디어 캐쉬 서버의 구조를 기술하고, 3장에서 On-Demand 캐쉬 모듈에 대한 RTSP/RTCP 세션 제어와 RTP 패킷 미디어 전송 구조를 기술한다. 4장에서 스트림 Splitting 모듈의 방송 스트림에 대한 세션 제어와 패킷 중계 전송을 기술하고, 5장에서 결론을 맺는다.

2. 스트리밍 미디어 캐쉬 서버

스트리밍 미디어 캐쉬 서버는 오디오와 비디오를 포함한 대용량 미디어의 효율적인 전송을 목적으로 사용자 네트워크에 위치하여 네트워크의 효율적인 구성과 사용자 QoS의 향상을 유도한다. 본 논문에서 구현한 캐쉬 서버는 RTP 패킷 수준의 캐싱 구조와 전송 구조를 지원하고, RTSP와 RTCP를 통한 스트리밍 서버와 클라이언트의 양방향 세션 제어를 지원한다. 표준 스트리밍 프로토콜인 RTSP와 RTP는 스트리밍 서버와 클라이언트의 일대일 관계만을 고려한 것으로 스트리밍 미디어 캐싱에 그대로 적용할 수 없다. 또한 기존의 스트리밍 환경과의 호환성 문제 때문에 프로토콜의 변경도 용이하지 않다. 따라서 캐쉬 서버는 표준 프로토콜을 지원하도록 스트리밍 서버 내부적인 기능 확장을 통해 구현되어야 하며, 이는 프락시 서버와 스트리밍 미디어 캐싱의 기능을 지원하도록 RTSP/RTCP 세션 관리와 RTP 미디어 전송 측면에서 이루어져야 한다. 세션 관리는 RTSP 세션 설정과 RTCP 미디어 전송 세션 관리로 구분한다. RTSP의 세션 관리는 미디어 정보 교환, 전송 프로토콜 정보 교환, 세션 제어를 수행하는 RTSP 기능에 대하여 캐쉬 서버의 투명성, 프로토콜의 번역 기능, 콘텐츠 정보 관리, 양방향 세션 관리 등의 기능을 개발해야 한다. RTCP 세션 관리는 스트리밍 세션의 설정 이후 스트리밍 서버와 클라이언트의 관계를 클라이언트와 캐쉬 서버, 캐쉬 서버과 스트리밍 서버 관계로 구성하고 캐쉬 상태를 반영한 양방향 스트리밍 정보 교환 기능을 제공해야 한다. 미디어 전송은 RTP 패킷 수준의 재전송 기술로 미디어에 대한 코덱 정보 없이 패킷에 담겨진 RTP 헤더의 패킷 시간, 순서, 세션 등의 정보를 번역하여 전송하는 기술이다.

스트리밍 미디어 캐쉬 서버는 그림 1과 같이 캐쉬 서버 관리자들과 캐쉬 화일 시스템으로 구성된다. 캐쉬 서버의 관리자들은 네트워크 관리자, 세션 관리자, 콘텐츠 관리자, 캐쉬 관리자, 프로토콜 제어기로 구성되고 각각의 역할은 다음과 같다. 네트워크 관리자는 스트리밍 서버와 클라이언트의 입출력을 담당하고 패킷 단위의 효율적인 입출력을 위해 버퍼 관리와 입출력 스케줄링을 제공한다. 버퍼 관리는 세션당 두개의 버퍼를 배정하고 각각 네트워크와 캐쉬의 입출력을 교대로 수행한다. 버퍼의 크기는 캐쉬 블럭과 일치시켜 빈번한 입출력 호출을 보완하도록 하였다. 입출력 스케줄링은 전송 버퍼의 상태에 따라 캐쉬의 출력에 대한 위급도를 반영하여 수행한다. 세션 관리자는 클라이언트의 요구를 받아 캐쉬와 스트리밍 서버의 상태 정보에 따라 수용 여부를 결정하고 설정된 세션의 종료 때까지 세션을 유지 관리한다. 또한, 기존의 스트리밍 프로토콜을 이용하여 스트리밍 서버 세션과 클라이언트 세션을 독립적으로 유지하고 캐싱될 미디어 정보를 콘텐츠 관리자에게 제공한다. 콘텐츠 관리자는 수용된 세션에 대해서 클라이언트로 전송할 캐쉬 미디어를 제공하고 스트리밍 서버로부터 받은 후미 미디어와 캐쉬 서버의 선두 미디어에 일관성 있는 관리를 지원한다. 그리고 미디어의 전송시 클라이언트의 성능 및 네트워크 대역폭에 따라 미디어의 전송 품질을 결정하고 캐싱될 미디어에 대해 손실 관리를 담당한다. 서비스에 따라 On-demand 캐쉬, Pass-through, Stream splitter 모듈로 구성된다. On-demand 캐쉬는 스트리밍 서버에서 화일로 구성된

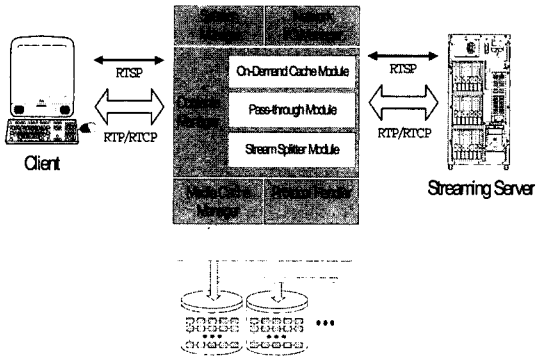


그림 1 캐쉬 서버 구조

미디어에 대해 캐싱 기능과 전송 서비스를 제공한다. Pass-through는 프락시 서버의 역할을 하고, Stream splitter는 실시간으로 방송되는 스트리밍 미디어에 대해서 단일 스트림으로 전송 받아 다수의 지역 네트워크 사용자들에게 복사를 통해서 전송하는 서비스다. 프로토콜 제어기는 상위 세션 관리자와 콘텐츠 관리자의 RTSP/RTP 프로토콜 처리를 담당한다. 클라이언트와 스트리밍 서버에서 오는 세션 관리 프로토콜을 번역하고 클라이언트에 대한 응답과 스트리밍 서버에 대한 요구 프로토콜을 생성한다. 캐쉬 관리자는 캐싱과 관련된 정책을 수행하고 세션 관리자와 콘텐츠 관리자에 세션유지를 위한 캐쉬 미디어의 정보를 전달한다.

