(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년05월30일
(11) 등록번호 10-1735435
(24) 등록일자 2017년05월08일

(51) 국제특허분류(Int. Cl.)
H04L 29/06 (2006.01) *H04L 12/835* (2013.01)
H04L 12/853 (2013.01) *H04N 21/2343* (2011.01)
H04N 21/472 (2011.01) *H04N 21/6587* (2011.01)
H04N 21/845 (2011.01)
(21) 출원번호 10-2011-0007236
(22) 출원일자 2011년01월25일
심사청구일자 2016년01월25일
(65) 공개번호 10-2012-0010090
(43) 공개일자 2012년02월02일
(30) 우선권주장
1020100070194 2010년07월20일 대한민국(KR)
(뒷면에 계속)
(56) 선행기술조사문헌
US20100094963 A1
US20100138736 A1
US20100153578 A1
US20100166309 A1

(73) 특허권자
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
경희대학교 산학협력단
경기도 용인시 기흥구 덕영대로 1732 (서천동, 경희대학교 국제캠퍼스내)
(72) 발명자
박경모
서울특별시 강남구 삼성로 212, 23동 1301호 (대치동, 은마아파트)
서덕영
경기도 성남시 분당구 수내로 174 201동 202호 (수내동,푸른마을벽산신성아파트)
(뒷면에 계속)
(74) 대리인
이건주

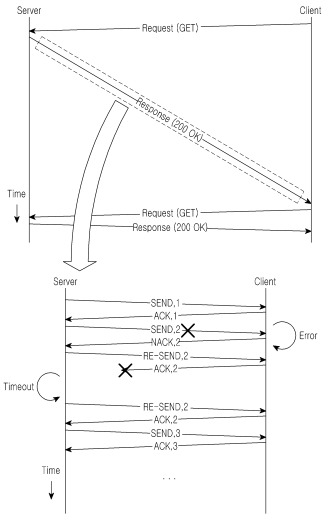
전체 청구항 수 : 총 14 항 심사관 : 김성태

(54) 발명의 명칭 **멀티미디어 스트리밍 서비스 제공 방법 및 장치**

(57) 요 약

본 발명은 미디어 스트리밍 서비스 제공 방법 및 장치에 관한 것이다. 본 발명에 따르면 클라이언트 측 버퍼의 상황 및 네트워크의 상황을 고려하여 서버, 클라이언트, 또는 프록시에서 타임아웃을 설정하고, 기준 시간 이내에 전송되는 데이터만을 처리하도록 하여 네트워크 상황이 좋지 않은 사용자의 종단 간 전송 지연을 줄인다. 또한 거짓 ACK(ACK spoofing)을 이용하여 무선 채널에서 손실이 나더라도 저속시작(slow start)이 일어나지 않도록 하여 전송지연을 감소시키고 가용한 비트율을 효율적으로 이용할 수 있도록 한다. 또한 프리뷰 채널 또는 세그먼트를 이용하여 초기에 저속 시작시에는 품질이 낮은 것부터 시작하도록 하여 초기 버퍼링 지연 및 채널변경지연 시간을 줄인다.

대 표 도 - 도1



(72) 발명자

이용헌

경기도 수원시 권선구 수성로 47, 3동 209호 (구운동, 삼환아파트)

송재연

서울특별시 강남구 선릉로85길 18, 성보 아파트 B동 805호 (역삼동)

(30) 우선권주장

1020100080553 2010년08월19일 대한민국(KR)
1020100101121 2010년10월15일 대한민국(KR)
1020110007075 2011년01월24일 대한민국(KR)

명 세 서

청구범위

청구항 1

멀티미디어 스트리밍 서비스를 제공하는 방법에 있어서,

서버에서 MPD(media presentation description)을 클라이언트로 전송하는 과정과,

상기 클라이언트로부터 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 수신하는 과정과,

상기 서버에서 상기 부분 요청 메시지에 대한 응답으로 상기 소정의 바이트 레인지에 대응하는 적어도 하나의 세그먼트를 상기 클라이언트로 전송하는 과정을 포함하며,

상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며,

상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함함을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 2

제1항에 있어서,

상기 프래그먼트 인덱스 정보는 상기 서브프래그먼트 각각에 대한 우선순위를 나타내는 레벨의 개수에 관한 정보를 더 포함함을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 3

제1항에 있어서,

상기 세그먼트 인덱스 정보는 상기 적어도 하나의 프래그먼트 내의 상기 서브프래그먼트들의 배치 방법을 나타내는 지시자를 포함함을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 4

멀티미디어 스트리밍 서비스를 제공받는 방법에 있어서,

클라이언트에서 서버로부터 MPD(media presentation description)을 수신하는 과정과,

상기 수신된 MPD에서 세그먼트 정보를 검출하는 과정과,

상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 상기 서버로 전송하는 과정과,

상기 부분 요청 메시지에 대한 응답으로 상기 소정의 바이트 레인지에 대응하는 적어도 하나의 세그먼트를 상기 서버로부터 수신하는 과정을 포함하며,

상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며,

상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함함을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 5

제4항에 있어서,

상기 프래그먼트 인덱스 정보를 이용하여 소정 레벨의 적어도 하나의 서브프래그먼트에 접근하는 과정과,

상기 적어도 하나의 서브프래그먼트를 디코딩하는 과정을 더 포함하는 것을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 6

제4항에 있어서,

상기 프래그먼트 인덱스 정보는 상기 서브프래그먼트 각각에 대한 우선순위를 나타내는 레벨의 개수에 관한 정보를 더 포함함을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 7

제4항에 있어서,

상기 세그먼트 인덱스 정보는 상기 적어도 하나의 프래그먼트 내의 상기 서브프래그먼트들의 배치 방법을 나타내는 지시자를 포함함을 특징으로 하는 멀티미디어 스트리밍 서비스 제공 방법.

청구항 8

멀티미디어 스트리밍 서비스를 제공하는 서버 장치에 있어서,

MPD(media presentation description)를 클라이언트로 전송하는 MPD 전송부와,

상기 클라이언트로부터 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 수신하고, 상기 서버에서 상기 부분 요청 메시지에 대한 응답으로 상기 소정의 바이트 레인지에 대응하는 적어도 하나의 세그먼트를 상기 클라이언트로 전송하는 송수신부를 포함하며,

상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며,

상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함함을 특징으로 하는 서버 장치.

청구항 9

제8항에 있어서,

상기 프래그먼트 인덱스 정보는 상기 서브프래그먼트 각각에 대한 우선순위를 나타내는 레벨의 개수에 관한 정보를 더 포함함을 특징으로 하는 서버 장치.

청구항 10

제8항에 있어서,

상기 세그먼트 인덱스 정보는 상기 적어도 하나의 프래그먼트 내의 상기 서브프래그먼트들의 배치 방법을 나타내는 지시자를 포함함을 특징으로 하는 서버 장치.

청구항 11

멀티미디어 스트리밍 서비스를 제공받는 클라이언트 장치에 있어서,

서버로부터 MPD(media presentation description)를 수신하는 MPD 수신부와,

상기 수신된 MPD에서 세그먼트 정보를 검출하고, 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 상기 서버로 전송하는 데이터 요청부와,

상기 부분 요청 메시지에 대한 응답으로 상기 소정의 바이트 레인지에 대응하는 적어도 하나의 세그먼트를 상기 서버로부터 수신하는 데이터 수신부를 포함하며,

상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차

지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며,

상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함함을 특징으로 하는 클라이언트 장치.

청구항 12

제11항에 있어서,

상기 프래그먼트 인덱스 정보를 이용하여 소정 레벨의 적어도 하나의 서브프래그먼트에 접근하여 상기 적어도 하나의 서브프래그먼트를 디코딩하는 디코딩부를 더 포함함을 특징으로 하는 클라이언트 장치.

청구항 13

제11항에 있어서,

상기 프래그먼트 인덱스 정보는 상기 서브프래그먼트 각각에 대한 우선순위를 나타내는 레벨의 개수에 관한 정보를 더 포함함을 특징으로 하는 클라이언트 장치.

청구항 14

제11항에 있어서,

상기 세그먼트 인덱스 정보는 상기 적어도 하나의 프래그먼트 내의 상기 서브프래그먼트들의 배치 방법을 나타내는 지시자를 포함함을 특징으로 하는 클라이언트 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 HTTP(Hypertext Transfer Protocol)와 TCP(Transmission Control Protocol) 기반의 멀티미디어 서비스에 관한 것으로, 특히 HTTP와 TCP 기반의 멀티미디어 서비스에서 서비스 품질을 보장하는 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] HTTP 기반 멀티미디어 서비스는 클라이언트의 콘텐츠 요청(GET)과 그에 대한 서버의 응답(Response)을 통해 데이터 송수신이 이루어진다. 최초 클라이언트의 접속 시 서버에서는 서비스 제공 가능한 콘텐츠 목록 및 각 콘텐츠에 대한 미디어 기술자 파일(Manifest file)을 전송한다. 미디어 기술자 파일에는 미디어의 종류, 평균 비트율, 일정 시간 단위로 분할된 콘텐츠 조각의 URI (Uniform Resource Identifier) 또는 URL (Uniform Resource Locator) 등과 같이 해당 콘텐츠를 서비스받기 위해 필요한 정보들이 기술되어 있다. 미디어 기술자 파일을 토대로 클라이언트는 자신에게 필요한 콘텐츠를 반복적으로 요청한다.

[0003] 클라이언트의 단말 및 네트워크 상황은 모두 다르기 때문에 서버에서는 다양한 단말, 다양한 네트워크에 속한 클라이언트의 서비스 요구를 만족시키기 위해 동일한 콘텐츠에 대해 각기 다른 품질을 갖도록 부호화된 스트림을 가질 수 있다. 그러므로 클라이언트는 자신의 단말 상황 또는 네트워크 상황에 적합한 품질의 스트림을 서버에 요청함으로써 끊김 없는 서비스를 제공받을 수 있다.

