Journal of Korea Multimedia Society Vol. 21, No. 8, August 2018(pp. 942-951)
https://doi.org/10.9717/kmms.2018.21.8.942

MPEG DASH SRD기반 다중 레벨 분할 영상 캐싱을 이용한 고품질 360 VR 영상 저지연 라이브 스트리밍 시스템 구현

김현욱[†], 최우성[††], 양성현[†††]

Implementation of High Quality 360 VR Video Low-latency Live Streaming System using Multi Level Tile Caching based on MPEG DASH SRD

Hyun Wook Kim[†], U Sung Choi[††], Sung Hyun Yang[†††]

ABSTRACT

In these days, 360 degree videos, which is provided via VR, have high resolution and quality of 8K. These kinds of videos inevitably require streaming technology which guarantees QoS. Therefore, we suggest MPEG DASH HTTP protocol which stably provides streaming services of high quality videos in 8K, which are 360 degree tile-encoded, on low-spec devices such as OTT and IPTV Settop and Tile Segment Cache management structure for servers operating streaming services.

Key words: Adaptive Streaming, MPEG-DASH, MPEG-DASH SRD, Tile Segment Cache, 360 VR, Tiled Video Streaming

1. 서 론

오늘날, 멀티미디어 플랫폼 기술은 4K급 이상의 고화질 미디어 서비스를 제공할 수 있을 정도로 발전했다. 최근에는 360도 동영상과 가상현실(VR, Virtual Reality)과 같은 새로운 동영상 서비스로 대중적으로 자리 잡고 있다. 하지만 아직까지 가상현실 기술을 통해 사용자에게 제공되고 있는 360도 동영상들의 서비스 품질이 만족스럽지 못한 측면이 있다[1]. 그 이유는 서비스되는 360도 비디오들은 8K(SUHD)급 이상의 고해상도와 높은 비트레이트(Bitrate)를 가지기 때문에, 이를 안정적으로 서비스할 수 있도록 QoS (Quality of Service)를 보장하는 스트리밍 기술이 해결되지 않고 있기 때문이다[2]. 이를 해결하기 위해서 최근에는 HEVC(High Efficiency Video Coding, H.265) 등의 초고효율 분할 압축과, MPEG DASH (Dynamic Adaptive Streaming over HTTP) SRD (Spatial Relationship Description)[3]과 같이 공간 정보를 활용할 수 있는 분할 영상 스트리밍(Tiled Video Streaming) 기술을 활용하여 사용자의 관심 영역(FoV: Field of View) 정보를 가지고 보여지는 관심 영역은 원본 화질의 영상 Tile을, 나머지는 저화

※ Corresponding Author : Hyun Wook, Kim, Address: (01897) Kwangwoon-ro 20, Nowon-gu, Seoul, Korea, TEL : +82-2-940-5751, FAX : +82-2-909-5715, E-mail : khw@kw.ac.kr
Receipt date : Jul. 13, 2018, Approval date : Jul. 25, 2018
[†] Dept. of Electronic Eng., Graduate School, Kwangwoon University
[††] Yoolicnt Co., Ldt. (E-mail : wkutt@naver.com)
[†††] Dept. of Electronic Eng., Graduate School, Kwangwoon University (E-mail : shyang@kw.ac.kr)
※ This research is supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2017 (R2017030018)

질 또는 저품질의 영상 Tile을 스트리밍하는 Adaptive Streaming 기술들이 많이 연구되어지고 있다[4,5,6]. 하지만 VR HMD(Head Mount Display)등과 같은 디바이스를 사용하여 360도 비디오를 시청할 경우 계속적인 화면 공간의 전환이 이루어지는데, 이에 따라 다시 관심영역을 추출하고 새로운 원본 화질의 Tile을 요청하고 수신하여 다시 HMD 디바이스에 렌더링을 하게 되는데, 이때 전환 지연이 발생하게 된다[1].

본 논문에서는 전환 지연 시간을 줄이기 위해 MPEG DASH SRD 프로토콜을 확장하여 클라이언트에서 FoV 정보를 스트리밍 서버에 전달할 수 있도록 클라이언트 플레이어를 구현하고, 프로토콜을 확장하였다. 그리고 이를 기반으로 서버에서는 미리 Tile Set을 그룹핑(Grouping)하여 캐싱(Caching)할 수 있도록 하고, 클라이언트에서 요청한 Tile Segment를 메모리 탐색하여 바로 응답할 수 있도록 하여, 지연 시간을 줄이고 이를 통한 전환 지연을 단축을 유도하였다. 그리고 테스트를 통해 성능 개선을 확인하였다.

본 논문의 구성은 2장에서 사전연구로 HTTP 기반의 Adaptive Streaming기술인 MPEG DASH SRD에 대해 알아본다. 그리고 3장에서 중앙 타일 정보를 기반으로 주변 타일에 대한 메모리 캐싱 기술을 제안하고, 이를 위한 DASH 스트리밍 프로토콜을 수정하고 적용한 서버와 플레이어를 설계하였다. 4장에서는 3장에서 설계한 플레이어와 서버로 시스템을 구현하고, 테스트를 통해 서버의 응답속도를 측정하였고, 성능이 개선됨을 확인하였다. 그리고 5장에서 결론을 맺는다.