그림 2는 화일 블럭에 기반한 캐쉬 입출력 구조를 보여주고 있다. 캐쉬 입출력 구조는 제어 구조와 고정 크기 캐쉬 블럭으로 구성된다. 제어 구조는 콘텐츠 리스트와 빈 블럭 리스트에 의한 캐쉬 블럭 관리 구조이다. 콘텐츠 리스트는 화일 블럭에 대한 리스트 구조를 내포하며 물리적인 위치에 상관없이 리스트의 순서에 따라 캐쉬 콘텐츠를 관리하다. 빈 블럭 리스트는 콘텐츠의 가감에 따라 발생하는 빈 블럭들의 집합이다. 캐쉬 블럭은 네트워크 입출력에 대한 디스크 입출력 단위이며 화일로 할당된다. 이것은 패킷 단위 캐싱에 있어 빈번하게 발생하는 디스크 입출력에 대한 효율적인 입출력 구조를 제공하고 단기간 개발의 효과를 얻을 수 있다. 캐쉬 저장 공간은 범용 화일 시스템으로 초기화된 디스크 드라이브에 고정 크기의 화일 블럭을 순차적으로 생성하여 디스크 저장 공간을 선점하도록 구성한다. 화일 블럭의 순차적인 할당은 범용 화일 시스템에서 화일의 저장 공간 할당 방식에 따라 화일 단위의 논리적인 연속성을 제공한다. 즉, 범용 화일 시스템이 제공하는 물리적인 연속 할당을 최대한 반영한 것이다. 캐쉬 블럭 입출력은 범용 화일 시스템이 제공하는 Direct I/O를 이용한다. Direct I/O는 화일 시스템의 버퍼 캐쉬를 경유하지 않고 디스크로부터 사용자 메모리에 입출력을 수행한다. 이는 낮은 수준의 입출력 제공과 멀티미디어에 대한 버퍼 캐쉬 부하를 감소시키는 장점이 있다.

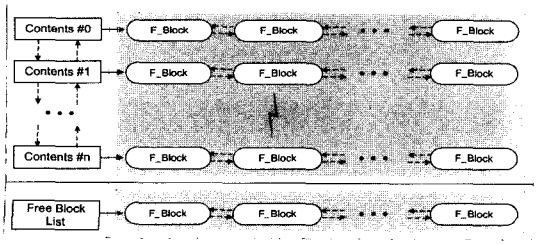


그림 2 캐쉬 입출력 구조

3. On-Demand 캐쉬 모듈

다음은 RTSP/RTP를 활용한 On-Demand 캐쉬 서비스 구성시 세션 제어와 응용에 대한 세부적인 구현을 기술하고 패킷 미디어에 대한 전송 구조를 제시한다.

3.1 RTSP 제어

RTSP는 DESCRIBE, SETUP, PLAY, TEARDOWN 메시지 교환에 따라 서비스 요청, 서비스 설정, 서비스 시작과 종료를 결정하는 상태 전이를 갖는다. 세션 설정 과정은 메시지 용도에 따라 두 단계로 나눠진다. 첫 번째 단계는 DESCRIBE 메시지를 사용한 세션의 준비 단계이며, 클라이언트의 요구에 대해 콘텐츠 유무를 판단하고 스트리밍 서버와 콘텐츠 정보를 교환한다. 캐쉬 서버에서는 클라이언트의 DESCRIBE 메시지를 받아 캐쉬 히트와 미스를 판단하고 DESCRIBE 메시지를 스트리밍 서버에 보내 콘텐츠 정보 수집과 캐쉬의 일관성을 검사를 한다. 두 번째 단계는 서비스 세션에 대한 개설, 종료와 사용자 제어를 제공하는 단계로 SETUP, PLAY, TEARDOWN 메시지로 구성된다. 이 단계에서는 클라이언트 요구에 대한 캐쉬 서버와 스트리밍 서버가 세션에 대한 자원 할당과 해제를 수행하며 VCR 원격 제어와 같은 사용자 편의를 제공한다.

그림 3은 캐쉬 서버의 RTSP 세션 처리 과정과 데이타 교환을 보여주고 있다. RTSP 세션 처리는 클라이언트 요구에 따른 캐쉬 히트와 미스로 구분한다. 캐쉬 미스는 클라이언트의 DESCRIBE 메시지에서 지정한 콘텐츠가 캐쉬 서버에 없는 경우이며 콘텐츠의 캐싱과 클라이언트로의 전송을 동시에 수행하게 된다. 캐쉬 서버는 받아 들인 콘텐츠의 일부 혹은 전체를 캐싱하여 동일한 요구에 대한 선행 미디어를 구성한다. 캐쉬 히트는 캐쉬 미스와 같이 클라이언트의 DESCRIBE 메시

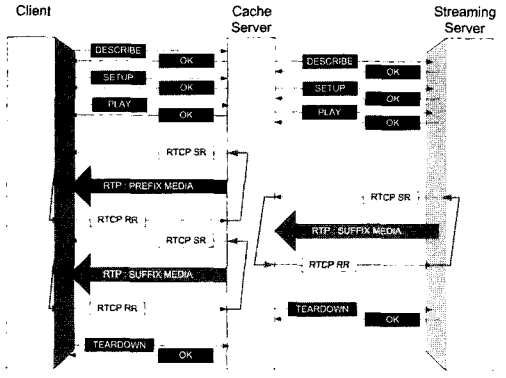


그림 3 RTSP 세션 처리와 데이타 교환

지에서 결정된다. 캐쉬 서버는 캐쉬 미스시에 캐싱된 선행 미디어를 클라이언트에 전송하고 스트리밍 서버로부터 나머지 부분인 후행 미디어를 받아들여 선행 미디어에 이어서 캐싱한다. 서비스 종료 후에 캐싱 정책에 따라 캐싱된 미디어 크기를 결정한다. 다음은 RTSP 세션 설정에 대한 메시지별 세부 절차와 캐쉬 서버의 응용을 기술한다. 클라이언트의 DESCRIBE 메시지는 다음과 같다.

```
DESCRIBE rtsp://207.208.214.10:554/realqt.mov RTSP/1.0
CSeq: 1
Accept: application/sdp
Bandwidth: 2147483647
Accept-Language: en-US
User-Agent: QTS (qtver=5.0b11;os=Windows NT 5.0Service Pack 1)
```

클라이언트의 DESCRIBE 메시지는 요구된 콘텐츠의 URI 정보와 클라이언트의 상태 정보를 가지고 있다. 상태 정보는 프로토콜, 대역폭, 언어 인코딩 그리고 클라이언트 버전 정보를 담는다. 캐쉬 서버에서는 DESCRIBE의 URI 정보를 얻어 캐쉬 히트와 미스를 판단하고 스트리밍 서버에 요구할 콘텐츠에 대한 DESCRIBE 메시지 구성을 결정한다. 다음은 캐쉬 서버의 DESCRIBE 메시지에 대한 스트리밍 서버의 응답이다.

```
RTSP/1.0 200 OK
Server: QTSS/3.0Preview [v240]-Linux
. . .
Content-Type: application/sdp
x-Accept-Retransmit: our-retransmit

Content-Base:rtsp://207.208.214.10:554/realqt.mov/
v=0
s=/realqt.mov
. . .
c=IN IP4 128.134.64.111
. . .
a=range:npt=0-360.98500
m=audio 0 RTP/AVP 96
b=AS:47
a=rtpmap:96 X-QDM/22050/2
a=control:trackID=3
a=x-bufferdelay:3.76
m=video 0 RTP/AVP 97
b=AS:767
a=rtpmap:97 X-QT/600
a=control:trackID=4
```