[0004] 미디어 기술자 파일은 일정한 시간 단위로 미디어 스트림을 구분한다. 그러므로 클라이언트에서는 매 요청시간마다 동 시각 범위의 스트림 조각 중 자신의 상황에 가장 적합한 하나를 선택하여 서버로 요청 메시지(HTTP GET)를 전송한다. 클라이언트의 요청에 대하여 서버에서는 해당 스트림 조각을 응답 메시지 헤더(상태 코드 200 OK)와 함께 전송한다.

[0005] 서버에서 전송하는 응답 메시지의 크기가 클 경우, 하나 이상의 TCP 패킷으로 구분되어 전송된다. 모든 구분된 TCP 패킷이 정상적으로 클라이언트에 수신되면 원래의 HTTP 응답 메시지 형태로 재구성하여 HTTP 미디어 스트리밍 클라이언트에 전달된다.

[0006] TCP에서는 데이터 송신 및 수신 과정의 신뢰성 보장을 위해 ARQ (Automatic Repeat Request) 방법을 사용한다. ARQ는 수신자로부터 수신된 데이터에 오류가 발견됨을 알리는 응답 메시지(NACK)를 수신하거나 또는 정해진 시간 안에 수신자로부터 어떠한 응답 메시지도 수신하지 못한 경우 해당 데이터를 자동 재전송하는 방법이다. 그러므로 HTTP나 FTP (File Transfer Protocol)와 같이 TCP를 이용하는 모든 서비스는 전송 신뢰성을 보장할 수 있다.

[0007] 만일 구분된 TCP 패킷의 일부가 전송도중 손실되거나 에러가 발생하는 경우, 해당 TCP 패킷은 정상 수신될 때까지 재전송 과정을 거치게 된다. 그러므로 HTTP나 FTP와 같이 TCP를 이용하는 모든 서비스는 전송 신뢰성을 보장할 수 있다.

[0008] 그러나 데이터를 전송하는 과정에서 혼잡이나 간섭으로 인해 TCP 패킷이 손실되거나 에러가 발견되거나 전송 채널 상황이 좋지 않아 TCP 계층에서 잦은 재전송이 일어나는 경우에 종단 대 종단 간의 전송 지연이 크다.

[0009] 또한 네트워크의 상황이 좋지 않고 HTTP 계층에서 전송하고자 하는 데이터의 크기가 클 경우, 종단 대 종단간 전송 지연은 더욱더 커진다. 따라서 멀티미디어와 같이 전송 지연에 민감한 서비스에서는 종단 대 종단 지연이 커질 경우 서비스 품질을 보장할 수 없다.

[0010] 도 1은 이러한 HTTP 계층의 데이터 송수신 과정(상단)과 TCP 계층의 전송 과정(하단)을 도시한 것이다.

[0011] 도 1의 상단은 HTTP 계층에서 클라이언트의 요청에 대한 서버의 응답 과정을 나타낸 것이다. 클라이언트에서는 서버로부터 전송된 응답 메시지(상태 코드 200 OK)를 완전히 수신한 뒤, 사용 가능하다.

[0012] 도 1의 하단은 서버로부터 전송된 응답 메시지가 TCP 계층에서 서버로부터 클라이언트로 분할 전송되는 과정을 나타낸 것이다. 전송 과정에서 에러 및 손실이 발생되는 경우 재전송을 수행하며, 분할된 모든 TCP 패킷이 전송될 때까지 반복하므로 전송 지연이 발생한다. 또한 손실이 발생하면 전송률이 크게 떨어지므로 전송이 지연된다.

[0013] 또한 기존의 HTTP 기반의 미디어 스트리밍 서비스는 일정한 시간 단위로 스트림을 구분하고, 클라이언트의 요청에 따라 해당 스트림을 전달하므로 초기 버퍼링에 의한 지연을 감소시키는 데 한계가 있다.

[0014] 또한 TCP에서는 흐름 제어(flow control)를 위해 저속 시작(slow start) 방법을 사용한다. 이 방법은 조금씩 비트 율을 늘려가는 방식으로 전송하다가 손실이 생기면 다시 비트 율을 크게 떨어뜨리고 다시 저속시작을 하는 것이다. 그러나 이 방식 또한 손실이 많은 무선통신에서는 가용한 비트 율을 충분히 사용할 수 없게 하여 시스템의 효율을 크게 떨어뜨린다.

발명의 내용

해결하려는 과제

[0015] 본 발명은 HTTP 기반 멀티미디어 서비스에서 종단 대 종단 간 전송 지연 문제로 인해 서비스 품질이 떨어지는 문제를 해결하고자 한다.

[0016] 또한 본 발명은 HTTP와 TCP를 이용하는 멀티미디어 스트리밍 서비스에서 발생할 수 있는 전송 지연 및 비효율적인 자원 활용 상황을 해결하고자 한다.

[0017] 또한 본 발명은 HTTP기반 멀티미디어 서비스에서 발생할 수 있는 전송지연을 감소시키고 가용한 비트 율을 충분히 활용하기 위한 방법을 제안함으로써 서비스 품질을 보장하고자 한다.

과제의 해결 수단

[0018] 본 발명의 실시 예에 따르면, 멀티미디어 스트리밍 서비스를 제공하는 방법에 있어서, 서버에서 MPD(media presentation description)을 클라이언트로 전송하는 과정과, 상기 클라이언트로부터 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 수신하는 과정과, 상기 서버에서 상기 부분 요청 메시지에 대한 응답으로 상기 소정의 레인지에 대응하는 적어도 하나의 세그먼트를 상기 클라이언트로 전송하는 과정을 포함하며, 상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며, 상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를

포함한다.

[0019] 또한 본 발명의 실시 예에 따르면, 멀티미디어 스트리밍 서비스를 제공받는 방법에 있어서, 클라이언트에서 서버로부터 MPD(media presentation description)을 수신하는 과정과, 상기 수신된 MPD에서 세그먼트 정보를 검출하는 과정과, 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 상기 서버로 전송하는 과정과, 상기 부분 요청 메시지에 대한 응답으로 상기 소정의 레인지에 대응하는 적어도 하나의 세그먼트를 상기 서버로부터 수신하는 과정을 포함하며, 상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며, 상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함한다.

또한 본 발명의 실시 예에 따르면, 멀티미디어 스트리밍 서비스를 제공하는 서버 장치에 있어서, MPD(media presentation description)를 클라이언트로 전송하는 MPD 전송부와, 상기 클라이언트로부터 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 수신하고, 상기 서버에서 상기 부분 요청 메시지에 대한 응답으로 상기 소정의 레인지에 대응하는 적어도 하나의 세그먼트를 상기 클라이언트로 전송하는 송수신부를 포함하며, 상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며, 상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함한다.

또한 본 발명의 실시 예에 따르면, 멀티미디어 스트리밍 서비스를 제공받는 클라이언트 장치에 있어서, 서버로부터 MPD(media presentation description)를 수신하는 MPD 수신부와, 상기 수신된 MPD에서 세그먼트 정보를 검출하고, 상기 MPD에 정의된 바이트 레인지를 가지는 미디어 데이터에 대한 부분을 요청하는 부분 요청 메시지를 상기 서버로 전송하는 데이터 요청부와, 상기 부분 요청 메시지에 대한 응답으로 상기 소정의 레인지에 대응하는 적어도 하나의 세그먼트를 상기 서버로부터 수신하는 데이터 수신부를 포함하며, 상기 세그먼트는 적어도 하나의 프래그먼트와, 상기 세그먼트 내에서 상기 적어도 하나의 프래그먼트 각각이 차지하는 바이트 레인지를 나타내는 세그먼트 인덱스 정보와, 상기 적어도 하나의 프래그먼트에 포함된 복수의 서브 프레그먼트의 서로 다른 레벨에 접근하기 위한 정보를 포함하는 프래그먼트 인덱스 정보를 포함하며, 상기 프래그먼트 인덱스 정보는 상기 서브 프레그먼트의 총 크기에 관한 정보를 포함한다.

발명의 효과

[0020] 이하에서 개시되는 발명 중 대표적인 것에 의해 얻어지는 효과를 간단히 설명하면 다음과 같다.

[0021] 본 발명은, 타임 아웃(timeout) 방식을 이용하여 전송을 중단함으로써 전송지연을 크게 줄일 수 있다. 또한 무선채널에서 페이딩에 의한 패킷 손실 시에는 거짓 ACK를 전송하여 비트 율을 충분하게 활용할 수 있다. 또한 프리뷰 방식을 이용하여 저속시작 환경에서 첫 화면이 빨리 뜨게 함으로써 초기 버퍼링 지연과 채널 변경 지연 시간을 크게 줄일 수 있다. 이로써 사용자는 가용한 무선자원을 충분히 활용하면서 좀더 쾌적하게 HTTP 스트리밍 서비스를 즐길 수 있다.

도면의 간단한 설명

[0022] 도 1은 HTTP 계층의 데이터 송수신 과정(상단)과 TCP 계층의 전송 과정(하단)을 도시한 도면;

도 2는 MPEG-4/AVC NAL 헤더 구조를 도시한 도면;

도 3은 클라이언트 Timeout 기반의 송신 및 수신 과정의 예를 도시한 도면;

도 4는 클라이언트 Timeout 과 HTTP 부분 전송을 이용하는 송신 및 수신 과정의 예를 도시한 도면;

도 5는 클라이언트 버퍼 모델의 예를 도시한 도면;

도 6은 전송지연이 발생하는 예를 도시한 도면;

도 7은 응용 계층 버퍼와 초기 지연을 도시한 도면;

도 8은 정상 버퍼 상태 도달 목표치 수정을 통한 버퍼링 지연 감소 방법을 도시한 도면;

도 9는 인트라 프레임 기반 버퍼링 지연 감소 방법의 예를 도시한 도면;

도 10은 본 발명의 실시 예에 따른 MPEG 및 3GPP HTTP 스트리밍의 미디어 인덱스 구조의 일 예를 보이고 있는 도면;

도 11은 본 발명의 실시 예에 따른 segment 구조의 일 예를 보이고 있는 도면;

도 12는 본 발명의 실시 예에 따른 sample들의 배치에 의한 스트림 구조를 보이고 있는 도면.

