센서기기 연동형 동적 적응 콘텐트 플레이어의 설계 및 구현

이용주°,박춘서,김정근,정진환,민옥기,김학영
한국전자통신연구원 디지털홈연구단 인터넷서버그룹 미디어스트리밍연구팀
{yongju°,parkcs,kjjk0301,jhjeong,ogmin,h0kim }@etri.re.kr

Design and Implementation of Dynamic Adaptive Contents Player using Sensor-Device

Yong-Ju Lee°, Choon-Seo Park, Jung-Keun Kim, Jin-Hwan Jeong, Ok-Gee Min, Hag-Young Kim
Media Streaming Research Team, Internet Server Technology Group, Digital Home Research Division, Electronic and Telecommunication Research Institute

요 약

최근 들어 인터넷, 인트라넷을 통해 양질의 동영상 데이터를 네트워크 상으로 실시간 또는 사용자의 요구에 따라 전송해 주는 스트리밍 서비스에 대한 요구가 증가하고 있다. 아울러, 유비쿼터스 환경의 도래와 함께 센싱을 통한 다양한 멀티미디어 서비스의 확장이 이루어지고 있으며, 유/무선 환경에서 사용자의 콘텐트에 대한 요구를 끊김 없이/연속 이동을 통해 볼 수 있는 사용자 맞춤형 서비스가 필요로 되고 있다. 이에 본 논문에서는 센서기기를 연동해서 사용자를 인식하고 양질의 동영상을 사용자의 요구 시점에 이동성을 보장하면서 사용자의 다양한 단말 조건을 인식하여 멀티미디어 서비스를 제공하는 내용을 담고 있다.

1. 서론

멀티미디어 스트리밍 기술은 다수의 사용자의요구 패턴과 이에 따른 다양한 콘텐트 포맷, 스트리밍을 위한 네트워크 환경과 밀접하게 발전해 왔으며, 사용자의 요구를 최대한 수용하기 위한 맞춤형 서비스 형태로 진보하고 있다. 이에 사용자의 요구를 센서로 인식하여, 콘텐트 정보를 자동으로 감지하고, 사용자가 보던 시점의 위치부터 서비스를 제공하는 쪽으로 서비스의 형태가 바뀌고 있다. 하지만 사용자의 요구에 물리적인 단말 장치의 능력에는 한계가 있으며, 이를 동적으로 적응하기 위한 사용자 단말 정보, 네트워크 상황 정보를 가진 적응형 콘텐트 플레이어에 대한 연구를 필요로 하며, 본 논문에서는 센서기기 연동형 동적 적응형 콘텐트 플레이어의 설계 및 구현을 다룬다. 2 장에서는 적응형 콘텐트 전송을 위한 서버/클라이언트 상에서의 연구 방향에 대하여 기술한다. 제 3 장에서 앞서 개발이 이루어지고 있는 오픈소스인 VLC 플레이어를 확장하여, 센서 연동모듈, 네트워크상황 인지 모듈, 사용자 대 콘텐트 변환 모듈을 추가하여 플레이어를 확장하는 부분을 다루고, 4,5 장에서는 이에 따른 플레이어 상에서의 스트림 분석을 통한 jitter 측정과 스트리밍 서버와의 RTSP 컨트롤 추가 부분과 구현 내용을 담고 있다. 마지막으로 6 장에서는 이에 따른 결론을 맺는다[1][2][3].