DESCRIBE 응답은 스트리밍 서버와 콘텐츠 관련 정보에 대한 RTSP 메시지와 콘텐츠 내역을 기술하는 SDP 프로토콜로 나누어진다. RTSP 메시지는 콘텐츠의 URI 확인과 유효성 정보, 프로토콜 등을 기술하고, SDP 프로토콜에서는 실제 소스 콘텐츠의 위치, 콘텐츠의 재생시간과 미디어 코덱 유형 정보등의 상세정보를 담는다. SDP 정보 중 c=IN IP4 source address는 콘텐츠를 소유한 스트리밍 서버 위치 정보이며 캐쉬 서버나 프락시 서버를 경유해 전송하는 경우에 스트리밍 서버를 표시하지 않도록 해야 한다. 즉, 클라이언트를 기준으로 콘텐츠 전송에 직접적인 관련이 있는 스트리밍 서버를 표시한다. a=range는 콘텐츠 재생 시간에 대한 기술로 SMPTE, NPT를 이용한다. SMPTE는 미디어에 대한 상대적인 시간 표현으로 hours:minutes: seconds: frames. subframes으로 나타내고, NPT는 시, 분, 초와 초에 대한 소수로 표현한다. 다음은 클라이언트의 SETUP 메시지이다.

```
SETUP rtsp://207.208.214.10:554/realqt.mov
. . .
Transport: RTP/AVP;unicast;client_port=6970-6971
x-retransmit: our-retransmit
x-transport-options: late-tolerance=1.867000
User-Agent: QTS (qtver=5.0b11;os=Windows T 5.0Service Pack 1)
Accept-Language: en-US
```

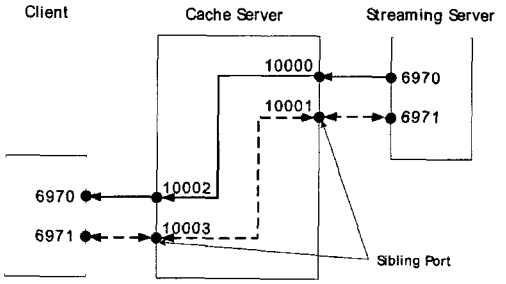
SETUP 메시지는 DESCRIBE에 기술된 URI와 미디어의 트랙 번호를 통해 요청한 콘텐츠를 지정하고 전송 프로토콜과 그 설정정보를 기술한다. 미디어의 트랙 정보는 DESCRIBE 응답의 미디어 정보를 참조하여 클라이언트나 캐쉬 서버에서 미디어 별로 SETUP 메시지를 구성한다. 전송 프로토콜에 대한 Transport 인자는 송신측에서 지원하는 프로토콜을 한 개 이상 나열하여 수신측에서 선택할 수 있는 권한을 준다. SETUP 메시지의 예에서 클라이언트는 RTP/AVP 프로토콜과 unicast 서비스를 지정했다. 프로토콜 설정 정보는 RTP : 6970, RTCP : 6971로 포트 배정에 대한 것이다. 다음은 스트리밍 서버의 SETUP 응답이다.

```
RTSP/1.0 200 OK
Server: QTSS/3.0Preview [v240]-Linux
. . .
Session: 872658782624407063O
Date: Tue, 27 Jan 2002 02:12:44 GMT
Expires: Tue, 27 Jan 2002 02:12:44 GMT
Transport:RTP/AVP;unicast;client_port=10000-10001;source=128.134.64.1
11;
server_port=6970-6971
x-Retransmit: our-retransmit
```

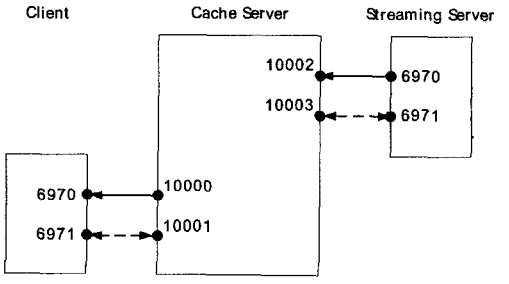
SETUP 응답은 요청한 미디어에 대해 세션을 배정하고 전송 플로토콜을 지정한다. 세션 배정은 Session 인자를 통해 고유 식별자로 할당하며 세션 종료시까지 유효하고, 캐쉬 서버와 스트리밍 서버는 각각의 클라이언트에 대한 Session 인자를 배정하고 관리한다. 전송 프로토콜은 클라이언트가 지정한 것 중 하나를 선택하

고 그에 상응하는 스트리밍 서버의 프로토콜 설정 정보를 담는다. SETUP 응답 예에서 스트리밍 서버는 RTP : 10000, RTCP : 10001로 포트 배정을 하였다.

캐쉬 서버의 RTP/RTCP 포트 배정은 그림 4와 같이 캐쉬 미스와 캐쉬 히트 상황에 따라 다른 구조를 갖는다. 캐쉬 미스가 발생했을 때 캐쉬 서버는 DESCRIBE 단계에서 미디어에 대한 캐싱 정보를 파악하고 클라이언트의 SETUP 메시지를 받아 요청된 미디어에 대해 세션 대기 리스트에 추가한다. 캐쉬 서버는 클라이언트와 세션을 설정하기 전에 소스 미디어를 소유한 스트리밍 서버와의 세션을 설정하고 RTP/RTCP 포트를 배정한다. 이 절차가 문제없이 수행된다면 클라이언트와의 세션을 설정한다. 캐쉬 서버 내부에서 클라이언트 세션과 스트리밍 서버 세션을 위해 배정된 포트를 자매포트(sibling port)라고 한다. 미디어 전송과 세션 제어는 캐쉬 서버의 자매포트를 통해 터널링되어 수행된다. 캐쉬 서버는 기존 프락시 서버와 동일한 세션 설정을 제공하며 내부적으로 전송된 미디어에 대한 캐싱을 수행한다. 캐쉬 히트일 때 포트 배정은 그림 4(아래쪽)과 같은 구조를 갖는다. 캐쉬 서버는 클라이언트와 스트리밍 서버에 대해서 독립적인 세션을 구성하며 클라이언트의 요구에 대해 캐싱된 미디어를 전송하고 스트리밍 서버로부터 캐싱되지 않은 부분을 받아들여 선행 미디어에 이어서 캐싱한다. 각각의 세션에 대한 캐쉬 서버의 포트 배정은 직접적인 연관성이 없는 독립된 구조이며 클라이언트 세션이 중단되어도 캐싱 정책에 따라 스트리밍 서버 세션은 유지될 수 있다. 다음은 세션 설정 이후 서비스의 시작을 요청하는 PLAY 메시지이다.