발명을 실시하기 위한 구체적인 내용

[0023] 이하 첨부된 도면을 참조하여 본 발명의 실시 예에 대한 동작 원리를 상세히 설명한다. 하기에서 본 발명을 설명함에 있어 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 그리고 후술 되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

[0024] 본 발명에서는 HTTP 기반 멀티미디어 스트리밍 서비스에서 서비스 품질을 보장하기 위해 전송 지연을 줄이는 방법으로 다음 세 가지 방식을 제안한다.

[0025] 첫째, 타임아웃 (Timeout) 기반의 데이터 송신 및 수신 기술을 제안한다. 멀티미디어 서비스는 지연에 민감하므로 신뢰성 보장을 위해 반복된 재전송을 수행하는 방법이 오히려 서비스 품질을 떨어뜨리게 된다. 따라서 본 발명에서는 클라이언트 측 버퍼의 상황 및 네트워크의 상황을 고려하여 서버, 클라이언트, 또는 프록시에서 Timeout을 설정하고, 기준 시간 이내에 전송되는 데이터만을 처리하도록 하여 네트워크 상황이 좋지 않은 사용자의 종단 간 전송 지연을 줄인다. 그러므로 부분 전송과 Timeout을 활용하는 경우, 전송 지연 감소, 채널 전환 및 랜덤 액세스에 따른 최초 화면 표시 지연 (Initial Presentation delay, Zapping delay) 감소 및 세분화된 서비스 품질 제공 등이 가능하다.

[0026] 둘째, 거짓 ACK(ACK spoofing)을 이용하여 무선 채널에서 손실이 나더라도 저속시작(slow start)이 일어나지 않도록 하여 전송지연을 감소시키고 가용한 비트 율을 효율적으로 이용할 수 있도록 한다.

[0027] 셋째, 프리뷰 채널 또는 세그먼트를 이용하여 초기에 저속 시작(slow start) 시에는 품질이 낮은 것부터 시작하도록 하여 초기 버퍼링 지연 및 채널변경 지연시간을 줄인다.

[0028] 이하에서는 본 발명의 실시 예에 따라 지연시간을 줄이는 방법을 상세하게 설명한다.

[0029] HTTP 부분 전송을 위한 미디어 기술자에는 각 미디어 콘텐츠의 우선순위와 각 우선순위별 범위 정보를 담는다. 예를 들어 현재 MPEG (Moving Picture Experts Group)-4/AVC (Advanced Video Coding)로 부호화된 비디오를 서비스한다면, 미디어 기술자에 기록할 때에는 기존의 HTTP 기반 미디어 스트리밍과 같이 주기적인 시간 단위 (예를 들어, 2초)마다 스트림을 구분하고, 구분된 스트림의 주소 정보 등을 미디어 기술자에 기록한다. 그러므로 기존의 기술에서 클라이언트는 오직 미디어 기술자에서 정해진 스트림 시간 간격으로만 서비스를 요청하고 수신받을 수 있다.

[0030] 본 발명에서는 미디어 기술자 파일에 미디어 우선순위 정보 및 우선 순위 정보를 이용하여 재구성된 스트림의 범위 정보 (스트림 파일 내 오프셋 또는 위치)를 기록하여 다양한 클라이언트의 요구를 만족시키고자 한다.

[0031] 도 2는 MPEG-4/AVC NAL 헤더 구조를 도시한 것이다.

[0032] 예를 들어, MPEG-4/AVC의 NAL (Network Abstraction Layer) 패킷 헤더 중 NRI(NAL Reference Index) 필드를 이용하여 같은 NRI 값을 갖도록 스트림을 재구성하고 NRI와 그 범위 정보를 미디어 기술자에 기록한다. 클라이언트에서는 NRI 필드를 이용하여 인트라 프레임으로만 구성된 데이터를 부분 전송 요청 (HTTP Partial GET)함으로써 콘텐츠의 일부 (예를 들어, 프리뷰)를 사용자에게 보여 주는 응용으로 활용할 수 있다.

[0033] 종래에는 전송 지연 문제를 해결하기 위해 TCP 계층에서의 ACK Spoofing 방식을 사용하였다. 그러나 이는 TCP 프로토콜 수정이 필요하다. 따라서 본 발명에서는 응용 계층에서의 Timeout을 제안한다. 응용 계층의 Timeout은 TCP 계층의 ARQ로 인해 발생할 수 있는 전송 지연을 해결하기 위해 서버, 클라이언트 또는 네트워크상의 프록시에서 활용할 수 있다.

[0034] 도 3은 클라이언트 Timeout 기반의 송신 및 수신 과정의 예를 도시한 것이다.

[0035] 도 3에 도시한 바와 같이, HTTP 스트리밍 클라이언트에서는 자신에게 필요한 데이터를 요청(HTTP GET)한 뒤 Timeout 이벤트를 위한 타이머 (또는 카운터)를 동작시킨다. 클라이언트의 요청에 대한 서버의 응답 메시지가 정의된 Timeout 이벤트가 발생하기 전에 도착한다면, 정상 수신된 것으로 처리한다. 그러나 응답 메시지가 도착하기 전에 Timeout 이벤트가 발생 되는 경우, 응답 메시지의 도착 여부와 관계없이 다음 행동을 수행한다.

[0036] 도 4는 클라이언트 Timeout 과 HTTP 부분 전송을 이용하는 송신 및 수신 과정의 예를 도시한 것이다.

[0037] 서버 및 프록시 기반의 Timeout 방법은 Timeout여부를 측정을 담당하는 주체가 달라지며, 그 동작 과정은 동일하다. Timeout을 이용하는 방법에서 가장 중요한 것은 Timeout 값을 결정하는 과정이다. 본 발명에서는 클라이언트의 버퍼상태를 기준으로 Timeout을 결정하며 부가적으로 버퍼 상태에 기반하여 클라이언트에서의 콘텐츠 품질 적응을 수행한다. 다음은 클라이언트 버퍼 모델의 예시이다.

[0038] 또한 상기에서 기술한 바와 같이, 위의 방식을 활용하기 위한 파일 구조로 전송 가능한 미디어 데이터 파일의 일정 주기 혹은 범위를 파일 서술자에 명시하도록 하고, 이를 서버 또는 클라이언트가 정보를 수신받아 이를 바탕으로 서버 혹은 클라이언트가 명시된 주기 및 범위를 참고하여 부분적인 전송 및 단위 구조 전송 방식을 활용하여 콘텐츠 품질 적응을 수행할 수 있도록 한다.

[0039] 도 5는 클라이언트 버퍼 모델의 예를 도시한 것이다.

[0040] 클라이언트의 버퍼는 크게 세 부분으로 나눌 수 있다. 먼저 정상적으로(Timeout 시간 내에) 도착한 데이터를 입력받는 입력부, 버퍼에 저장된 데이터를 복호하기 위해 데이터를 출력하는 출력부, 마지막으로 현재 버퍼 레벨에 따라 서비스 품질 적응을 수행하며, Timeout을 결정하는 버퍼 제어가 존재한다.

[0041] 입력부는 클라이언트의 서비스 요청 및 네트워크 상황에 따라 데이터의 입력률이 결정된다. 출력부는 현재 출력되는 콘텐츠의 품질에 따라 결정된다. 버퍼 제어부는 먼저 버퍼의 크기를 결정한다. 서비스 콘텐츠에서 요구되는 버퍼, 네트워크 지연으로 인해 필요한 버퍼 그리고 유/무선 구간에서의 재전송으로 발생되는 지연에 대처하기 위한 재전송 버퍼의 세 가지 버퍼 요구량을 합쳐서 전체 클라이언트 버퍼의 크기를 결정한다.

[0042] 다음은 버퍼의 레벨을 이용한 서비스 품질 결정 방법에 대해 설명한다. 도 5에서 버퍼의 내부에 존재하는 두 개의 결정 포인트(Decision point) 또는 레벨과 현재의 버퍼 길이를 비교하여 다음 서비스의 품질 레벨 및 Timeout 시간을 조정한다. 예를 들어 버퍼 제어부는 다음과 같이 동작할 수 있다.

[0043] 버퍼에 쌓인 데이터의 크기(Buffer length)가 상단의 Decision point 보다 큰 경우, 버퍼 오버 플로우 발생이 예상되므로, Timeout을 크게 하여 전송 신뢰성을 높이며, 다음 세그먼트에 대한 요청 전에 시간 간격을 둔다. 또한 버퍼에 쌓인 데이터의 크기(Buffer length)가 하단의 Decision point 보다 작은 경우, 버퍼 언더 플로우 발생이 예상되므로, 서비스 레벨을 낮추고, Timeout을 작게 한다.

[0044] Decision point와 현재 버퍼의 길이만을 이용하여 서비스 레벨 및 Timeout 값을 결정하는 경우, 잦은 변화로 인해 사용자가 느끼는 품질은 더 낮아질 수 있으므로 카운터 등을 도입하여 빈번한 서비스 레벨의 변화 및 Timeout 변화를 줄일 수 있다. 그리고 Timeout으로 인해 일부의 데이터가 손실될 수 있으므로, 채널 코딩 방법을 통해 손실된 데이터를 복원할 수 있다.