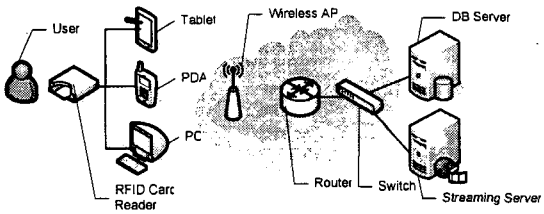
2. 관련 연구

사용자 단말의 다양화/요구의 증가로 일반적인 미디어 스트리밍 방법에서도 다양한 변화가 요구된다. 기존의 스트리밍 연구의 방향은 크게 전송 측에서의 컨트롤과 전송 량의 변화추이를 전용 프로토콜인 RTP/RTCP 를 통해 보완하는 방법들과, 효율적인 전송을 위한 여러 가지 프록시/버퍼/캐싱 알고리즘에 대한 연구, 유/무선 /P2P 등과 같은 여러 망과 망 특성을 고려한 콘텐트 스트리밍 및 분산/배포 방법들이 주를 이루고 있다. 하지만 인터넷 망 자체가 Best-Effort 망이고, 단말의 다양화를 피할 수 없기 때문에 점점 콘텐트의 특성을 고려한 버전(Version)/레이어(Layer)에 주로 연구 방향으로 제시되고 있으며, 실시간 적응을 위한 트랜스코딩/스케이러블코딩과 같은 MPEG 자체의 특징 부여 측면이 강해지고 있다. 또한 유비쿼터스가 화두로 제시되면서 다양한 단말에 혼재된 환경 속에서의 센서네트워크 상으로 융합되면서, 적응형 콘텐트 전송이 여러 분야에서 연구되고 있다.

3. 플레이어 확장 설계

(그림 1)에서 사용자의 자신의 RFID tag 를 가지고 있으며, 각각의 RFID tag 는 고유한 tag ID 를 가지고 있다. RFID card reader 를 사용자 단말은 tagID 를 인식하고 이를 DB/Event 서버에 접근을 통해 사용자 ID 로 변환된다. 변환된 사용자 ID 와 기본적인 단말의 CPU/Display 정보 등을 추출하여 Streaming 서버에 컨트롤 전송 프로토콜인 RTSP 를 통해 전송된다. 이때 초기 OPTIONS 메시지를 통해 URL/USER_ID 로 커넥션을 맺게 되며 이를 통해 스트리밍 서버 앞 단에서 USER_ID 를 가지고 기존의 보던 콘텐트를 연속 이어보기가 가능하게 된다. 스트리밍 서버에는 하나의 콘텐트에 대해 여러 비트레이트를 가진 버전에 존재하며, 이 콘텐트 이름과 비트레이트

정보를 초기 OPTIONS 에 응답 메시지를 전송하게 된다. 이를 가지고 사용자 단말에서는 초기 네트워크 부하 량을 측정하기 위해 가장 상위보다 UP/DOWN 방식으로 서버에 부가 패킷을 요구하게 된다. 이후 OPTIONS 에서는 이를 통해 적절한 대역폭양을 측정하여, 적절한 콘텐트 ID 를 통해 스트리밍 서버와 접속하여 서비스를 받게 된다. 이후에 단말의 네트웍 대역폭의 변화를 측정하기 위한 단말 플레이어에서 타이머 쓰레드가 실행되어 주기적으로 대역폭을 측정하게 된다. 이를 통해 서비스 도중에서 네트워크 대역폭의 변화를 감지하여 끊김없는 이어보기가 가능하게 된다.



(그림 1) 스트리밍 서버/ 동적 적응형 플레이어의 전체구조

4. 스트림 분석과 센서기기 연동

플레이어 단에서의 스트림 분석 과정은 MPEG-2 TS 의 경우 스트림되어지는 TS Packet 의 손실(TS Discontinuity) 또는 지연(Deferred PTS)으로 패킷 단위의 분석과 Demuxing 을 거친 Video/Audio 스트림 개별 패킷에서의 디코딩 시간을 감안한 분석으로 나누어 볼 수 있다.

스트림 패킷 분석과정에서는 가령 유선인 경우에는 손실보다는 지연되는 경우가 발생 빈도가 높으므로 PTS 의 값을 통해 어느 정도의 예측이 이루어진다. 지연의 경우 Picture 을 도달 시점을 기준으로 display 되어야 할 시간이 현재 시간에 rendering 되는 시간을 더한 값보다 작으면 skipping 이 이루어진다. 무선인 경우에는 패킷 손실로 볼 연속패킷의 양을 가지고 적응의 기본 측정 단위로 삼는다.