(a) 캐쉬 미스



(b) 캐쉬 히트

그림 4 RTP/RTSP 포트 배정

```
PLAY rtsp://207.208.214.10:554/realqt.mov RTSP/1.0
. . .
Range: npt=0.000000-51.985000
x-prebuffer: maxtime=2.000000
Session: 8726587826244070630
```

PLAY 메시지는 미디어의 URI, 재생 요청 범위와 수신 버퍼의 제한 값을 담는다. 재생 요청 범위는 DESCRIBE 단계와 동일한 Range 인자를 배정한다. 캐쉬 서버에서는 캐싱된 미디어의 Range 값에 따라 후행 미디어를 위한 Range 값을 결정한다. 일반적으로 클라이언트의 PLAY와 DESCRIBE의 Range 값은 동일하다. 다음은 서비스의 수락여부를 담은 PLAY 응답이다.

```
RTSP/1.0 200 OK
. . .
RTP-Info:
url=trackID=3;seq=17115;ssrc=1116617086;rtptime=1495207282,
url=trackID=4; seq=20706; ssrc=253917952;rtptime=2095478881
```

PLAY 응답은 세션에 대한 전송 프로토콜의 기준 정보를 제공하여 네트워크 상태 검사와 에러 복구에 활용한다. PLAY 응답의 예에서 기준 정보는 RTP-Info이며 세션당 RTP 선행 패킷에 대한 seq, rtptime 정보를 제공한다. *seq, rtptime*는 상대적인 변위량으로 스트리밍 서버의 시작 시점으로부터 누적된 값이다. 캐쉬 서버는 *RTP-Info*의 *seq, rtptime*를 기준으로 전송될 패킷에 대해서 절대 변위량을 계산한다.

```
TEARDOWN rtsp://207.208.214.10:554/realqt.mov RTSP/1.0
CSeq: 5
Session: 8726587826244070630
User-Agent: QTS (qtver=5.0b11;os=Windows NT 5.0Service Pack 1)

RTSP/1.0 200 OK
Server: QTSS/3.0Preview [v240]-Linux
Cseq: 5
Session: 8726587826244070630
Connection: Close
```

위 예는 TEARDOWN 메시지와 응답이다. 지정된 Session에 대한 자원을 해제하고 종료를 수행한다. 캐쉬 서버는 사용자의 의도적인 서비스 종료로 TEARDOWN이 발생하는 경우 스트리밍 서버측의 세션을 고려해야 한

다. 즉, 캐쉬 정책에 의해서 계산된 미디어 캐싱량에 따라 세션의 유지와 종료를 결정한다.

3.2 RTP 캐싱과 스트리밍

캐쉬 서버는 RTP 패킷 수준의 캐싱과 스트리밍 기능을 제공하기 위해 패킷 미디어의 캐싱 구조와 전송 기능을 제공한다. 패킷 미디어의 캐싱 구조는 그림 5와 같이 세션에 대한 다중 미디어 관리 구조이며 콘텐츠 정보, RTP 패킷 인덱스와 패킷 미디어로 구성된다. 콘텐츠 정보는 캐쉬 미스 일 때 RTSP 세션 설정 과정에서 교환되는 프로토콜 정보와 미디어 정보로 구분하여 저장하고 캐쉬 히트일 때 클라이언트와 스트리밍 서버에 대한 RTSP 구성 정보로 활용한다. RTP 패킷 인덱스는 RTP 패킷의 seq, ts, 크기와 패킷 미디어에 대한 변위 값으로 구성되고, 패킷 미디어와 인덱스의 분리는 UDP 기반의 RTP 전송에서 패킷 도착 순서의 역전 현상을 쉽게 복원할 수 있게 한다. 패킷 미디어의 캐싱 구조는 선행 미디어와 후행 미디어 사이에 일관성 유지를 지원한다. 캐쉬 일관성 유지는 RTSP 콘텐츠 재생 범위 지정과 RTP 시간정보를 일치시키는 과정으로 미디어의 중복 전송과 다중화된 콘텐츠에 대한 기준 미디어를 선택하는 어려운 문제이다. 미디어의 중복 전송은 스트리밍 서버가 요구된 후행 미디어의 앞부분을 전송하여 선행 미디어의 뒷 부분과 중복되서 발생하는 문제이며 클라이언트의 재생시 노이즈의 원인이 된다. 캐쉬 서버는 중복된 부분을 검출하기 위해 RTSP의 PLAY 응답에 담겨진 미디어 기준 정보를 활용한다. 기준 미디어 선택은 스트리밍 서버의 콘텐츠가 다중 미디어 구조일 때 각 미디어에 대한 캐싱량의 조절과 후행 미디어 전송을 위한 기준을 제공한다. 즉, RTSP의 재생 범위 지정이 다중화된 콘텐츠에 대한 것이고 개별 미디어 단위로 적용하지 않기 때문이다. 캐쉬 서버는 다중화된 콘텐츠에 대해 캐싱을 위한 기준 미디어를 선택하고 후행 미디어의 연결 정보로 활용한다. 기준 미디어 선택은 RTP 시간을 기준으로 가장 적게 혹은 많이 전송받은 미디어를 고려할 수 있다. 적게 받은 미디어를 선택하는 경우 각각의 재생률을 기준으로 초과한 부분을 삭제하지만 RTP 패킷을 미디어 재생률에 적용하는 것은 정확성에 문제가 있어 미디어 중복을 발생시킨다. 많이 받은 미디어 선택은 나머지 미디어에 대한 삭제가 없고, 클라이언트의 재생시 약간의 지터가 발생할 수 있지만 사용자 QoS에 영향을 미치지 않는다.

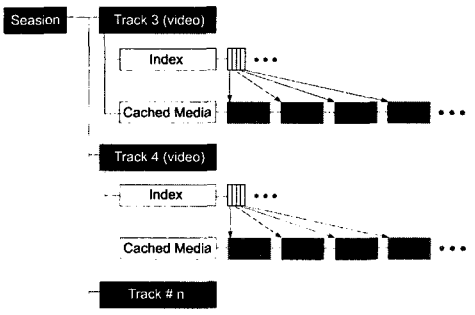


그림 5 패킷 미디어의 캐싱 구조

패킷 미디어는 RTP 패킷 캐싱시 seq, ts, 크기와 패킷 변위를 기록한 인덱스 값을 이용하여 전송한다. 그림 6은 캐쉬 서버에서 콘텐츠 전송을 위한 기본 구조를 나타낸 것으로 미디어 절대 시간과 시스템 시간을 더해 패킷 전송 시간을 결정하고 초기 전송 시간으로부터 현재 미디어 시간과 시스템 시간을 통해 패킷 전송과 지연을 결정한다. 패킷 전송은 시스템 시간을 기준으로 최저 전송 지연과 최고 전송 시간 안에 있는 RTP 패킷을 대상으로 한다. 전송시 RTP 패킷의 sequence number, time stamp, ssrc or csrc 등은 캐쉬 서버의 서비스 세션 정보를 반영하여 스트리밍 서버의 미디어 전송과 동일하게 구성된다. 패킷 드롭은 캐쉬 서버의 전송 지연으로 시스템 시간과 RTP 패킷 시간과의 차가 최저 전송 지연 시간을 넘는 RTP 패킷을 전송 대상에서 제외하는 것으로 RTP의 미디어 실시간 전송 구조를 고려한 것이다.