[0045] 1) 일부 수신 허용 방식에 의한 성능 향상 방식

[0046] 원래 k개의 심볼로 이루어진 세그먼트를 전송함에 있어서 k-n개의 패리티 심볼을 추가하여 n개 심볼의 전송을 시도한다. n개의 심볼 중에 어느 것이나 k개만 수신하면 위의 timeout 방식을 이용하여 전송을 중단한다. 이로써 전송에 걸리는 시간을 줄일 수 있다. 즉, k개를 전송하여 k개 모두를 수신하는 것보다, k보다 큰 n개를 전송하여 k개만 수신한 후 전송을 중단함으로써 전송에 걸리는 시간을 줄인다. 이때 n-k개의 패리티 심볼의 생성을 위해 RS부호(Reed Solomon code)를 이용하였으면 k개의 수신으로 복원이 보장되지만, Raptor code를 이용하면 k보다 몇 개 더 수신하여야 완전 복원의 확률이 높아진다. RS 부호를 이용할 때에는 수신되는 심볼 수를 수시로 확인하여 k개가 수신되면 timeout을 한다. Raptor code를 이용하는 경우에는 Raptor decoding을 동시에 수행하여 디코딩이 완료되면 timeout을 하는 방식을 이용하면 더욱 효율적이다.

[0047] 2) 거짓 ACK (ACK spoofing)에 의한 성능 향상 방식

[0048] 기존의 TCP에서는 흐름 제어(flow control)를 위해 도 6의 (b)와 같이 저속시작(slow start) 방식을 이용한다. 즉, 조금씩 비트율을 늘려가는 것이다. 또한 흐름 제어와 혼잡 제어(congestion)를 위해 손실이 발생하면 바로

비트율을 크게 떨어뜨리고 다시 저속시작을 한다. 이 같은 것이 반복되면 (b)와 같은 형태의 비트율이 되어 가용한 비트율을 충분히 사용할 수 없게 되어 전송 지연이 발생한다. 특히, 무선통신에서는 채널에서 페이딩에 의한 비트에러 또는 충돌(collision)에 의해 패킷이 손실되므로 이로 인하여 충분하게 비트율이 올라가지 않은 상태에서 저속 시작을 하게 되어 더욱 전송 지연이 심하게 발생한다. 따라서, 본 발명에서는 채널에서 페이딩이나 충돌에 의해 패킷이 손실된 경우에는 제대로 받았다고 거짓 ACK를 전송함으로써 저속시작의 회수를 줄인다. 위에서 설명한 '일부 수신 방식'을 이용하면 일부만 수신하여도 되므로 거짓 ACK을 이용하여 비트 율을 충분히 활용하면서도 원하는 세그먼트를 완전하게 받을 수 있다.

[0049] 3) 프리뷰 방식을 활용한 성능 향상 방식

[0050] 본 발명에 따르면 HTTP 기반 멀티미디어 스트리밍에서 초기 지연(Initial delay) 및 채널 변경 지연 (Zapping delay)과 같이 버퍼에 의해 발생하는 지연을 감소시킬 수 있다.

도 7은 응용 계층 버퍼와 초기 지연을 도시한 것이다.

[0051] 도 7에 도시한 바와 같은 버퍼에 의한 초기 지연은 디지터링 (de-jittering)과 디인터리빙 (de-interleaving) 등을 고려하여 결정된다. VOD (Video-On-Demand)의 경우 통상적으로 5~10초 걸린다. 버퍼링 지연 이후에는 정상 버퍼 상태 (steady buffer state)가 유지되어, 더 이상 인터럽트 없이 재생된다 (interrupt-free playback). 디지터링 지연은 단 대 단 경로의 전송률의 변화와 콘텐츠가 요구하는 비트율의 가변성에 의해서 결정된다.

[0052] 만일 디지터링과 디인터리빙에 의한 지연을 T_D라고 하고, 미디어 스트림의 최고 전송률을 R_{max}라고 할 때, 초기 버퍼링 양은 $B_{max} = R_{max}*T_D$라고 할 수 있다. 즉, 수신자 버퍼에 $B_{max} = R_{max}*T_D$만큼 데이터가 저장되면, 클라이언트에서 플레이가 시작될 수 있다. 시그널링 지연은 줄일 수 없지만 버퍼링 지연은 다음 방법으로 줄일 수 있다.

[0053] 첫째, R_{max}를 줄여서 버퍼링 지연을 줄일 수 있다. 다시 말해 버퍼의 정상 버퍼 상태 목표치를 내리는 것을 의미한다. 도 8은 정상 버퍼 상태 도달 목표치 수정을 통한 버퍼링 지연 감소 방법을 도시한 것이다.

[0054] 예를 들어, 계층부호화 인코딩하여, 비디오 스트림을 최대 비트율 $r_{max}(<R_{max})$인 기본계층 스트림과 고급계층 스트림으로 나누어 저장하고, 초기에는 $b_{max} = r_{max}*T_D$에 도달할 때까지는 기본계층만 전송한다면 버퍼링 지연을 줄일 수 있다. 보통 $10r_{max}=R_{max}$이므로 이 방법으로 버퍼링 지연이 대략 1/10로 감소한다. B_{max}까지 버퍼링이 이루어지면 정상적인 품질로 비디오를 재생한다. 이러한 방법은 계층 부호화 및 일반적인 부호화 방법에서도 적용 가능하다. 그러나 네트워크 상황이 좋지 않을 경우, 비효율적이다.

[0055] 둘째, 정상 버퍼 상태 (steady buffer state)에 도달하지 않았더라도, 플레이 할 수 있는 정보만 도착하면 바로 플레이를 시작한다면 초기 화면이 뜨는 데까지 시간을 줄일 수 있다.

[0056] 도 9는 인트라 프레임 기반 버퍼링 지연 감소 방법의 예를 도시한 것이다.

[0057] 도 9와 같이, 인트라 (Intra) 프레임이 하나라도 도착하면 바로 플레이를 시작하여 초기화면이 뜰 때까지의 시간을 줄일 수 있다. 그러므로 본 발명에서 제안하는 우선순위 정보가 기록된 미디어 기술자 파일과 HTTP 부분 전송을 이용할 경우, 초기 버퍼링 지연을 감소시킬 수 있다.

[0058] 만일 기본계층의 GOP 주기가 TD의 1/10이라면 이 방법으로 버퍼링 지연을 1/10 줄일 수 있다. 기본계층은 인트라 주기를 고급계층의 인트라 주기보다 작도록 인코딩한다면 도움이 될 것이다. 기본계층의 첫 화면은 사용자에게 이 채널에서 무엇이 방송되고 있는지 충분히 알려줄 수 있다. 이때 기본계층의 인트라 프레임이 도착했음을 시스템에 알려줄 수 있는 마커가 필요하다.

[0059] 본 발명에서 제안하는 두 가지 방법은 처음 콘텐츠를 시작하는 경우와 채널을 변경하였을 때, 모두 적용할 수 있다. 이 방법을 사용하면 현재 5~10초 정도인 초기 지연 (initial delay) 및 채널 변경 지연 (zapping delay)를 약 200ms 이내로 줄일 수 있다.

[0060] 이하 본 발명의 실시 예의 활용 예에 대해 설명한다.

[0061] 본 발명의 실시 예는 현재 MPEG-DASH 및 3GPP에서 표준화 중인 HTTP 스트리밍에 활용될 수 있다. MPEG-DASH 및 3GPP에서는 HTTP 스트리밍을 위해 Media Presentation Description (MPD) 미디어 기술자를 정의하고 있다. HTTP 스트리밍 클라이언트 (Streaming client)는 미디어 콘텐츠에 대한 MPD를 통해 하나 이상의 Period, Representation 그리고 Segment의 관계를 도 10과 같이 인덱스 (index) 할 수 있다.

[0062] 본 발명의 실시 예에서 제안하는 추가적인 미디어 기술자는 MPD의 SegmentInfo에 하기 <표 1>과 같은 확장이 가능하다.

표 1

[0063]

	type	A		CM Must be present if the "range" attribute is present	segment에 대한 range가 존재하는 경우, 해당 byte-range에 대한 타입을 구분한다. Segment index box의 경우, sidx(s), priority인 경우, 0,1,...,N, sub-time인 경우 time으로 기록한다. Priority인 경우 '0'을 가장 높은 우선순위로 한다. 단일 segment를 하나 이상의 range로 구분하는 경우, 각각을 ';'로 구분한다.
	duration	A		CM Must be present if the "type" attribute is "time"	각 range에 대한 duration을 정의한다. 단일 segment를 하나 이상의 range로 구분하는 경우, 각각을 ';'로 구분한다.

[0064] 이를 통해 본 발명의 실시 예에 의하면, 아래에서 제안하는 각 user-case에 대해 효과적인 동작이 가능하다.

[0065] *(Case 1) "Peter watched a football game of his favorite team. The goal in the 64th minute made him so excited that he wants to watch it over and over again. He finds a recording of the game on the homepage of his local TV station and requests the segment containing the goal"*

[0066] 일반적으로 미디어 표현 서술 (Media Presentation Description)과 세그먼트 색인 상자(sidx box)에 있는 색인 정보는 피터의 DASH 클라이언트를 64 분에 일치하는 세그먼트 내의 세그먼트와 무비 Fragment(s)를 확인하는 것이 가능하게 한다. 상기 Fragment(s)는 호쾌한 골의 재연을 제공하는 HTTP 부분 요구를 사용하여 직접 요구될 수 있었다.

[0067] 하지만 본 발명의 실시 예에 의하면, 하기의 두 가지 방안에 의한 동작이 가능하다.

[0068] 먼저 현재 WD에서 Peter가 시청을 원하는 64th minute의 Movie Fragment(s)에 접근하기 위해서는 아래와 같은 과정이 필요하다.

[0069] MPD의 SegmentInfo를 통해 원하는 Movie Fragment(s)에 가장 근접한 세그먼트를 인덱스하고, 이때 세그먼트 내에서 원하는 시간의 Movie Fragment(s)를 인덱스하기 위해 sidx(s)의 전송을 요청한다.

[0070] 상기 sidx(s)의 전송을 요청할 시에 Single sidx로 구성된 세그먼트인 경우, sidx 크기 정보 (size information) 없이 1회의 HTTP Partial request를 수행한다. 그렇지 않고 Multiple sidxs로 구성된 세그먼트인 경우에는 sidx 크기 정보 (size information) 없이 2회 이상의 HTTP Partial request를 수행한다.

[0071] 그 후 다운-로드된 sidx(s)를 통해 원하는 Movie Fragment(s)를 요청하게 된다.