2. 관련 연구

3.1 MPEG-DASH SRD

MPEG-DASH SRD 기술은 기존 네트워크에 Adaptive한 Streaming 기술인 MPEG-DASH에서 영상의 공간 정보를 활용하여 확장한 표준기술로, 저품질, 고품질의 분할 영상 Tile을 공간에 따라 혼용 배치하여 사용이 가능하다. 예를 들어 사람이 관심있게 보는 영역인 ROI(Region of interesting)에 해당하는 Tile은 고품질로 인코딩된 Tile로, 나머지 영역은 저품질로 Tile로 Stitching된 영상을 재생할 수 있다. 또한 360 VR 영상을 TV등으로 재생 할 때, 실제 보여지는 영역의 고품질 Tile만 Stitching하여 재생할 수도 있다. 따라서, 사람에게는 고품질의 비디오 서비스르 제공하면서, 네트워크 뿐만 아니라, 디코더(Decoder), 렌더러(Renderer)의 효율을 높일 수 있는 기술로 주목 받고 있다.

하지만 기존 MPEG-DASH SRD 연구[3][7]에서는 품질이 다른 분할(Title) 기법을 제시하되 스트리밍 기법에 대한 고찰이 충분히 이루어지지 않았다. 이에 논문에서는 MPEG-DASH SRD 스트리밍 기술의 성능 개선을 위한 고찰을 수행하고, 이를 위한 스트리밍 기법을 구체적으로 제시하고 구현하였다.

3. 본 론

이번 장에서는 MPEG DASH 스트리밍 서버의 응답 지연 시간 단축을 위해, HMD Tracking 정보를

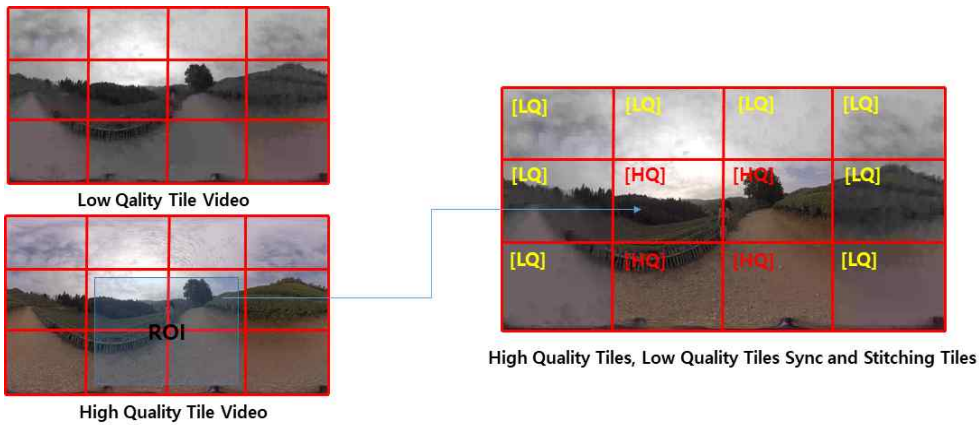

Fig. 1. MPEG DASH SRD Spatial Partitioning and Tiling for Foveated Rendering.

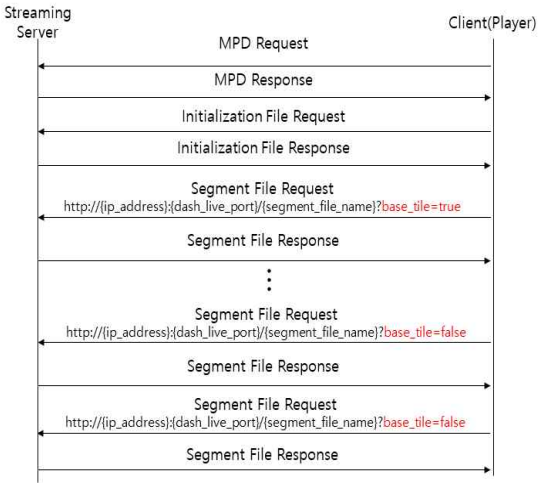

Fig. 2. MPEG DASH based Clinet Server Protocol Sequence Diagram.

이용하여 고화질 Tile Segment와 저화질 Tile Segment들을 예측하고, 미리 메모리 버퍼에 캐싱(Caching)하여, 클라이언트의 요청(Request) 시 메모리에서 탐색(Searching)하여 클라이언트로 전달하여 응답 시간을 단축하는 방법을 제안한다.

3.1 스트리밍 프로토콜 설계

스트리밍 프로토콜은 아래 Fig. 2와 같이 MPEG DASH에서 정의된 것처럼 HTTP 1.1 GET 프로토콜을 사용하여 클라이언트에서 스트리밍 서버로 MPD, Initialization 파일을 요청하여 수신한다. 그리고 공간과 시간 단위로 분할된 Tile Segment 파일들을 필요에 따라 요청한다. 이때, 클라이언트에서는 Tile Segment를 요청하는 URI에 현재 요청하는 Tile Segment가 HMD에 렌더링(Rendering)하는 중앙에 위치하는 Center Tile을 명시하는 HTTP GET Parameter로 'center_tile=true'를 추가하고, 전달하여 서버가 해당 Center Tile 정보를 수신하고, 이를 이용하여 미리 Tile Segment 파일들을 메모리 버퍼에 캐싱 할 수 있도록 하였다.

클라이언트에서는 서버에서 Center Tile Segment 요청을 우선 수신하여 요청 예측 Tile Segment들을 캐싱 할 수 있도록, 아래 Fig. 3과 같이 Center Tile Segment를 요청하고 수신하는 Thread를 항상 우선 생성 및 실행할 수 있도록 하였다.