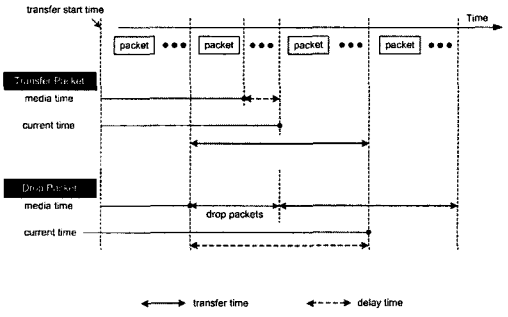


그림 6 패킷 미디어 전송 구조

3.3 RTCP 제어

RTCP 제어는 RTP 패킷 전송에 대한 세션 관리를 제공하며, 스트리밍 서버의 패킷 전송 상태와 클라이언트의 수신 상태를 주고 받는 양방향 제어 구조이다. 그림 7은 RTCP 제어 구조를 보여 준다. RTCP 제어 구

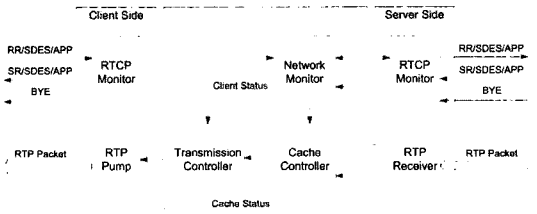


그림 7 RTSP 제어 구조

조는 클라이언트와 스트리밍 서버의 RTP/RTCP 세션 관리와 캐쉬 제어 구조를 제공한다. 클라이언트 세션 관리는 RTCP 제어를 통한 사용자 QoS 모니터링과 패킷 미디어의 전송을 담당하고, 스트리밍 서버 세션 관리는 RTCP 수신자 보고와 RTP 수신를 담당한다. 캐쉬 제어 구조는 전송 제어기, 네트워크 모니터, 캐쉬 제어기가 있다. 전송 제어기는 클라이언트의 수신 보고를 통해 패킷 손실과 전송 지연을 분석하고 캐쉬 미디어의 상태에 따라 전송 유지와 종료를 결정한다. 즉, 캐쉬 서버의 전송 정책은 캐싱된 미디어에 대해 전송 품질을 조절하지 않는 정적 정책이다. 네트워크 모니터는 RTCP 모니터로부터 클라이언트의 상태를 보고 받아 전송 제어기에 통보하고, RTP 수신자로부터 수신 상태를 보고받아 분석하여 스트리밍 서버와 캐쉬 제어기에 보고한다. 캐쉬 제어기는 미디어 전송과 수신에 대해 유효성을 검사하고, 미디어 캐싱에 대해 캐싱된 선행 미디어와 수신된 후행 미디어의 일관성을 유지한다. 캐쉬 미디어의 일관성은 수신된 RTP 패킷 헤더 정보를 절대 정보로 전환하여 선행 미디어에 이어 순차적으로 캐싱하고 캐싱 히트 상황에서 발생할 수 있는 중복된 부분을 제거하여 유지한다.

캐쉬 서버에서 RTCP 세션 관리 절차는 다음과 같다. RTSP 세션 설정 이후, 캐쉬 서버는 RTCP의 SR 메시지 전송을 통해 스트리밍의 시작과 RTP 패킷 전송 정보를 클라이언트로 주기적으로 보낸다. 클라이언트는 대역폭 정보와 세션 정보를 RTCP의 RR 메시지를 구성하여 응답한다. 또한, 스트리밍 서버의 SR에 대해서 클라이언트와 동일한 방식으로 RR을 보고한다. 클라이언트의 RTCP 세션은 RTP 스트리밍의 종료를 알리는 캐쉬 서버의 RTSP TEARDOWN이 발생할 때까지 세션을 유지한다. 캐쉬 서버의 내부에서는 클라이언트의 RR의 보고를 기반으로 클라이언트 대역폭을 유추하여 사용자 QoS를 감시하고 세션 유지를 결정한다. 캐쉬 서버는 사용자 QoS를 만족하지 못하는 경우 클라이언트 정책에 따라 세션 종료나 클라이언트 버퍼링을 허용한다. 캐쉬 서버의 RTCP 세션 종료는 BYE를 통해 알리고 RTP 전송을 중단한다. 스트리밍 서버의 RTCP 세션은 클라이언트 세션의 시작과 함께 캐쉬 서버의 RTSP DESCRIBE로 시작된다. 스트리밍 서버의 SR 보고 이후 캐쉬 서버는 네트워크 상태와 패킷 전송 상태를 검사해 RR로 응답한다. 캐쉬 서버는 캐싱의 비손실 정책에 따라 스트리밍 서버가 손실 정책을 사용하지 않도록 해야 한다. 만약 스트리밍 서버 정책이 손실 정책으로 전환된다면 세션 중단을 요구하고 재설정 과정을 시도한다. 캐쉬 서버가 지원하는 RTCP 메시지는 SR, RR, SDES, BYE, APP이다. 이중에서 SDES, APP, BYE는 독립적으로 혹은 SR, RR과 함께 이용한다. 다음은 RTCP의 세부적인 절차로 RTCP SR의 예이다.

```
(SR ssrc=0x552c p=0 count=0 len=6 ntp=985986124.3852585984 ts=18391
pset=1
osent=1373) (SDES p=0 count=1 len=6 (ssrc=0x552c
CNAME="QTSS983777135"))
```