손실과 지연으로 filtering 되지 못하는 경우는 video 와 audio 로 개별적으로 측정이 이루어진다. Video 의 경우, 새로운 IPicture 가 발생하는 빈도의 일정 임계 값을 주기적으로 체크하여 임계 값에 도달할 시점에 버려지는 픽쳐의 개수(i_trashed_pic)와 디코딩 되는 (i_pic)의 백분율을 통해 일정 기간 동안 trashed 되는 양이 증가하면 이 시점을 적응시점으로 여긴다. Audio 의 경우, 오디오 Output 의 최소 준비시간(AOUT_MIX_PREPARE_TIME)과 현재 시간의 더한 값이 버퍼의 시작 시간 보다 작은 경우 PTS 를 만족하지 못하므로 버퍼 Dropping 이 발생하게 된다. 이 경우 연속된 dropping buffer 의 카운터를 가지고 Audio 의 끊김이 발생할 시점에 적응시점으로 여긴다.

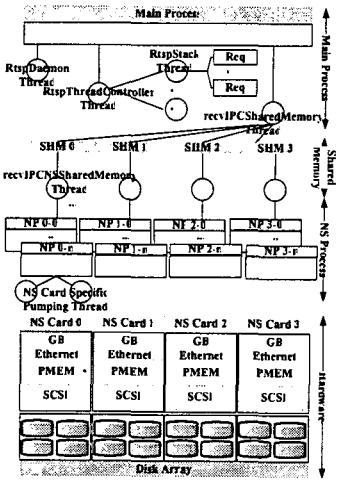
스트림 분석을 통해 얻은 각각의 데이터를 수집하기 위해서는 콘텐트 재생 동안에 MonitoringThread 가 Invoke 되어 일정 주기마다 각각의 데이터를 수집하여 스트림되는 양을 측정해야만 한다. 측정되는 값은 IDecodingPictureRate, AudioDecodingDropBufferCounter, TSDiscontinuityCounter 로 수집되며 각각의 값의 임계 치를 두어 판단의 근거로 삼는다. 가령 현재 서버에 한 콘텐트에 대해 3 가지의 버전이 존재하며 각각의 버전은 동일 콘텐트이지만 비트레이트가 다른 콘텐트가 존재한다면, 초기 RTSP 의 OPTIONS 로 콘텐트의 ID 와 비트레이트 정보를 받아온다. 이를 가지고 현재 가장 상위 비트레이트의 콘텐트가 재생되고 있다면, 측정한 값들이 임계 값 이상을 유지하고 있다면, 현재 재생을 그대로 수행하며, 임계 값 이하로 떨어지면 상위보다 낮은 버전으로 콘텐트를 바로 교체하게 된다. 교체 시점에서 바로 아래와 그 다음 아래 버전으로 내려갈 수 있는 여지가 있으며, 이 경우에는 단계적으로 Downstream 하는 방식을 채택하였다. 중간 버전의 콘텐트가 재생되고 있는 경우에는 현재 상태를 유지하는 경우/ 하위버전으로 내려가야 할 경우/ 상위버전으로 올라가야 할 경우의 3 가지 경우가 존재한다. 현재상태의 유지하는 경우와 내려가야 할 경우는 임계 값을 통해 판단을 할 수 있지만, 상위 버전으로 올라가야 할 경우는 추가적인 부가 패킷을 새 소켓으로 맺고 패킷을 보내보고 Upstream 을 해야 한다. 최하위 버전으로 내려간 경우에는 상위버전으로 올라가야 할 경우와 최하위버전을 유지하는 경우/ 최하위버전도 플레이 가능하지 않은 경우를 들 수 있다. 최하위버전을 유지하는 경우와 상위버전을 올려야 하는 경우는 중간버전의 판단과 동일한 방법으로 결정된다. 여러 버전을 가지고 동적 적응 플레이를 하는 경우에 각각의 버전 사이의 비트레이트 간격과 간격 사이에 벌어진 적응/ 가장 하위버전으로 플레이가 불가능하게 망 상황이 악화된 경에는 준비된 콘텐트로 안 되는 경우에는 서버상황에 실시간 트랜스코딩을 통해 해결할 수 있지만, 본 논문에서는 클라이언트에서의 적응만을 다룬다.