위와 같이 클라이언트에서는 Tile Segment 요청 URL 생성 후 중앙에 해당하는 Center Tile Segment 정보를 GET 1.1 파라미터로 추가하고, 독립적인 쓰레드로 생성하여 우선 서버로 HTTP Request가 전달되도록 설계 및 구현하였다. 그리고 나머지 Tile Segment를 요청하고 처리하는 쓰레드들은 쓰레드 큐에 적재하였다가, Center Tile Segment 처리 쓰레드에서 Request를 전달 한 후 실행되며, 그 이후의 Tile Segment 요청 처리 쓰레드 들은 바로 실행이 된다.

HMD의 Tracking 좌표에 따른 중앙 Tile 계산 방

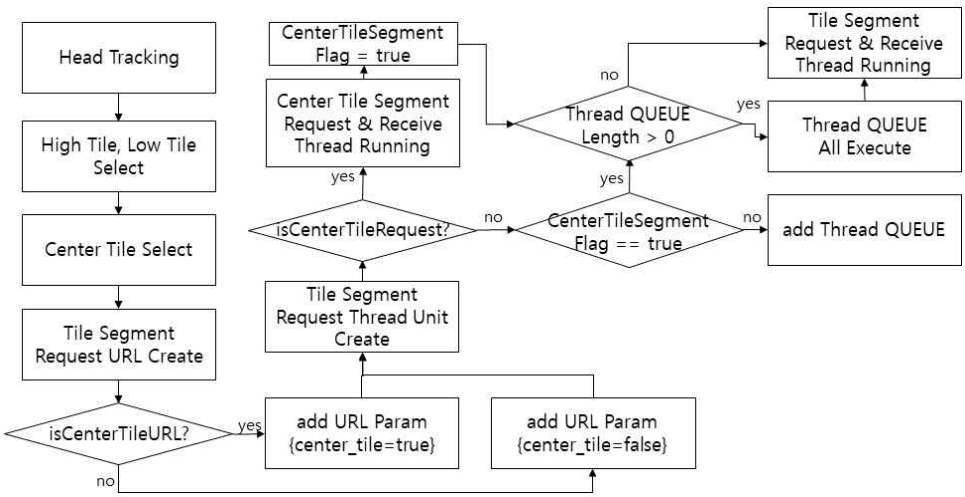

Fig. 3. Client Streaming Request Process.

법은 다음과 같다. HMD의 Head Tracking 좌표 값 또는 Eye Tracking 좌표를 구하는 방법은 기존 연구 [7]에서 상세히 기술하였으며, 본 논문에서는 이를 이용한 Tile을 선택하기 위한 추가적인 응용 연구를 수행하였다. 예를 들어 아래 Fig. 4와 같이 7680x3840 해상도를 가지고 6×6으로 분할된 영상의 x, y 좌표 값이 나타내는 Tile을 구하는 방법은 아래 식 1과 같다.

$$row = \begin{cases} 6 & \text{if } (y = 3840) \\ \left\lfloor 1 \le \frac{y}{640} + 1 < 7 \right\rfloor & \text{if } (0 \le y < 3840) \end{cases}$$
$$col = \begin{cases} 6 & \text{if } (x = 7680) \\ \left\lfloor 1 \le \frac{x}{1280} + 1 < 7 \right\rfloor & \text{if } (0 \le x < 7680) \end{cases} \quad (1)$$

예를 들어 x좌표가 4100이고, y좌표가 2000일 경우 좌표값이 포함된 Tile은 $\left(\left\lfloor \frac{4100}{1280}+1 \right\rfloor, \left\lfloor \frac{2000}{640}+1 \right\rfloor\right)$로 $(3+1,3+1)$이며 소숫점 제외 정수로 표현하여 $(4,4)$의 성분을 나타낸다.

3.2 저지연 라이브 스트리밍 서버 설계

본 절에서는 저지연 라이브 스트리밍 서버 구조를 설계하고, DASH 수신부의 프로세스를 설계하고 설명하였다. 그리고 Tile Segment 요청에 대한 빠른 응답을 위해 캐시 메모리 구조와, 캐시 적재 및 탐색 관리 방법을 제안하고, 설계하였다.

저지연 라이브 스트리밍 서버는 실시간으로 시간 단위로 분할되어 수신되는 영상을 수신하고, 수신된 영상을 다시 Dashing하여, 이를 파일 또는 메모리상에서 관리하고, 클라이언트의 요청에 따라 Dashing된 MPD 또는 Initialization, Segment Tile을 제공하는 역할을 수행한다. 전체적인 스트리밍 서버 구조는 Fig. 5와 같으며, 필요 기능에 따라 독립적인 모듈로 동작할 수 있도록 설계 하였다.

Fig. 4. 8K 6x6 Tile Video Matrix Image.

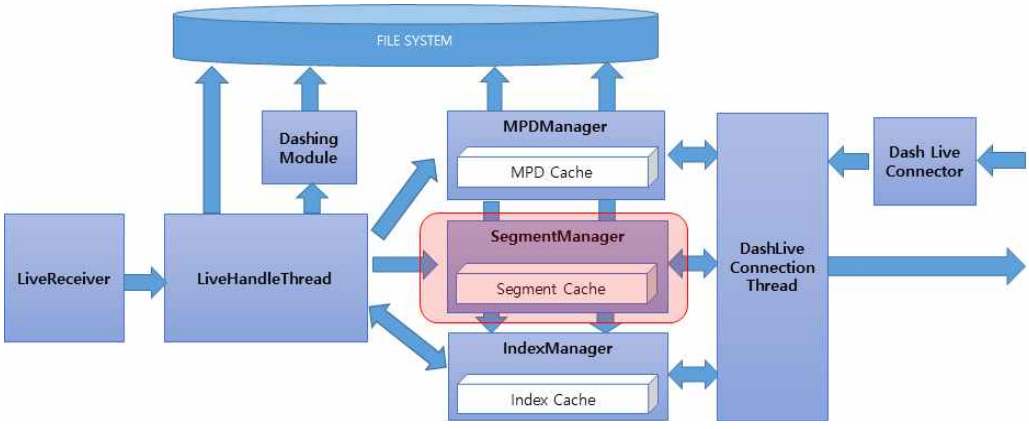

Fig. 5. Low Latency Live Streaming Server Structure.