SR 메시지는 스트리밍 서버의 전송 정보를 담는다. 내부 인자인 *ssrc*는 스트리밍 서버의 고유 식별자이며 RTSP에서 할당한 *sessionid*와 일치한다. *ntp*는 스트리밍 서버의 시간정보이고 *ts*는 가장 최근에 보낸 RTP 패킷의 *ts*값과 일치한다. *osent*는 스트리밍이 시작되어 전송된 RTP 패킷의 수를 나타낸다. SR과 함께 보내진 SDES는 스트리밍 서버의 고유 식별자에 해당되는 스트리밍 서버의 이름 정보를 담는다. 스트리밍 서버와 캐쉬 서버는 SR 메시지를 주기적으로 스트리밍이 종료될 때까지 보내며 세션 연결 상태를 유지하는 수단으로 사용된다. 다음은 RTCP RR의 예이다.

```
(RR ssrc=0x1190 p=0 count=0 len=7 (ssrc=0x552c fraction=0
lost=89128960 last_seq=21555 jit=0 lsr=409880225 dlsr=1015172431))
(SDES p=0 count=1 len=18 (ssrc=0x1190 CNAME="QTS 1929737000"
NAME="QuickTiem Streaming" TOOL="QuickTime Steaming
/5.0b11"))
(APP p=0 subtype=0 len=13 (ssrc=0x1190 NAME=QTSS SET=0
SEQNUM=21804))
```

RR 메시지는 스트리밍 서버를 위해 캐쉬 서버의 수신 정보를 담는다. 캐쉬 서버의 수신정보는 RR의 *fraction*, *lost*, *last_seq*, *jit*, *lsr*, *dlsr*로 표현된다. 다음은 RTCP BYE의 예이다.

```
SR ssrc=0x552c p=0 count=0 len=6 ntp=985986273.4084514218
ts=6001389 pset=682
osent=842995)
(SDES p=0 count=1 len=6 (ssrc=0x552c CNAME="QTSS983777135"))
(BYE p=0 count=1 len=1 (ssrc=0x552c))
```

BYE는 세션 *ssrc*에 대해서 미디어 전송의 종료를 의미한다. RTSP의 TEARDOWN과 RTCP의 BYE는 서로 비슷한 의미를 갖는다. TEARDOWN은 클라이언트에서 스트리밍 서버로 전송되고 DESCRIBE 단계부터 PLAY까지 설정된 모든 트랙를 종료한다. 반면에 BYE는 스트리밍 서버에서 클라이언트로 통보하며 명시된 세션에 대한 종료이다.

4. 스트림 Splitting 모듈

스트림 Splitting 모듈은 스트리밍 서버의 방송 스트림을 다수의 클라이언트 수신 그룹에 중계 전송 기능을 제공한다. 스트림 Splitting 모듈은 단일 방송 스트림에 대해 스트리밍 서버와 캐쉬 서버 사이에 중계 세션을 구성하고 수신 그룹에 대한 세션 관리, 전송 관리를 위해 RTSP 메시지 필터, RTCP 메시지 필터와 RTP/RTCP 중계 전송기로 구성된다. 그림 8은 스트림 Splitting 모듈을 보여준다.

RTSP 메시지 필터는 스트리밍 서버가 제공하는 방송 스트림에 대해 중계 세션을 구성하여 관리한다. 중계 세션은 첫 클라이언트 요구를 받아 스트리밍 서버와 캐쉬 서버 사이에 논리적인 스트림 전송 경로로 설정되고 다수의 클라이언트에 대한 전송 구조를 갖는다. 세션 정보는 클라이언트 요구 메시지와 스트리밍 서버의 RTSP 응답 메시지를 일대일 순서쌍으로 구성된 RTSP 메시지 맵으로 관리된다. 중계 세션 개설 이후, RTSP 메시지 필터는 스트리밍 서버에 상관없이 세션 제어를 통해 클라이언트의 세션 참여를 관리하고, 클라이언트가 세션에 없는 경우에 중계 세션을 종료한다. 방송 스트림에 대한 RTSP 메시지 구성은 On-Demand 서비스 구성과 동일하며, DESCRIBE, SETUP, PLAY, PAUSE, TEARDOWN의 기본 단계로 구성된다.

그림 9는 스트림 Splitting 서비스를 위한 RTSP/RTP 세션 설정과 데이타 전송 과정을 보여준다. DESCRIBE는 중계 세션 개설을 위한 스트리밍 서버와

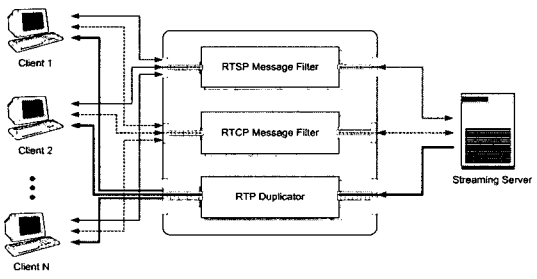


그림 8 스트림 Splitting 모듈 구조

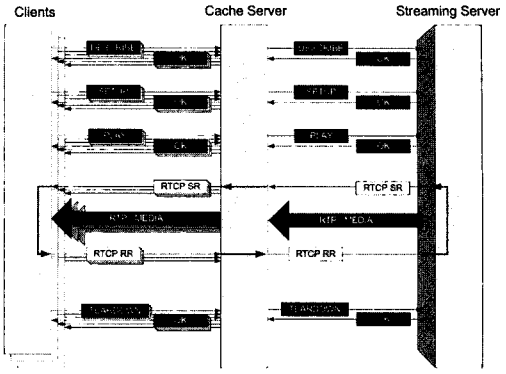


그림 9 스트림 Splitting을 위한 RTSP/RTP 세션과정

방송 스트림의 정보를 제공한다. 캐쉬 서버는 방송 스트림 리스트를 구성하며, 중계 세션이 개설되지 않은 상태에서 주기적으로 방송 스트림 리스트를 재구성한다. SETUP은 중계 세션 개설과 세션 참여 의사를 담는다. 캐쉬 서버는 방송 스트림에 대한 첫 클라이언트 요구를 받아 중계 세션을 개설하고 세션에 참여하는 클라이언트에 대해 자원을 할당한다. 또한, SETUP 요구에 제시된 프로토콜 정보를 참조하여 스트림 전송을 위한 포트를 설정한다. PLAY는 세션 시작과 RTP 패킷의 시작 정보를 담는다. 캐쉬 서버는 중계 세션에 전송된 RTP 패킷 정보를 PLAY의 trackID, seq, rtptime에 반영한다. PAUSE와 TEARDOWN은 중계 세션의 일시적인 중지와 탈퇴를 의미한다. 캐쉬 서버는 개별 클라이언트의 PAUSE와 TEARDOWN 요구에 대해 중계 세션에 반영되지 않도록 독립적으로 구성된다.