5. VLC 플레이어/스트리밍 서버 확장 구현

VLC 플레이어는 cross-platform 에서 호환되는 멀티미디어 플레이어로 오픈 소스로 널리 알려져 있다. VLC 소스상에서의 구현은 크게 Interface 부분의 Timer 모듈, VLC 에 플러그인 되는 Streaming 전용 모듈인 livedotcom 모듈, 부가패킷을 통해 네트워크 대역폭을 측정하는 Network 모듈, 스트림 분석을 수행하는 Analyzer 모듈, 분석된 결과를 가지고 판단이 이루어지는 Decision 모듈이 추가 구현되었다. Timer 모듈은 크게 ShortTimer/MidTimer/LongTimer 로 구분되며 각각의 시간 간격을 혼재/분산 함으로써 보다 적응적인 분석이 가능하게 한다. 이 모듈은 스트림 분석 시점을 항상 Invoke 해주는 모듈이다. 새로운 콘텐트로의 변환을 위해 스트리밍 서버에 콘텐트 정보를 전송하기 위해 기존의 livedotcom 모듈에서는 OPTIONS 에 추가 파라미터를 내장하였고, 네트워크 대역폭을 측정하는 Network 모듈은 OPTIONS 컨트롤을 통해 서버상에서 특정 포트로 소켓을 생성하여 부가 패킷을 주고 받는다. 스트림 분석 모듈은 앞서 4 장에서 기술한 내용과 같으며, 이 모든 모듈을 상위 클래스인 Decision 모듈에서 동적 적응이 가능하게 된다[4][5][6].

스트리밍 서버의 전체 구조는 그림 2 와 같다. 스트리밍 컨트롤 전송 프로토콜인 RTSP 를 기반으로 다양한 사용자 커넥션을 관리, 분배하는 MAIN 프로세스와 공유메모리를 사용하여 내부 IPC 메시지를 통해 내부 컨트롤을 생성하여 실제 동영상 서비스를 수행하는 NS 프로세스로 크게 나뉘어진다[4][5]. (그림 2)에서 메인 프로세스에서 생성된 RtspDaemon 쓰레드가 RtspThreadController 쓰레드를 생성한다. 이 쓰레드가 일정량의 커넥션을 관리 하는 RtspStackThread 를 관리한다. 사용자의 커넥션 개수가 증가하면 새로운 RtspStackThread 가 생성되어 추가된 커넥션 들을 관리하게 된다. 기본값으로 50 개의 커넥션을 RtspStackThread 가 관리하는 구조이다.

유메모리 영역을 확보하여 NS 프로세스와 IPC 를 통해 컨트롤을 주고 받는다[7][8].



(그림 2) 미디어 스트리밍 전체 구조

NS 프로세스에는 건네 받은 컨트롤을 수집하는 recvIPCNSSharedMemoryThread 가 존재하며, 이 NS 프로세스 역시 설정 시에 여러 개의 프로세스로 나뉘어지며, 아울러 개별 NS 프로세스에서 한 서비스를 담당하는 쓰레드를 생성 및 운용한다. 스트리밍 가속 하드웨어는 GB, PMEM, SCSI 컨트롤러 기능을 가지고 있으며 디스크 어레이를 하나의 스토라이핑된 마운트 포인트로 제공한다.

이러한 스트리밍 서버에서 추가 구현된 OPTIONS 에 들어가는 파라미터는 아래와 같다. MagicID 는 OPTIONS 는 세션 이전에 사용되는 컨트롤 이므로 고유한 ID 를 주고 받아야 하며, Action 은 실제 수행하는 일을 나타내며 이 경우에는 새로운 일이 할당된 것이다. 아래의 예에서는 1M 의 패킷을 2 초 후에 5 초간 UDP 로 클라이언트 포트에 보내달라는 요청의 예이다.

```
C->S: OPTIONS rtsp://foo.com/foo.mpg RTSP/1.0
      MagicID: 192.168.0.150-1234
      Action: New
      Protocol: UDP
      Client Port: 3456-3457
      DefferedTime:2
      DurationTime:5
      Bitrate:10485760
```

아울러, 플레이어상에서 스트림의 끊김 없는 재생을 위해서는 기존의 세션을 유지하면서 콘텐트에 대한 적응이 필수적이다. 이를 위해 앞서 설명한 RTSP 컨트롤을 통해 새로운 콘텐트의 플레이 요구가 오면 새 콘텐트의 기존 콘텐트의 NPT 값을 통해 적절한 위치로 이동하여 스트리밍을 연속 재생한다. 여기서 기존의 콘텐트와 다른 Pid, Bitrate 정보 등을 가지고 있으므로 사전에 asset 인스톨 단계에서 개별 콘텐트에 대한 PAT, PMT, A/V 관련 PES header 를 만들어 놓고 스트리밍 변환 시에 새 콘텐트에 인식 파일을 먼저 보내고 A/V 패킷 데이터를 보내게 된다. 이를 통해 끊김 없는 데이터 전송과 플레이가 가능하지만 기존 플레이어에 남아 있는 디코더 데이터의 디코딩과정과 콘텐트 변환과정에서 약간의 jitter 는 발생하게 된다. 하지만 새로운 콘텐트를 개능하게 된다.