영상을 실시간으로 수신하는 LiveReceiver모듈과, 이를 실시간으로 처리하기 위한 하위 쓰레드 모듈인 LiveHandleThread으로 수신부를 구성하였다. 시간단위로 분할되어 수신되는 영상을 LiveHandle Thread에서 바로 Dashing을 수행하여 File System에 적재하고, Dashing된 정보를 MPD Manager, Index Manager, Segment Manager 모듈로 전달하여 각 파일들을 메모리에 캐싱 또는 파일로 관리하도록 하였다.

전반적인 구성은 기존 라이브 스트리밍 서버[7]와 동일하지만, 본 논문에서 제안한 Center Segment Tile 정보를 기반으로 Cache를 처리하기 위해 Dash LiveConnectionThread에서 URI 파싱에 따른 Tile Segment 캐싱 요청 처리 로직을 아래 Fig. 6과 같이 추가하였다. 그리고 아래 Fig. 7과 같이 Segment Cache를 레벨 구조로 설계하였고, 아래 Fig. 7, Fig 8과 같이 Segment Manager에서 Tile Segment를 캐싱하고, 하위 레벨 캐시로 Drop하는 기능을 추가하였고, Searching 알고리즘을 수정하였다.

SegmentManager에서 관리하는 Segment Cache 구조는 아래 Fig. 7과 같이 총 3개의 레벨로 구성되어 있다. Lv1 캐시는 클라이언트에서 수신한 Center Tile Segment 정보를 기반으로, 주변 Tile Segment 파일들을 Caching하고 있으며, 내부 구조는 Key/Value 형태의 HashMap으로 이루어져 있다. Lv2 캐시에서는 최근에 Dashing된 Tile Segment들이 Caching되어 있다. 마지막으로 Lv3 캐시는 LRU(Last Recently Used) 캐시로 구성하였다. 각 캐시의 크기는 사전에 설정되어 있다.

Lv1에는 새로운 Tile Segment를 Caching 할 때, 기존 마지막에 Caching된 Tile Segment를 Lv 3 캐시로 아래 Fig. 8과 같이 이동한다.

Tile Segment 탐색은 아래 Fig. 9과 같이 Lv1, Lv2, Lv3, File System 순으로 이루어진다.

DashLiveConnectionThread에서 신규 Center Tile Segment 요청을 받을 때 마다 Segment Manager에서는 해당 Center Tile Segment의 주변 Tile Segment들을 Lv1 캐시로 Caching한다. 그 과정은 아래 Fig. 10과 같다.

Center Tile Segment의 주변 Tile Segment의 집합은 아래 식 2과 같이 정의할 수 있다.

$$M_{m\times n}(GoT)_{ij} \ni \left[a_{k,l}\right]_{i-1\le k\le i+1,\, j-1\le l\le j+1} - \{a_{i,j}\} \text{ if} \begin{cases} k=m\ (k\le 0) \\ k=1\ \ (k>m) \\ l=n\ \ (l\le 0) \\ l=1\ \ (l>n) \end{cases} \tag{2}$$

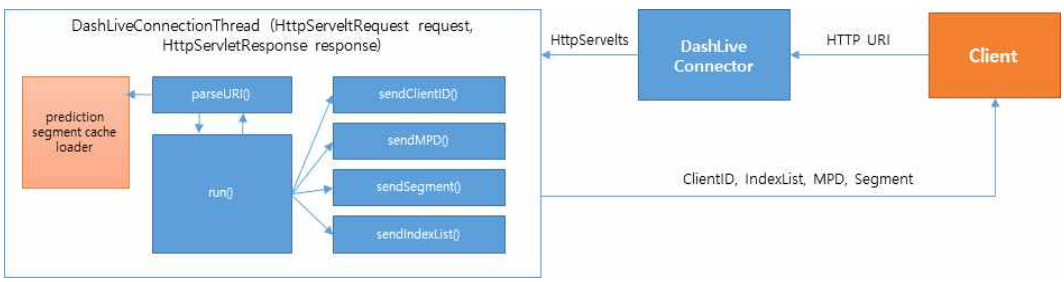

Fig. 6, Streaming Service Modules Detail Structure.

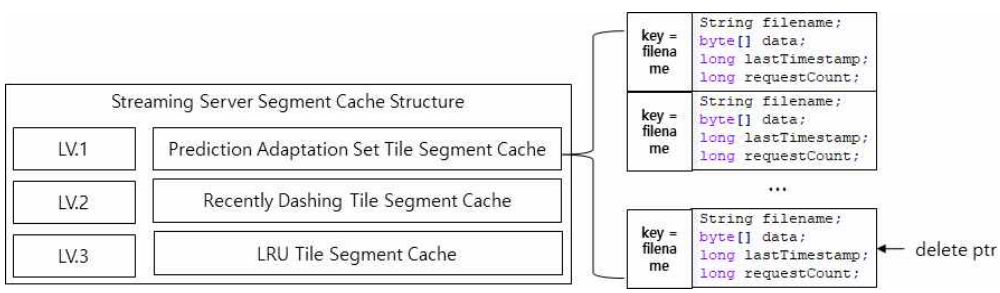

Fig. 7. Tile Segment Level Cache Structure.