RTCP 메시지 필터는 중계 세션에 대한 RTCP 세션 제어를 제공한다. RTCP SR 메시지는 스트리밍 서버에서 캐쉬 서버로 전송되며 캐쉬 서버에서 다수의 클라이언트에 복사 전송된다. RTCP RR 메시지는 중계 세션 내에 클라이언트들의 수신 상태를 캐쉬 서버에서 취합하여 스트리밍 서버로 전송한다. 캐쉬 서버의 RR 메시지 구성은 대표 클라이언트 선택과 대표 메시지를 선택하는 방법을 고려할 수 있다. 대표 클라이언트는 중계 세션 내에 수신 상태가 가장 좋은 클라이언트로 선택되며 스트리밍 서버의 전송 품질을 결정하는 기준이 된다. 대표 클라이언트 선택은 임의의 탈퇴와 예외 상황 발생시 중계 세션을 중지시키는 상황이 발생할 수 있으며, 동적으로 클라이언트를 선택하여도 캐쉬 서버 부하와 전송 지연을 발생시킬 수 있다. 대표 메시지는 스트리밍 서버의 SR 메시지를 기준으로 캐쉬

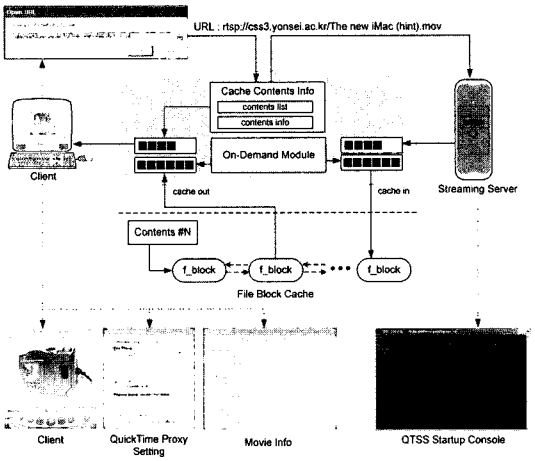
서버에 전송된 RR 메시지 중에서 수신 상태가 적절한 것을 선택한다. 캐쉬 서버는 SR 메시지를 전송하고 지정된 시간 간격 내에 RR 메시지를 수용하여 전송되는 순서에 따라 중계 세션의 수신 상태를 갱신한다. RTCP 메시지 필터는 대표 메시지 선택 방법을 적용하고 있으며 RR 메시지 선택과 캐쉬 서버의 RR 메시지 구성을 위한 투명성을 제공한다.

RTP 중계 전송기는 중계 세션으로 전송되는 RTP 패킷을 버퍼로 전송 받아 클라이언트 세션 리스트에 기술된 SSRC의 배정과 복사 전송을 반복해서 수행한다. 클라이언트는 RTSP PLAY의 trackID, seq, rtptime에서 기술한 RTP 패킷 정보를 기준으로 스트리밍을 시작하고, 이후에 전송된 패킷들은 상대적인 변위로 처리하기 때문에 RTP 중계 전송기에서 RTP 패킷 정보를 개별 클라이언트에 따라 재구성될 필요가 없다.

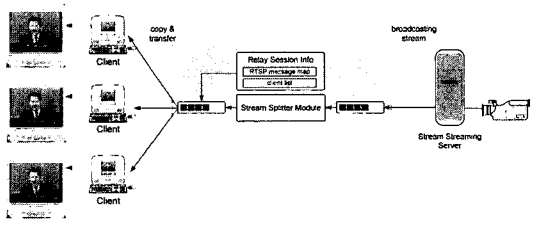
5. 구 현

스트리밍 프로토콜에 기반한 캐쉬 제어구조는 RTSP/RTP에 기반한 세션 제어, 패킷 미디어 캐싱, 캐쉬 미디어 스트리밍 기능들을 제공하여 On-Demand, 스트림 Splitting 서비스 모듈로써 구현되었다. 스트리밍 미디어 캐쉬 서버의 개발 환경은 다음과 같다. 호스트 서버는 Dual P-III 866 MHz에 512MB를 장착한 Compaq Proliant ML370이며, 리눅스 2.4에 기반한 SGI xFS[14]화일 시스템을 사용하였다. 구현 시스템과 연동한 스트리밍 서버는 Apple QTSS 3.0[15]이며, 클라이언트는 Quick Time Player Pro 5.0이고, 스트리밍 미디어 형식은 Apple에서 제공하는 Hinted Track화된 MOV이다.

그림 10은 구현된 스트리밍 미디어 캐쉬 서버의 On-Demand 캐싱과 스트림 Splitting 기능 구현에 따른 스트리밍 서비스 환경과 클라이언트의 재생 상황을 보여준다. 그림 10(a)에서 On-Demand 캐싱 실험은 클라이언트의 요구가 발생하면 콘텐츠 관리자를 통해 콘텐츠 리스트와 정보를 통해 콘텐츠의 유무를 확인한다. 캐쉬 미스인 경우에는 스트리밍 서버부터 콘텐츠에 대한 정보 확인하고 캐쉬 수용제어에 따라 서비스와 캐싱 여부를 결정한다. 캐쉬 히트인 경우, 스트리밍 서버로부터 콘텐츠 갱신여부를 확인하고 화일 블럭 캐쉬로부터 콘텐츠에 대한 세부정보를 얻어 클라이언트로의 스트림 전송과 캐싱되지 않은 나머지 부분에 대해 선반입을 수행한다. On-Demand 캐싱 실험은 선행 미디어가 캐싱된 상태에서 스트림에 대한 새로운 요구가 발생했을때, 선행 미디어와 후행 미디어의 연결을 통해



(a) On-Demand 모듈



(b) 스트림 Splitter 모듈

그림 10 캐쉬 서버 모듈 구현

클라이언트에서 비디오 장면과 오디오 음원의 중첩없이 재생됨을 확인할 수 있었다. 그림 10(b)은 스트림 Splitting을 보여준다. 스트림 Splitting 서비스는 클라이언트의 생중계 채널에 접속여부를 확인하고 생중계 서버의 방송 여부를 확인한다. 이후 스트림 Splitting 모듈은 중계 세션을 구성하고 클라이언트 리스트를 관리한다. 서버로부터 전송된 생중계 스트림은 중계 세션에 가입한 클라이언트를 대상으로 복사 전송 기능과 일대다의 세션 제어를 제공한다. 이와같은 과정에 따라 클라이언트들의 방송 스트림 재생이 성공적으로 수행됨을 확인하였다.

6. 결 론

본 논문에서는 RTSP/RTP 프로토콜에 기반한 멀티미디어 스트리밍 환경에서 효율적인 미디어 전송을 위한 스트리밍 미디어 캐쉬 서버의 핵심 요소를 설계하고 구현하였다. On-Demand 캐싱과 스트림 Splitting 서비스는 클라이언트의 재생 실험을 통해 검증하였다.