센서는 기본적으로 리더와 Tag 으로 구분되며, Tag 은 TagID 로 사용자의 구분을 할 수 있게 된다. TagID 는 사용자에게 주어진 고유한 Identifier 이며, 사용자의 등록/관리는 센서기기 측에서 많은 연구와 구현이 이루어져 있으므로 본 논문에서는 임의의 사용자가 임의의 Tag 을 가지고 있으며, 데이터베이스 등록된 사용자의 접근을 가정한다. 리더 프로그램은 리더와의 연결 및 종료, 데이터 읽기를 처리하는 클래스, 리더 어플리케이션의 화면을 그리며 리더와의 연결 수행 및 Tag 정보를 얻기 위해 해당 명령어를 수행하며, 이벤트 서버로 데이터를 전송하기 위한 네트워크 모듈로 나뉘어진다. Tag 를 통해 인식된 Tag Data 는 데이터베이스 서버에 접속되어 사용자로 변환된다. 이 사용자 정보는 기존의 동영상을 보고, 어느 지점까지 본 것에 대한 정보와 링크되어 있으며, 클라이언트는 이 정보를 VLC 플레이어 상에서 Event Server 와의 연동으로 가지게 된다. 클라이언트 정보는 다시 스트리밍 서버에 기존의 스트리밍 컨트롤 프로토콜인 RTSP 를 통해 스트리밍 서버에 전송되고 스트리밍 서버는 콘텐트 재생 전에 콘텐트의 ID 정보, 비트레이트 정보 등을 서버로부터 건네 받게 된다. 이 정보는 앞서 얘기한 스트림 분석과정에 입력으로 사용된다.

6. 결론 및 향후 연구

본 논문에서는 센서 연동형 적응형 콘텐트 전송을 위한 클라이언트 상에서의 동적 콘텐트 재생을 가능하게 하기 위한 설계 및 구현을 담고 있다. 논문에서는 적응형 콘텐트 전송을 위해 크게 센서기기 연동 부분/스트리밍 서버의 추가 컨트롤 구현 부분/오픈 소스인 VLC 플레이어 상에서의 4 가지 모듈의 추가를 통해 끊김 없는 동영상 재생이 가능하게 되며, 네트워크 부하에 따른 영향을 최소화할 수 있는 방법을 제안했다.

참고문헌

[1] H. Schulzrinne, R. Lanphier, and A. Rao, "Real time streaming protocol (RTSP)," RFC 2326, Internet Engineering Task Force, Apr. 1998

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "RTP: a transport protocol for real-time applications". RFC 1889, Internet Engineering Task Force, Jan. 1996.

[3] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control," RFC 1809, Internet Engineering Task Force, Jan. 1996.

[4] Y.J. Lee, O.G. Min, S.J. Mun, H.Y. Kim, "Enabling High *Performance Media Streaming Server on Network Storage* Card", Internet and Multimedia Systems and Applications, IASTED, Aug, 2004

[5] O.G. Min, H.Y. Kim, T.G. Kwon, "A Mechanism for Improving Streaming Throughput on the NGIS System", Internet and Multimedia Systems and Applications, IASTED, Aug, 2004

[6] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha. "Streaming Video over the Internet: Approaches and Directions". IEEE Transactions on Circuits and Systems for Video Technology, Mar. 2001

[7] T.S. Chua, J. Li, B.C. Ooi, and K. Tan. "Disk striping strategies for large video-on-demand servers". In ACM Int. Multimedia Conference, pages 297–306, 1996.

[8] M Chesire, A Wolman, G M Voelker, and H M Levy. "Measurement and Analysis of a Streaming-Media Workload". In USITS, 2001.