[0072] 다음으로 본 발명에서 제안하는 MPD extension(range with type="sidx(s)")을 이용하는 경우에는 아래와 같은 과정이 필요하다.

[0073] MPD의 SegmentInfo를 통해 원하는 Movie Fragment(s)에 가장 근접한 segment를 인덱스하고, 이때 세그먼트 내에서 원하는 시간의 Movie Fragment(s)를 인덱스하기 위해 sidx(s)의 전송을 요청한다.

[0074] 상기 sidx(s)의 전송을 요청할 시에 Single sidx로 구성된 세그먼트인 경우, sidx 크기 정보 (size information)와 함께 1회의 HTTP Partial request를 수행한다. 그렇지 않고 Multiple sidxs로 구성된 세그먼트인 경우에는 sidx 크기 정보 (size information)와 함께 1회의 HTTP Partial request를 수행한다.

[0075] 그 후 다운-로드된 sidx(s)를 통해 원하는 Movie Fragment(s)를 요청하게 된다.

[0076] 상술한 바와 같이 본 발명의 실시 예서 제안하는 MPD extension을 이용하는 경우, 사용자가 원하는 Movie Fragment(s)를 요청하기 위해 필요한 server-client 간의 interaction을 감소시킬 수 있으며, sidx(s)의 크기 정보 (size information)가 존재하므로 정확한 크기의 HTTP Partial request가 가능하다.

[0077] *(Case 2) "Paul can't go to a concert of his favorite band, but finds out, that there will be a live stream of it on the bands homepage. He starts watching, but later on a friend of him comes by and wants to see it too. Paul stops the stream and they watch it from beginning to end."*

[0078] 일반적으로 라이브 스트림을 위한 미디어 표현 서술 (Media Presentation Description)는 미디어를 포함하는 세그먼트의 순서를 기술한다. 상기 세그먼트는 동일한 URL에 유효하게 남아 있다. 미디어 표현 서술 (Media Presentation Description)의 timeShiftBufferDepth 분야는 해당 세그먼트가 유효 (이 경우에는 불명확하게)하게 남아 있는 타임 윈도우를 나타낸다. 폴의 DASH 클라이언트는 Presentation의 시작에 상응하는 세그먼트를 언제나 쉽게 요구할 수 있고, 그로 인하여 Presentation 을 처음부터 재생할 수 있다.

[0079] 하지만 본 발명의 실시 예에 의하면, MPD의 SegmentInfo를 통해 원하는 Movie Fragment(s)에 가장 근접한 segment를 인덱스하고, 이때 세그먼트 내에서 원하는 시간의 Movie Fragment(s)를 인덱스하기 위해 sidx(s)의 전송을 요청한다.

[0080] 상기 sidx(s)의 전송을 요청할 시에 Single sidx로 구성된 세그먼트인 경우, sidx 크기 정보 (size information) 없이 1회의 HTTP Partial request를 수행한다. 그렇지 않고 Multiple sidxs로 구성된 세그먼트인 경우에는 sidx 크기 정보 (size information) 없이 2회 이상의 HTTP Partial request를 수행한다.

[0081] 그 후 다운-로드된 sidx(s)를 통해 원하는 Movie Fragment(s)를 요청하게 된다.

[0082] *(Case 3) "Mary wants to watch a stream of a TV series, but wants to skip the opening credits. She requests the stream starting from minute 3."*

[0083] 일반적으로 미디어 표현 서술 (Media Presentation Description)과 세그먼트 색인 상자에 있는 색인 정보는 메리의 DASH 클라이언트를 3 분에 상응하는 세그먼트 내에서 세그먼트와 Movie Fragment(s)를 확인할 수 있도록 한다. 이때 상기 Fragment(s)는 3 분에 시작되는 재연을 제공하는 HTTP 부분 요구를 사용하여 직접 요구될 수 있다.

[0084] 하지만 본 발명의 실시 예에 의하면, 하기의 두 가지 방안에 의한 동작이 가능하다.

[0085] 먼저 현재 WD에서 Mary가 시청을 원하는 3분의 Movie Fragment(s)에 접근하기 위해서는 아래와 같은 과정이 필요하다.

[0086] MPD의 SegmentInfo를 통해 원하는 Movie Fragment(s)에 가장 근접한 segment를 인덱스하고, 이때 세그먼트 내에서 원하는 시간의 Movie Fragment(s)를 인덱스하기 위해 sidx(s)의 전송을 요청한다.

[0087] 상기 sidx(s)의 전송을 요청할 시에 Single sidx로 구성된 세그먼트인 경우, sidx 크기 정보 (size information) 없이 1회의 HTTP Partial request를 수행한다. 그렇지 않고 Multiple sidxs로 구성된 세그먼트인 경우에는 sidx 크기 정보 (size information) 없이 2회 이상의 HTTP Partial request를 수행한다.

[0088] 그 후 다운-로드된 sidx(s)를 통해 원하는 Movie Fragment(s)를 요청하게 된다.

[0089] 다음으로 본 발명에서 제안하는 MPD extension(range with type="sidx(s)"을 이용하는 경우에는 아래와 같은 과정이 필요하다.

[0090] MPD의 SegmentInfo를 통해 원하는 Movie Fragment(s)에 가장 근접한 segment를 인덱스하고, 이때 세그먼트 내에서 원하는 시간의 Movie Fragment(s)를 인덱스하기 위해 sidx(s)의 전송을 요청한다.

[0091] 상기 sidx(s)의 전송을 요청할 시에 Single sidx로 구성된 세그먼트인 경우, sidx 크기 정보 (size information)와 함께 1회의 HTTP Partial request를 수행한다. 그렇지 않고 Multiple sidxs로 구성된 세그먼트인 경우에는 sidx 크기 정보 (size information)와 함께 1회의 HTTP Partial request를 수행한다.

[0092] 그 후 다운-로드된 sidx(s)를 통해 원하는 Movie Fragment(s)를 요청하게 된다.

[0093] 상술한 바와 같이 본 발명의 실시 예서 제안하는 MPD extension을 이용하는 경우, 사용자가 원하는 Movie Fragment(s)를 요청하기 위해 필요한 server-client 간의 interaction을 감소시킬 수 있으며, sidx(s)의 크기

정보 (size information)가 존재하므로 정확한 크기의 HTTP Partial request가 가능하다.

[0094] *(Case 4) "Peter requests a new movie that was published recently and is not stored in the selected server, the movie file is distributed to the server efficiently to serve for the request."*

[0095] 일반적으로 이것은 서버 또는 CDNs를 캐싱 (caching)하는 HTTP의 동작에 대한 표준 모드에 대응한다. 초기에 선택된 서버는 피터의 네트워크와 잠재적 CDN 로드 밴런싱 알고리즘에 부착 포인트에 의해 결정된다. 피터의 DASH 클라이언트는 무비의 세그먼트를 위한 HTTP 요청을 전송한다. 이들은 그 후에 표준 HTTP cache 절차를 밟고 업-스트림 서버에서 세그먼트를 요구하는 HTTP cache에서 존재하지 않는다. 결국 요구는 세그먼트가 유효하기 위하여 보장되는 오리지널 서버로 도달한다.

[0096] 이에 반하여 본 발명에서는 오리지널 서버 (Origin server)에서는 사용자의 요청에 대한 빠른 응답을 제공하기 위하여, 사용자의 다운로드 요청 중 중요도가 높거나(e.g. type="0", 사용자로부터의 요청 빈도가 높은 데이터 (e.g. type="sidx(s)에 대하여 HTTP 캐싱 서버 (caching server) 또는 CDNs을 선택적으로 활용하도록 함으로써 네트워크 대역폭 (bandwidth) 또는 저장 공간 측면에서 효율성을 높일 수 있다.

[0097] *(Case 5) "Frank is watching a stream with a mobile device. In the beginning, he is moving a lot causing high network throughput fluctuations. As he finds a cafe, he sits down, enjoys some coffee and watches the rest of the stream in relatively stable network conditions using the cafe,s WLAN."*

[0098] 일반적으로 전송률은 현재 representation (비트율)에서 계속적인 스트리밍이 부족하거나 고품질 representation에서의 스트리밍이 충분할 때, Frank의 DASH 클라이언트에 의하여 입력 데이터 레이트와 끊임없는 감시를 결심하도록 한다. 이 시점에서는 미디어 표현 서술 (Media Presentation Description)에 있는 색인 정보는 스위칭을 위한 alternative representation의 적당한 세그먼트를 확인하기 위하여 이용된다. 상기 세그먼트의 세그먼트 색인 박스는 현재 representation의 가장 최근에 요구된 데이터의 종료 묘사 시간 (end presentation time) 보다 늦지 않은 랜덤 액세스 포인트의 형식 내에서 스위치 포인트를 확인하기 위해 사용된다. DASH 클라이언트는 새로운 representation에서 데이터를 요구하는 것을 시작하고, 오래된 representation의 미디어를 따르는 끊임없는 playout을 위한 미디어 플레이어에게 제공한다.

[0099] Frank가 WLAN 영역에 들어갈 때, DASH 클라이언트는 고품질 representation의 스트리밍을 지원하기 위하여 자료가 충분히 빨리 도착하고 있다는 것을 검출한다. 이때 상기 스위칭 절차는 반복되고, 고품질 representation과 함께 계속하여 presentation한다.

[0100] 네트워크 throughput fluctuation이 심한 모바일 환경에서 seamless한 서비스를 제공하기 위해서는 throughput 변화에 따른 representation switching이 빠르게 이루어져야 한다. 클라이언트는 MPD를 통해 현재의 throughput 에 적합한 representation와 switching-time에 적합한 segment를 결정할 수 있다. 그러나 해당 segment내에 존재하는 적합한 Random Access Point를 알기 위해서는 sidx(s)를 다운로드 받아야 한다.

[0101] 따라서 본 발명의 실시 예에서는 segment 내 sidx(s)의 배치에 따라 1회 이상의 HTTP 부분적 요구 (Partial request)가 필요하다.