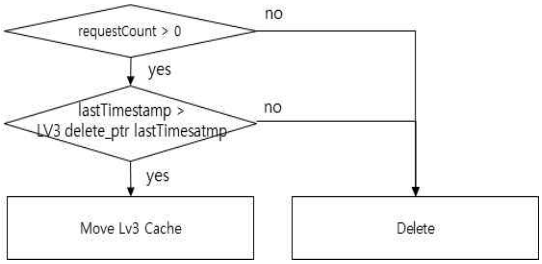

Fig. 8. Level 1 Caching Tile Segment Dropping Algorithm.

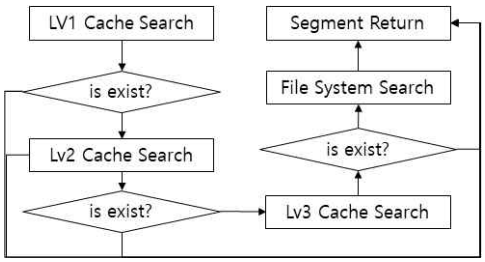

Fig. 9. Tile Segment Searching Algorithm.

$M_{m\times n}$ 은 전체 영상의 Tile Segment들의 인덱스(index)를 의미한다. 인접 Tile Segment들의 집합은 GoT(Grouping of Tiles)으로 정의하였다. k, l은 인접 Tile Segment들의 i, j 배열 인덱스이다.

만약 아래 Fig. 11와 같이 $M_{4\times 4}$로 분할된 영상에서 $\{a_{2,2}\}$ 인접 성분의 집합을 구한다면 아래 식 3과 같이 $\{a_{1,1}, a_{1,2}, a_{1,3}, a_{2,1}, a_{2,3}, a_{3,1}, a_{3,2}, a_{3,3}\}$를 인접한 Tile Segment로 정의하고 Lv1 캐시로 해당 Tile Segment들을 캐싱한다.

$$\{a_{2,2}\}GoT \ni [a_{k,l}]_{1\le k\le 3, 1\le l\le 3} - \{a_{2,2}\} \text{ if, } \begin{cases} k=4\,(k\le 0) \\ k=1\,(k>4) \\ l=4\,(l\le 0) \\ l=1\,(l>4) \end{cases}$$

$$\{a_{2,2}\}GoT \ni \{a_{1,1}, a_{1,2}, a_{1,3}, a_{2,1}, a_{2,2}, a_{2,3}, a_{3,1}, a_{3,2}, a_{3,3}\} - \{a_{2,2}\} \quad (3)$$

4. 시스템 구현 및 실험 결과

4.1 저지연 라이브 스트리밍 서버 구현

스트리밍 서버는 아래 Fig. 12과 같이 JAVA SDK 1.8 64bit으로 구현하였으며, MPEG DASH HTTP 서블릿 패킷 핸들링을 위해 Embedded WAS인 Apache Jetty를 사용하였다.

logs는 로그가 저장되는 디렉토리이며, temp는 Dashing 하기 전 수신 받은 파일을 임시로 저장하는 디렉토리이다. vod는 아래 Fig. 13과 같이 Dashing

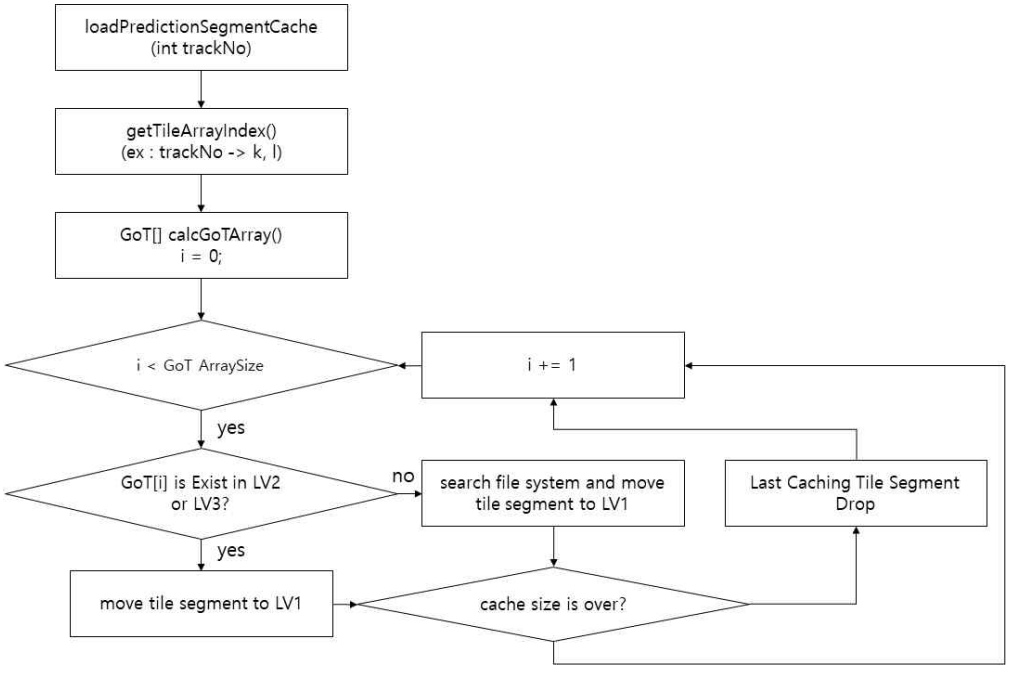

Fig. 10. Prediction Adaptation Set Tile Segment Cache Management Algorithm.