구현된 스트리밍 미디어 캐쉬 서버는 기존 스트리밍 캐싱 시스템과 구별되는 다음과 같은 특징을 갖는다. 첫째, 논문에서 구현한 스트리밍 미디어 캐싱 시스템은 표준 프로토콜에 기반한 기존 스트리밍 환경에 대해 완전한 호환성을 갖는다. 기존 스트리밍 환경은 멀티미디어 캐싱을 지원하지 않거나 개발 환경에 의존적인 독자적인 캐싱 환경을 제공한다. 이는 멀티미디어 캐싱 환경에 대한 호환성 제공을 어렵게 하며 각각의 스트리밍 환경에 대한 시스템 구축으로 제반 비용을 증가시키는 요인이 될 수 있다. 반면에 표준 프로토콜 기반의 미디어 캐싱은 미디어 코덱, 미디어 저장 방식 등에 독립적으로 적용할 수 있어 개발 기간 단축과 강한 호환성을 갖는다. 둘째, 구현한 시스템은 콘텐츠을 부분적으로 유지관리하는 동적 캐싱을 지원한다. 기존 CDN 환경에서의 스트리밍 미디어 캐싱은 콘텐츠 미러링 방식으로 완전한 콘텐츠를 복사하는 정적 캐싱 방식을 사용하고 있다. 이는 안정된 콘텐츠 서비스를 제공할 수 있는 장점이 있지만 콘텐츠 단위의 캐싱과 캐쉬 아웃으로 네트워크 트래픽을 증가시키는 요인이 될 수 있고, 효율적인 캐싱 정책에 제한요소가 될 수 있다. 제시한 동적 캐싱은 사용자의 콘텐츠 활용 경향을 반영하는 효율적인 캐싱 정책에 따라 부분적으로 콘텐츠를 관리할 수 있게 하며 선행 미디어와 후행 미디어의 일관성과 유지관리를 제공한다. 셋째, 구현한 스트리밍 캐쉬 시스템은 캐쉬 미디어를 재구성하지 않는 단순한 구조를 유지하여 시스템 부하를 줄일 수 있다. 기존 스트리밍 미디어 캐싱은 각 개발 환경에 맞는 미디어 저장 구조를 재구성하거나 스트리밍 서버의 원본 미디어와 동일한 형식으로 복원하는 구조를 갖는다. 이는 캐쉬 서버에 미디어 복원에 대한 부하를 가중시키며 그 구조를 복잡하게하는 단점을 갖는다. 이러한 패킷 단위 캐싱은 재구성 과정이 없으며 프로토콜에 명시된 시간과 패킷 순서를 활용한 단순 캐싱 구조를 제공한다. 또한 논문에서 구현한 시스템은 패킷 단위 기반의 스트리밍 구조를 제공하여, 미디어 종류에 상관없이 전송 프로토콜의 시간 정보를 활용하여 기존 스트리밍 서버와 동일한 서비스를 제공할 수 있게 한다. 넷째, 기존 상용화된 대부분의 스트리밍 캐싱 시스템들은 내장형 시스템으로 개발되어 있어 이식성이나 범용성이 낮은 반면에, 논문에서 구현한 스트리밍 미디어 캐싱 시스템은 패킷으로 구성된 고정 블럭 단위의 효율적인 입출 구조를 고려할 수 있어 범용 운영체제와 이종 컴퓨터 환경에 이식성이 뛰어난 범용 캐싱 모델을 구축할 수 있다.

향후 과제는 본 논문에서 제시한 프로토콜 수준의 스트리밍 미디어 캐싱 및 재전송 기술을 바탕으로, 효율적인 캐쉬 교체 정책과 캐쉬 입출력 시스템의 개발을 통해 통합된 스트리밍 미디어 캐싱 시스템의 개발에 있다.

참 고 문 헌

[1] Yeuwei Wang, Zhi-Li Zhang, David H.D. Du, and Dongli Su, "A Network Consious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers," *IEEE Infocom*, April 1998, pp.660-667.

[2] M. Abrams, C. Standridge, G. Abdulla, S. Williams, and E. Fox. "Caching Proxies: Limitations and Potentials," *Proceedings of 1995 World Wide Web Conference*, Boston, 1995, pp.119-133.

[3] R. Frederick, J. Geagan, M. Kellner, A. Periyannan, *Caching Support in RTSP/RTP Servers*, Draft-periyannan-rtsp-caching-01.txt, March 2000.

[4] Schulzrinne. H., *Real Time Streaming Protocol (RTSP)*, RFC 2326, April 1998.

[5] Schulzrinne. H., *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889, January 1996.

[6] *A Hierarchical Internet Object Cache*, URL: http://harvest.cs.colorado.edu.

[7] *Squid Web Proxy Cache*, URL: http://www.squid-cache.org.

[8] Asit Dan, Daniel M. Dias, Rajat Mukherejee, Dinkar Sitaram and Renu Tewari, "Buffering and Caching in Large-Scale Video Servers," *Proceedings of IEEE CompCon*, Los Alamitor, January 1995, pp.217-224.

[9] Asit Dan and Kinkar Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads," *Proceedings of IS&T SPIE Multimedia Computing and Networking Conference*, San Jose, January 1996, pp.344-351.

[10] Stephane Gruber, Jennifer Rexford and Andrea Basso, "Protocol considerations for a Prefix-Caching Proxy for Multimedia Streams," *Proceedings of WWW Conference*, May 2000, pp. 657-668.

[11] *Inktomi: Caching and Media Appliances*, URL: http://www.inktomi.com/products/cnsl.

[12] *NetAppliance : Streaming Media Solution*, URL: http://www.netapp.com/solutions/streaming_sol.html.

[13] *CacheFlow : cIQ™ Streaming Intranets*, URL : http://www.cacheflow.com/products/solutions/enterprise/streaming.cfm.

[14] *SGI's xFS File System for Linux*, URL: http://oss.sgi.com/projects/xfs.

[15] *QuickTime Streaming Server*, URL: http://www.apple.com/quicktime/products/qtss.

오 재 학

1997년 광운대학교 컴퓨터과학 학사 1999년 광운대학교 컴퓨터과학 석사 2002년~현재 광운대학교 컴퓨터과학과 박사과정 관심분야는 멀티미디어 시스템, 실시간 시스템, 대용량 저장장치

차 호 정

1985년 서울대학교 컴퓨터공학 학사 1987년 서울대학교 컴퓨터공학 석사 1991년 영국 University of Manchester 전산학 박사. 1993년~2001년 광운대학교 컴퓨터과학과 부교수. 2001년~현재 연세대학교 컴퓨터과학과 부교수. 관심분야는 멀티미디어 시스템, 운영체제, 분산 컴퓨팅

최 영 근

1980년 서울대학교 수학교육과(이학사) 1982년 서울대학교 계산통계학과(이학석사). 1989년 서울대학교 계산통계학과(이학박사). 1992년~현재 광운대학교 컴퓨터과학과 교수. 1992년~2000년 광운대학교 전산정보원 원장. 2002년~2002년 광운대학교 정보통신연구원장. 2002년~현재 광운대학교 교무연구처장. 관심분야는 프로그래밍 언어, 병렬 프로그래밍 언어, 객체지향 설계 및 분석, 분산컴퓨팅기술, Mobile agent