[0102] 한편 MPD extension을 활용하는 경우에 있어서, MPD SegmentInfo에 "sidx(s)" type의 레인지를 활용할 경우, 1회의 HTTP 부분적 요구를 통해 세그먼트의 모든 sidx(s)를 다운받을 수 있다.

[0103] 임시 레벨 (Temporal level) 또는 프레임 타입 (Frame type)에 따라 range(s), type 그리고 duration(s)을 정의한 경우, 가장 높은 우선순위(e.g. type="0" ; I frame)를 갖는 range(s)와 그 구간 (duration(s))을 통해 sidx(s) 없이 스위칭이 가능하다.

[0104] Throughput fluctuation이 심한 모바일 네트워크 환경에서 throughput 변화에 빠르게 representation switching을 하기 위해서는 server-client 간의 인터렉션을 줄이는 것과 bandwidth 효율적인 sidx(s) request가 필요하다. 이러한 측면에서 MPD extension을 이용할 경우, sidx(s) 다운로드를 위한 interaction을 줄이거나 우선순위 타입 및 구간 (duration)을 통해 추가적인 인터렉션 (interaction) 없이 MPD만을 통한 representation switching이 가능해 진다.

[0105] *(Case 6) "Tom has a DASH-ready mobile phone and wants to watch movie content that is available at 3 bit rates. While watching, Tom boards a crowded tram and available bandwidth of the tram goes below those bit rates. Tom can continue to watch the movie."*

[0106] 일반적으로 이번 user-case 내에서 예정된 사용자 경험은 명확하지가 않다. 유효한 대역 폭이 가장 낮은 유효한 비트 레이트의 아래에 남아 있다면, 톰은 다양한 방법으로 playout에서 쉼없이 정상적인 프레임 레이트로 3개의 비트 레이트의 가장 낮은 비트 레이트에 의해 movie를 보는 것을 계속할 수 있다.

[0107] 이에 반하여 본 발명의 실시 예에 따르면, 가장 작은 대역 폭의 representation보다 클라이언트의 throughput이 작은 네트워크 환경에서, 세스먼트 단위의 HTTP 요구를 하는 경우, 클라이언트는 Buffer-underflow로 인해 재생을 일시 중지하고 버퍼링을 수행한다. 그리고 sidx(s) 정보를 다운로드 받은 경우에는 세그먼트 내에서 독립적인 디코딩이 가능한 movie fragment(s)만을 HTTP 부분적 요청 (Partial request)함으로써 Buffer-underflow로 인한 pause 대신 low frame rate로 재생한다.

[0108] 한편 본 발명의 다른 실시 예에 따라 MPD extension을 이용하는 경우, 앞에서 살펴본 실시 예와 동일하게 sidx(s) 정보를 이용하여 세그먼트 내에서 독립적인 디코딩이 가능한 movie fragment(s)만을 HTTP 부분적 요청 (Partial request)하고, 이를 재생한다.

[0109] MPD SegmentInfo에 hierarchical prediction structure의 임시 레벨 (temporal level)에 따른 우선순위(e.g. type="0,1,...,N")와 레인지(range)가 존재하는 경우, 단말은 우선 순위가 높은 레인지(range)만을 HTTP 부분적 요청 (Partial request)하고, 이를 재생한다.

[0110] 그리고 MPD extension에서 레인지(range)와 "sidx(s)" type을 사용하면, sidx(s)를 모두 수신하는데 필요한 server-client 간의 interaction을 줄일 수 있다.

[0111] 본 발명에서 제안하는 미디어 우선 순위 정보 및 스트림의 범위 정보는 미디어 파일 포맷에 기술됨으로써 다음과 같이 활용될 수 있다. 일례로 현재 표준화 중인 MPEG-Dynamic Adaptive Streaming over HTTP (DASH) 및 3GPP Adaptive HTTP Streaming (AHS)에서는 각 미디어 스트림을 일정한 단위로 구분 (movie fragment; moof)하고, 하나 이상의 moof를 시간적인 순서에 따라 그룹화 (segment)하여 활용한다. 단일 moof는 다수의 미디어 sample (e.g. 비디오 프레임 또는 MPEG/AVC NAL unit)의 그룹 (e.g. Group of Picture; GOP)으로 구성될 수 있다. 그러므로 단일 segment 내에 존재하는 moof에 대한 인덱스 정보를 제공하기 위하여, DASH에서는 segment index box(sidx)를 활용하고 있다. 도 10은 본 발명의 실시 예에 따른 MPEG 및 3GPP HTTP 스트리밍의 미디어 인덱스 구조의 일 예를 도시한 것이고, 도 11은 본 발명의 실시예에 따른 segment 구조를 도식화하여 나타낸 것이다.

[0112] 도 11에서 S와 f1,f2,...,f6는 각각 sidx와 moof(fragment)를 의미한다. 상기 sidx에서는 각 moof box의 파일 내 위치 정보를 나타냄으로써 필요에 따라 특정 moof 단위의 접근이 가능하도록 한다. 하기 <표 2>에서는 sidx box의 syntax의 일 예로써 MPEG-DASH 및 3GPP AHS의 sidx syntax를 나타낸다.

표 2

[0113]

```
aligned(8) class SegmentIndexBox extends FullBox('sidx', version, 0) {
        unsigned int(32) reference_track_ID;
        unsigned int(16) track_count;
        unsigned int(16) reference_count;
        for (i=1; i<= track_count; i++)
        {
                unsigned int(32)        track_ID;
                if (version==0)
                {
                        unsigned int(32)        decoding_time;
                } else
                {
                        unsigned int(64)        decoding_time;
                }
        }
        for(i=1; i <= reference_count; i++)
        {
                bit (1)                         reference_type;
                unsigned int(32)        reference_offset;
                bit(1)                          contains_RAP;
                unsigned int(31)        RAP_delta_time;
        }
}
```

[0114] 도 11과 같은 segment 구조를 갖는 경우, 각 moof는 temporal level이 서로 다른 프레임(e.g. I, P, B 프레임)들을 포함한다. 이때 사용자의 요청에 의해 트릭 모드 및 랜덤 액세스 상황이 발생하게 되면, 각 moof 내에서 일부의 sample 그룹만(e.g sub-fragment)을(e.g. I 프레임 그룹) 재생한다. 이를 지원하기 위하여 moof에 속한 sample들 중 일부 그룹에 접근 가능하도록 지원하는 인덱스 정보의 추가가 필요하다.

[0115] 다음의 활용 예는 이를 지원하기 위한 방법을 제시한다.

[0116] **1. 제1활용 예 (sidx_extension)**

[0117] 하기 <표 3>은 sidx extension을 통한 방법을 살펴보기 위해 제안될 수 있는

[0118] syntax의 일 예를 보이고 있다.

표 3

[0119]

```
aligned(8) class SegmentIndexBox extends FullBox('sidx', version, 0) {
        unsigned int(32) reference_track_ID;
        unsigned int(16) track_count;
        unsigned int(16) reference_count;
        for (i=1; i<= track_count; i++)
        {
                unsigned int(32)        track_ID;
                if (version==0)
                {
                        unsigned int(32)        decoding_time;
                } else
                {
                        unsigned int(64)        decoding_time;
                }
        }
        for(i=1; i <= reference_count; i++)
        {
                bit (1)                         reference_type;
                unsigned int(31)        reference_offset;
                bit(1)                          contains_level; //indication flag for level,
added
                unsigned int(31)        subsegment_duration; //reduce one bit for
contains_level, modified
                bit(1)                          contains_RAP;
                unsigned int(31)        RAP_delta_time;

                if (contains_level) //added
                {
                        unsigend int(8)         assemble_type; //define assemble type, add
                        unsigned int(16) level_count; //number of level(e.g. temporal id),
added
                        for(i=1; i <= level_count; i++) //added
                        {
                                unsigned int(8)         level; //level(e.g. temporal id level)
define, added
                                bit(1)                          reserved_bit; //reserved
                                unsigned int(31)        level_offset; //offest, added
                                if(assemble_type == 0x01 || assemble_type == 0x03)
                                {
                                        unsigned int(16)        offset_count //define number
of samples in level, added
                                        for(j=1; j <= offset_count; j++) //added
                                        {
                                                unsigned int(32)        offset; //offset from
level_offset, added
                                                unsigend int(32)        size; //size of each
sample in level, added
                                        }
                                )
                        }
                }
        }
}
```

[0120] 상기 <표 3>에서의 변수와 그 의미는 다음과 같이 정리될 수 있다.

[0121] *contains_level*: sub-fragment 단위의 인덱스 정보를 포함하는지 여부를 나타내는 플래그 비트

[0122] *assemble_type*: 각 moof의 미디어 샘플의 배치방법에 대한 구별자

[0123] 하기의 <표 4>는 assemble_type의 사용 예를 보이고 있다.

표 4

[0124]

assemble_type	Definition
0x00	*Sample group box(e.g. tele box)*
0x01	Assembled by equal temporal_level(e.g. IIPPBB…)
0x02	Stereoscopic plain(e.g. LRLRLR…)
0x03	Assembled by view(e.g. LLLRRR…)
0x04	SVC plain(e.g. BEBEBE…)
0x05	Assembled by equal scalability type(e.g. BBBEEE…)
0x06 ~	Reserved

[0125] *level_count*: fragment 내 레벨의 총 수

[0126] *level*: 각 레벨을 정의함. 낮은 숫자일수록 높은 우선순위를 의미함.

[0127] *level_offset*: 각각의 레벨에 대한 위치 정보

[0128] *offset_count*: 각 sample 단위 액세스가 필요한 경우, 이를 지원하기 위한 offset_count

[0129] *offset*: 각 샘플의 위치 정보

[0130] *size*: 샘플별 크기

[0131] *reserved_bit*: 확장을 위한 reserved_bit

[0132] 상기와 같은 sidx_extension을 통해 사용자는 특정 레벨의 sample 그룹에 직접 접근이 가능해지며 이를 통해 트릭 모드, Picture in Picture (PIP), 네트워크 환경에 따른 rate-adaptation을 보다 효과적으로 지원할 수 있다.