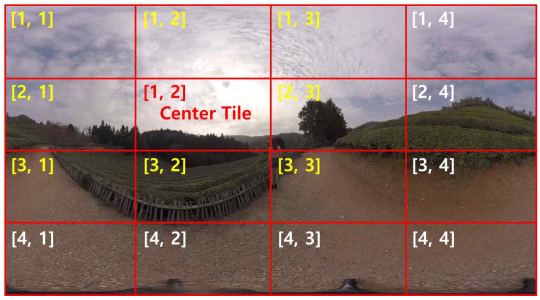

Fig. 11. 4×4 Segment Tile Video Image.

된 영상 파일들이 timestamp 기반의 Index로 생성된 폴더에 저장 관리되는 디렉토리이다. dash-live-streaming-server-0.2-all.jar는 서버 실행 파일이며, start.bat은 서버 실행을 위한 배치 파일이다. log4j.properties와 server.conf는 서버 실행을 위해 필요한 환경설정 값을 가지는 설정파일이다.

4.2 MPEG DASH SRD 기반 Foveated Rendering Player 구현

테스트를 위한 MPEG DASH SRD 기반의 Foveated Rendering 플레이어 UI 구성은 아래 Fig. 14와 같이 고품질과 저품질 Tiles를 확인할 수 있도록 전체 화면과, HMD의 좌안과 우안에 렌더링 되는 화면을 확인할 수 있도록 총 3개의 뷰로 구성하였다.

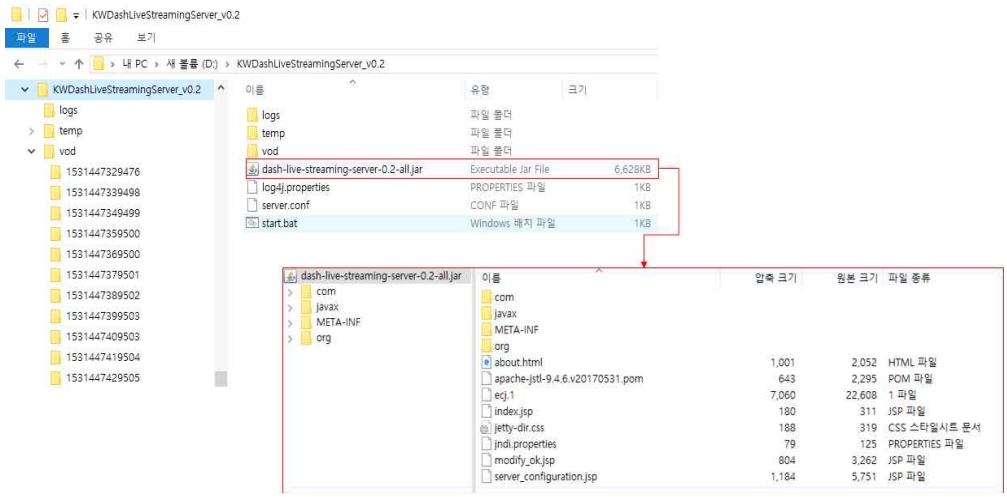

Fig. 12. Dash Streaming Server Implementation Result.

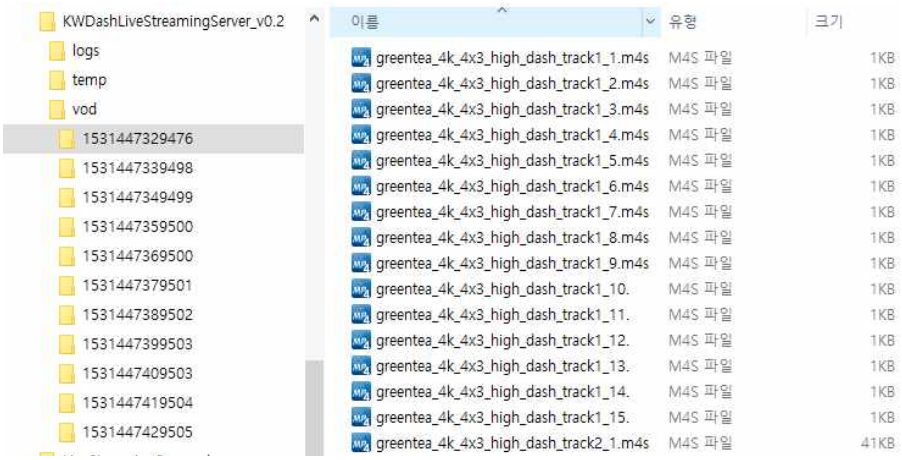

Fig. 13. Index and Dashing File List.

Table 1. Specification of experiment Streaming environment

Item	Streaming Server	360 VR Player Client PC
CPU	Intel Xeon E5-2687W v4 @3GHz (12Core, 24Thread)	Intel Core-i7 6900K @3.2GHz (8Core, 16Thread)
Memory	DDR4 32GB	DDR4 16GB
Ethernet	1Gbps	1Gbps
GPU	-	NVIDIA Geforce GTX 1080Ti

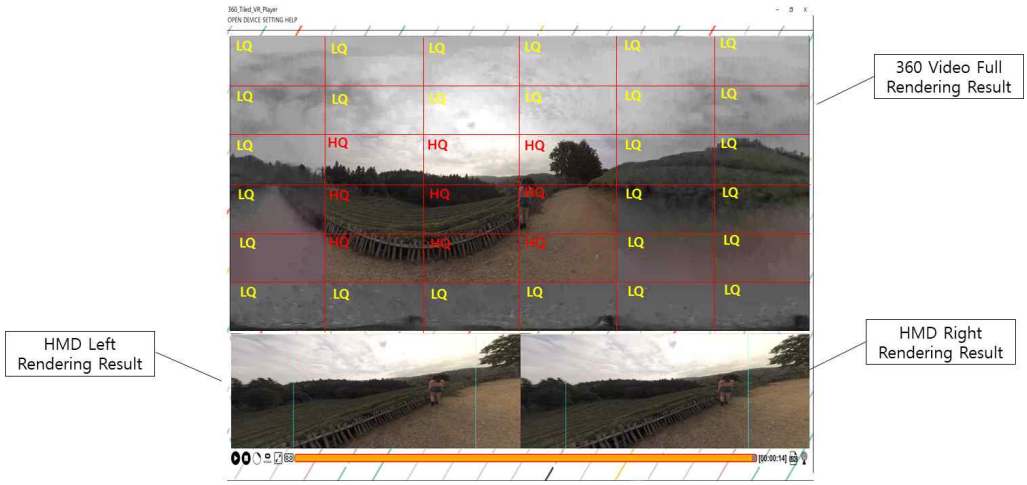

Fig. 14. Monitoring Result of 360 VR Player (Client).

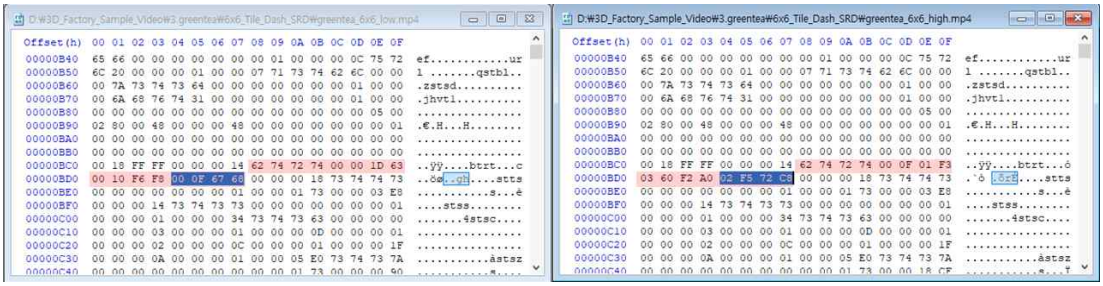

Fig. 15. Bitrate Comparison between Low Quality Video Tile and High Quality Video Tile.

4.3 라이브 스트리밍 서버 성능 측정 환경 구성

메모리 캐싱을 통한 Tile Segment 응답 속도 향상 정도를 확인하기 위해 아래 Table 1과 같이 서버 및 클라이언트 환경을 구성하였다.

테스트를 위한 시료 영상으로는 6×6으로 분할된 7680×3840 해상도를 가지는 8K급 비디오 영상 고품질, 저품질 2개의 파일을 시료로 사용하였다. 고품질 영상 Tiles의 비트레이트는 위 Fig. 15과 같이 약 50000kbps 정도이며, 저품질 영상 Tiles의 비트레이트는 약 1000kbps로 분할 인코딩 하였다.

위와 같이 시료 영상들을 서버로 연속적으로 입력하였으며, 서버에서는 위 Fig. 13과 같이 일정 주기로 Indexing 및 Dashing을 수행하였다. MPEG DASH SRD를 위한 Dashing 모듈로는 Miltimedia Open Source Project인 GPAC의 MP4Box[8]를 사용하였으며, 스트리밍 서버에서는 해당 코드를 빌드하여 JNI로 연동하였다. 입력받은 영상은 “urn:mpeg:dash:profile:isoff-live:2011” 프로파일과, "urn:mpeg:

dash:srd:2014″ scheme에 맞추어 Dashing 하였다.

4.3 라이브 스트리밍 서버 성능 측정

라이브 스트리밍 서버의 하위 Lv2, Lv3 2개의 캐쉬로 동작하였을 때와, Lv1, Lv2, Lv3 캐쉬 모두 사용하였을 때의 한 화면을 구성하는 모든 Tile Segment들을 응답하는데 걸린 총 시간을 Time Segment된 수만큼 측정하여 성능 차이를 비교하였다. 위 영상 시료는 약 14.83초로, 총 15개의 Time Segment로 분할됨으로, 총 15번씩 응답시간을 측정하였다. 그리고 HMD 대수를 1대, 2대, 3대, 4대까지 늘려 측정하였다. Lv1의 캐쉬 사이즈는 HMD 대수 x 36으로 설정하였으며, Lv2는 최신 Dashing된 Tile Segment 수에 맞추어 설정하였다. 본 시료 영상의 Dashing Tile Segment 수는 576이다. 그리고 마지막으로 LRU 캐시의 크기는 50개의 Tile Segment가 저장되도록 고정하여 테스트를 하였으며, 그 결과는 아래 Fig. 16과 같다.

HMD가 1개 일 때, 첫 번째 Lv1 캐시를 검색한 후, Lv2 캐시를 검색하기 때문에 2Level 캐시를 사용하였을 때 보다 총 Tile의 응답 속도의 합이 오래 걸리지만, HMD 개수가 늘어날수록 크기가 작은 Lv1 캐시에서의 검색에 따른 응답 속도가 개선이 되는 것을 확인할 수 있었다.