[0133] 다음의 예제에서는 하나 이상의 우선순위(즉, 레벨)를 포함하는 moof에서(e.g. SVC, Stereoscopic, MVC, etc) 우선순위에 따른 sub-fragment의 인덱스 정보를 제공하기 위한 방법을 보인다.

[0134] 도 12의 상단에서는 Stereoscopic 영상의 좌, 우 영상에 대한 sample들이 교차 배치된 스트림을 나타낸다. 하단에서는 보다 효과적인 인덱스 및 접근을 위해 시점에 따라 재정렬하고, 이를 다시 Temporal level에 따라 재정렬한 결과를 나타내고 있다.

[0135] 이와 같이 하나 이상의 레벨을 기술하기 위해서는 상기 <표 3>에서 제안하는 sidx_extension을 통해 시점에 따른 sub-fragment를 인덱싱한 뒤, 추가 확장을 위한 비트(reserved_bit)를 통해 temporal level에 따른 sub-fragment의 인덱스 정보를 제공한다.

[0136] **2. 제2활용 예(Sample group index box extension)**

[0137] sample group index box(sgix)는 각 moof 내에서 sample group(sub-fragment)에 대한 인덱스를 제공하기 위해 존재하는 box를 의미한다. 앞선 sidx_extension 방법과 동일하게 각 sample group(sub-fragment)에 대하여 샘플 그룹의 type과 각 그룹에 대한 level 정보를 추가한다.

[0138] 하기 <표 5>는 Sample group index box에 우선 순위 정보를 추가한 syntax의 일 예를 보이고 있다.

표 5

[0139]

```
aligned(8) class SampleGroupIndexBox
        extends FullBox('sgix', 0, 0) {
        unsigned int(32)                fragment_count;
                unsinged int(16)        level_count;
                unsigned int(8)         fragment_type;
                for( i=0; i < fragment_count; i++) {
                        for ( j=1; j < level_count; j++) {
                                unsigned int(8)         level;
                                unsigned int(32)        offset;
                                unsigned int(32)        size;
                        }
                }
}
```

[0140] *fragement_type*: 각 sample group(sub-fragment)의 구성 방법을 나타내며, 앞선 sidx_extension에 사용된 assemble_type과 동일하다.

[0141] *level*: 각 sample group(sub-fragment)에 대한 level을 정의한다. 낮은 숫자일수록 높은 우선순위를 의미한다.

offset: 각 샘플에 대한 크기를 나타낸다.

size: 각 샘플의 오프셋 정보를 나타낸다.

[0142] **3. 제3활용 예(sidx_extension)**

[0143] 하기 <표 6>은 sidx 확장을 통한 방법에 따른 syntax의 일 예를 보이고 있다.

표 6

[0144]

```
aligned(8) class SegmentIndexBox extends FullBox('sidx', version, 0) {
        unsigned int(32) reference_track_ID;
        unsigned int(16) track_count;
        unsigned int(16) reference_count;
        for (i=1; i<= track_count; i++)
        {
                unsigned int(32)        track_ID;
                if (version==0)
                {
                        unsigned int(32)        decoding_time;
                } else
                {
                        unsigned int(64)        decoding_time;
                }
        }
        for(i=1; i <= reference_count; i++)
        {
                bit (1)                         reference_type;
                unsigned int(31)        reference_offset;
                bit(1)                          contains_level; //indication flag for level,
added
                unsigned    int(31)        subsegment_duration;    //reduce    one    bit    for
contains_level, modified
                bit(1)                          contains_RAP;
                unsigned int(31)        RAP_delta_time;

                if (contains_level) //added
                {
                        unsigend int(8)         assemble_type; //define assemble type, add
                        unsigned  int(16)  level_count;  //number  of  level(e.g.  temporal  id),
added
                        for(i=1; i <= level_count; i++) //added
                        {
                                unsigned int(8)         level; //level(e.g. temporal id level)
define, added
                                bit(1)                          reserved_bit; //reserved
                                unsigned int(31)        level_offset; //offest, added
                                if(assemble_type == 0x01 || assemble_type == 0x03)
                                {
                                        unsigned  int(16)       offset_count  //define  number
of samples in level, added
                                        for(j=1; j <= offset_count; j++) //added
                                        {
                                                unsigned  int(32)       offset;  //offset  from
level_offset, added
                                                unsigend  int(32)       size;  //size  of  each
sample in level, added
                                        }
                                )
                        }
                }
        }
}
```

[0145] 상기 <표 6>에서 사용되는 변수와 그 의미는 다음과 같다.

[0146] *contains_level*: sub-fragment 단위의 인덱스 정보를 포함하는지 여부를 나타내는 플래그 비트

[0147] *assemble_type*: 정수형 값을 갖는 변수. 샘플 그룹의 타입을 구분하는 구분자로 활용된다. 여기서 샘플 그룹은 그 특성에 따라 구성될 수 있다. 예를 들어, 크게 비디오와 오디오로 구분 가능하다.

[0148] 이때 비디오는 목적에 따라 더 세분화된 샘플 그룹의 구성이 가능하다. 예를 들어, 랜덤 엑세스 및 트릭 모드를 지원하기 위해 독립적인 디코딩이 가능한 샘플만으로 샘플 그룹으로 구성할 수 있다. 시간의 확장성 (Temporal scalability)을 지원하는 비디오의 경우, 같은 temporal 레벨의 샘플만으로 샘플 그룹 구성이 가능하다. 스테레오스코픽 (Stereoscopic) 비디오는 시점에 따른 구성이 가능하며, MVC 역시 View id를 통한 구성이 가능하다.

[0149] *level_count*: moof 내 레벨의 총 수

[0150] *level*: 각 레벨을 정의함. 낮은 숫자일수록 높은 우선순위를 의미함. level은 level_count의 의미에 따라 다르게 활용할 수 있다. 만약 level_count의 값이 높거나 낮음에 따라 우선순위를 부여한다면, level은 활용되지 않는다. 그러나 level_count의 순서와 관계없이 level을 부여하고자 하는 경우, level 필드는 의미를 갖는다.

[0151] *level_offset*: 각각의 레벨에 대한 위치 정보

[0152] *offset_count*: 각 sample 단위 액세스가 필요한 경우, 이를 지원하기 위한 offset_count

[0153] *offset*: 각 샘플의 위치 정보

[0154] *size*: 샘플별 크기

[0155] *reserved_bit*: 확장을 위한 reserved_bit

[0156] **4. 제4활용 예(Sample group index box extension)**

[0157] sample group index box(sgix)는 각 moof 내에서 sample group(sub-fragment)에 대한 인덱스를 제공하기 위해 존재하는 박스 (box)를 의미한다. 앞선 sidx_extension 방법과 동일하게 각 sample group(sub-fragment)에 대하여 샘플 그룹의 타입과 각 그룹에 대한 레벨 정보, 그리고 샘플 그룹에 대한 크기 정보(SampleGroupSize)를 추가한다.

[0158] 하기 <표 7>은 샘플 그룹 인덱스 박스 (Sample Group Index Box)에 우선 순위 정보를 추가한 syntax의 일 예를 보이고 있다.

표 7

[0159]

```
aligned(8) class SampleGroupIndexBox
        extends FullBox('sgix', 0, 0) {
        unsigned int(32)                fragment_count;
                unsinged int(16)        level_count;
                unsigned int(8)         fragment_type;
                for( i=0; i < fragment_count; i++) {
                        for ( j=1; j < level_count; j++) {
                                unsigned int(8)                 level;
                                unsigned int(32)        SamepleGroupSize;
                        }
                }
}
```

[0160] 상기 <표 7>에서 사용되는 변수와 그 의미는 다음과 같다.

[0161] *fragement_type*: 정수형 값을 갖는 변수. 샘플 그룹의 타입을 구분하거나 혹은 샘플 그룹의 진입점을 지칭하는

구분자 등으로 활용된다. 이때 상기 샘플 그룹은 그 목적에 따라 다양하게 구성될 수 있다. 예를 들어 비디오와 오디오와 같이 미디어의 타입으로 구분 가능하다. 보다 세분화하는 경우, 비디오는 목적에 따라 더 작은 단위의 샘플 그룹의 구성이 가능하다. 예를 들어, 랜덤 액세스 및 트릭 모드를 지원하기 위해 독립적인 디코딩이 가능한 샘플들로 샘플 그룹을 구성할 수 있다. 시간의 확장성 (Temporal scalability)을 지원하는 비디오의 경우, 같은 temporal 레벨의 샘플만으로 샘플 그룹 구성이 가능하다. 스테레오스코픽 (Stereoscopic) 비디오는 시점에 따른 구성이 가능하며, MVC 역시 동 시점의 샘플들로 구성이 가능하다. Scalable Video에서 하나 이상의 scalability를 동시에 지원하는 경우, 이를 위한 하나 이상의 샘플 그룹이 동시에 존재할 수 있다. 상기에서 fragment_type을 통해 지칭하는 샘플 그룹은 샘플 그룹의 타입 혹은 샘플 그룹의 진입점으로 서술될 수 있다.

[0162] *level*: 각 sample group(sub-fragment)에 대한 레벨을 정의함. 낮은 숫자일수록 높은 우선순위를 의미함. level은 level_count의 의미에 따라 다르게 활용할 수 있다. 만약 level_count 값의 높거나 낮음에 따라 우선순위를 부여한다면, level은 활용되지 않는다. 그러나 level_count의 순서와 관계없이 level을 부여하고자 하는 경우, level 값은 의미를 갖는다. 하기의 <표 8>에서는 level을 활용하지 않는 경우의 syntax를 나타낸다.

[0163] *SampleGroupSize*: 각 샘플 그룹의 크기를 나타낸다. 각 샘플에 대한 크기와 오프셋 정보는 각 샘플 그룹에서 획득할 수 있으므로, 여기에서는 샘플 그룹 단위의 크기만을 제공한다.