Lv1 캐시로의 Tile Segment Add/Drop에 따른 CPU Usage가 증가할 것으로 예상하였으나, 더 작은 캐시에서 검색하여 전송하기 때문에 아래 Fig. 17과 같이 큰 차이는 확인 할 수 없었다.

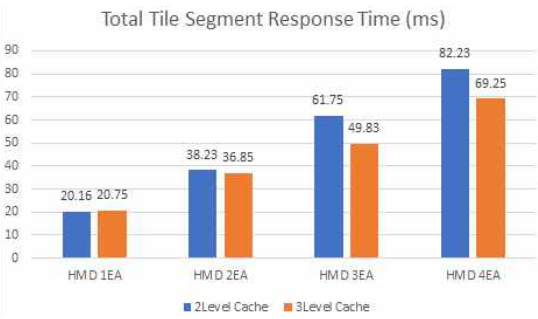

Fig. 16. Tile Segment Response Time Summary.

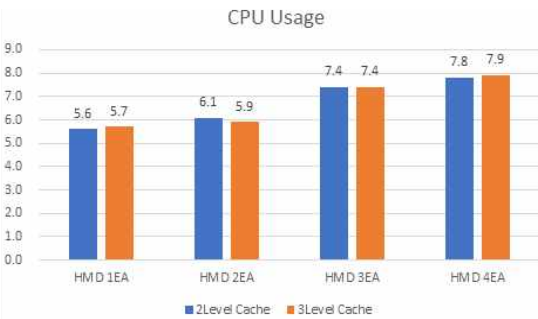

Fig. 17. CPU Usage measurement result during Streaming Time.

5. 결 론

본 논문에서는 MPEG DASH SRD 기반 8K급 고품질 분할 영상 스트리밍 응답 속도를 향상시키기 위해, HTTP 프로토콜을 수정하고, 플레이어와 스트리밍 서버를 설계하였다. 스트리밍 서버에서는 플레이어에서 전달하는 중앙 공간에 위치하는 Tile정보를 플레이어로부터 수신하고, 이를 중심으로 주변에 위치할 고품질 타일을 메모리에 캐싱하여 빠른 응답을 할 수 있도록 하기 위한 캐시 버퍼 구조 및 관리 방법을 제안하고 설계하고, 구현하였다. 그리고 실험을 통해 서버레벨에서의 응답 속도가 향상됨을 확인하였다. 또한 서버의 리소스 사용량을 확인하여 안정적인 서비스가 제공될 수 있음을 확인하였다. 앞으로 더욱더 효율적이고 재사용률이 높은 메모리 캐싱 구조를 연구하고, 이를 위한 프로토콜 연구를 지속해 나갈 예정이다.

REFERENCE

[1] C. Kim, *Adaptive 360-Degree Video Streaming Using Spatial Segmentation*, Doctor's Thesis of Kwangwoon University, 2017.

[2] J. Jeong, T. Ban, and H. Jung, "An Implementation of Dynamic and Adaptive Streaming System over HTTP," *Journal of the Korea Institute of Information and Communication Engineering*, Vol. 16, No. 3, pp. 476-481, 2012.

[3] O.A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S.Y. Lim, "MPEG-DASH SRD-Spatial Relationship Description," *MMSys'16 Proceedings of the 7th International Conference on Multimedia Systems*, No. 5, 2016.

[4] J. Kim, J. Lee, K. Kim, and D.Y. Suh, "User Profile Based Seamless Framework Under HTTP Adaptive Streaming Environment," *Journal of Broadcast Engineering*, Vol. 16, Issue 1, pp. 155-173, 2011.

[5] M. Hosseini and V. Swaminathan, "Adaptive 360 VR Video Streaming Based on MPEG-DASH SRD," *Proceeding of 2016 IEEE International Symposium on Multimedia*, pp. 407-408, 2016.

[6] G. Park, G. Lee, J. Lee, and K. Kim, "HTTP Adaptive Streaming Method for Service-compatible 3D Contents Based on MPEG DASH," *Journal of Broadcast Engineering*, Vol. 17, Issue 2, pp. 207-222, 2012.

[7] H.W. Kim and S.H. Yang, "Implentation of 360 VR Tiled Video Player with Eye Tracking Based Foveated Rendering," *Journal of Korea Multimedia Society*, Vol. 21, No. 7, pp. 795-801, 2017.

[8] H. Kim, J. Yang, Y. Kim, S. Yoon, and W. Park, "MPEG-DASH Based Live Streaming Server design," *Proceeding of the Winter Conference of the Korea Multimedia Society*, pp. 312-313, 2017.

[9] GPAC Multimedia Open Source Project MP 4Box, https://gpac.wp.imt.fr/mp4box/ (accessed Jul., 25, 2017).

김 현 욱

2009년 광운대학교 컴퓨터공학과 (공학사)
2013년~2016년 ㈜케이사인 정보보안연구소 선임연구원
2009년~현재 광운대학교 대학원 전자공학과 석박사통합과정
관심분야: 스마트 홈, 임베디드 시스템, 콘텐츠 응용 기술

최 우 성

2005년 일본규슈대학교 재료공학 이학박사
1993년~2017년 원광대학교 공과대학 정보통신공학과 전임강사, 조교수, 부교수, 교수
2018년~현재 (주)유일씨엔티 이사
관심분야: 무선통신, 이동통신

양 성 현

1993년 광운대학교 대학원 전기공학(자동제어) 공학박사
1991년~현재 광운대학교 전자공학과 교수
2005년~현재 광운대학교 Smart H&B Technology Center 센터장
연구분야: 스마트 홈, 디지털 로직, 임베디드 시스템, IoT, 상황인식(행동인식알고리즘) 시스템