[0164] 하기의 <표 8>은 샘플 그룹 인덱스 박스 (Sample Group Index Box)에 fragment_type만으로 샘플 그룹을 나타내는 경우의 syntax의 일 예를 보이고 있다.

표 8

[0165]

```
aligned(8) class SampleGroupIndexBox
        extends FullBox('sgix', 0, 0) {
        unsigned int(32)                fragment_count;
                unsinged int(16)        level_count;
                unsigned int(8)         fragment_type;
                for( i=0; i < fragment_count; i++) {
                        for ( j=1; j < level_count; j++) {
                                unsigned int(32)        SamepleGroupSize;
                        }
                }
}
```

[0166] 본 발명에서 제안하는 기술을 이용하여 Trick and Random Access(TRA)를 지원하는 과정은 아래와 같다.

[0167] 사용자로부터 요청받은 특정 시점으로 이동, 혹은 배속 재생과정에서, Media Presentation Description(MPD)를 이용하여 해당 시점에 대한 세그먼트 정보를 확인한다. 그리고 해당 세그먼트의 sidx(segment index box)를 이용하여 요청받은 시점에 해당하는 movie fragment정보를 확인하고, 제안하는 SampleGroupIndex box를 이용하여, 해당 movie fragment에서 독립적인 복호가 가능한 샘플에 접근한다.

[0168] level의 또 다른 활용 예로는 샘플 그룹 식별자로의 활용이다. 만일 하나 이상의 샘플 그룹을 식별자를 이용하여 관계 지어야 한다면, level을 통해 동일한 값을 부여함으로써 가능하다. 예를 들어 스테레오스코픽 (Stereoscopic) 영상의 좌, 우 시점의 샘플 그룹이 각각 존재하는 경우, 이를 관계짓기 위하여 같은 값을 부여한다. 이와 같은 방법으로 SVC, MVC와 같은 응용에서도 level을 이용하여 샘플 그룹간의 관계를 정의할 수 있다.

[0169] level이 활용되지 않는 상황에서 스테레오스코픽 (Stereoscopic) 서비스의 샘플 그룹간 관계를 기술하기 위해 본 발명에서는 추가적인 새로운 box를 도입한다. 예를 들어 스테레오스코픽 (Stereoscopic) 서비스의 TRA를 지원하기 위해 본 발명에서 제안하는 box를 활용하는 경우, 상기 fragment_type과 SampleGroupSize 정보를 제공한다. 이를 지원하기 위해서는 각 샘플 그룹들 간의 시간적, 공간적 관계를 dependency_ID와 샘플 그룹 ID를 이용하여 명시할 수 있어야 한다.

[0170] **5. 제5활용 예(Sub fragment index box extension)**

[0171] Sub Fragment Index Box(sfix)는 상기 제 2 및 제 4 활용 예에서 볼 수 있는 바와 같이, sub-sample group 또는 sub-fragment에 대한 인덱스 정보를 제공한다. 본 발명에서 제안하는 바와 같이, 사용자의 요구, 또는 네트워크 환경과 같은 원인으로 인하여 fragment의 일부인 sub-fragment 또는 sub-sample group에 접근하는 상황을 지원하기 위하여 하기 <표 9>와 같이 sfix를 확장한다.

[0172] 하기 <표 9>는 sub-fragment 또는 sub-sample group 인덱스 정보를 제공하기 위한 sfix extension의 syntax의 일 예를 보이고 있다.

표 9

[0173]
```
aligned(8) class SubFragmentIndexBox
        extends FullBox('sfix', 0, flags) {
                        unsigned int(32)              fragment_count;
                        unsigned int(8)         fragment_level_count;
                        for ( j=1; j < fragment_level_count) {
                                if ((flags & 1) == 0)
                                        unsigned int(32) track_id;
                                else
                                        unsigned int(32) sub_track_id;
                        }
                        for( i=0; i < fragment_count; i++)
                                for ( j=0; j < fragment_level_count; j++)
                                        unsigned int(32) accumulated_level_size;
}
```

[0174] 상기 <표 9>와 같이 sfix는 각 sub-fragment 또는 sub-sample group에 대한 인덱스 정보를 제공한다. SVC, MVC, Trick and Random Access(TRA) 상황과 같이 fragment의 일부를 조합하거나, 일부만을 활용하는 상황(예를 들어, TRA를 위하여 fragment 중에서 독립적인 복호가 가능한 sub-fragment 또는 sub-sample group에 접근해야 하는 상황, SVC와 같이 adaptation이 가능한 코덱의 경우, 특정 scalability를 갖는 sub-fragment 또는 sub-sample group만을 활용하는 경우)에서 각 sub-fragment 또는 sub-sample group에 접근하기 위한 식별자를 제공하여야 한다. 따라서 <표 9>와 같이 track_id 또는 sub_track_id를 제공함으로써 이를 가능하게 한다.

[0175] 상기 <표 9>에서의 변수와 그 의미는 다음과 같이 정리될 수 있다.

[0176] ***fragment_count***: sfix가 포함하는 fragment의 개수를 나타낸다.

[0177] ***fragment_level_count***: 각 fragment가 포함하는 level의 개수를 나타낸다.

[0178] ***track_id***: 해당 sub-fragment 또는 sub-sample group이 track_id에서 가리키는 track의 데이터 일부를 포함하고 있음을 의미한다.

[0179] ***sub_track_id***: 해당 sub-fragment 또는 sub-sample group이 sub_track_id에서 가리키는 sub_track의 데이터 일부를 포함하고 있음을 의미한다. 현재 sub-track의 id는 제공되지 않고 있으나, 향후 sub-track-id가 제안될 수 있으며, 그렇지 않을 경우, track 내에 포함된 sub-track의 개수에 따라 순차적으로 sub-track-id를 증가시키는 것을 약속하여 활용할 수 있다.

[0180] ***accumulated_level_size***: 각 sub-fragment 또는 sub-sample group에 대한 총 크기를 나타낸다.

[0181] 상기의 sfix extension에서는 각 sub-fragment 또는 sub-sample group의 구성 정보(제 1 내지 제4 활용 예에서 fragment_type 또는 assemble_type)를 포함하지 않는다. 그러므로 해당 sub-fragment 또는 sub-sample group의 종류는 각 sub-fragment box 또는 sub-sample group box에서 명시할 수 있을 것이다.

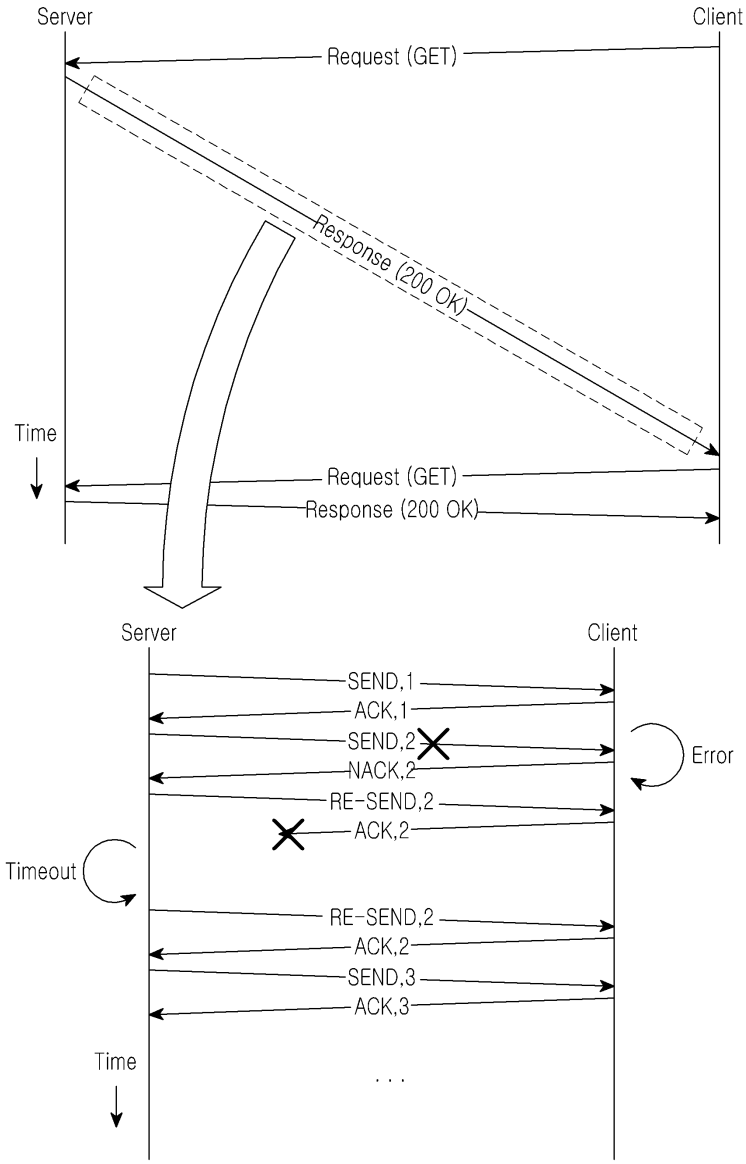
[0182] 경우에 따라 상기에서 제안되는 방법들의 조합을 통해서 상기 발명의 효과를 얻을 수도 있다.

[0183] 이상에서 본 발명의 실시 예에 대하여 상세하게 설명하였지만 본 발명의 권리범위는 이에 한정되는 것은 아니고 다음의 청구범위에서 정의하고 있는 본 발명의 기본 개념을 이용한 당업자의 여러 변형 및 개량 형태 또한 본

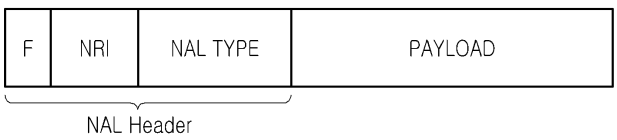
발명의 권리범위에 속하는 것이다.

도면

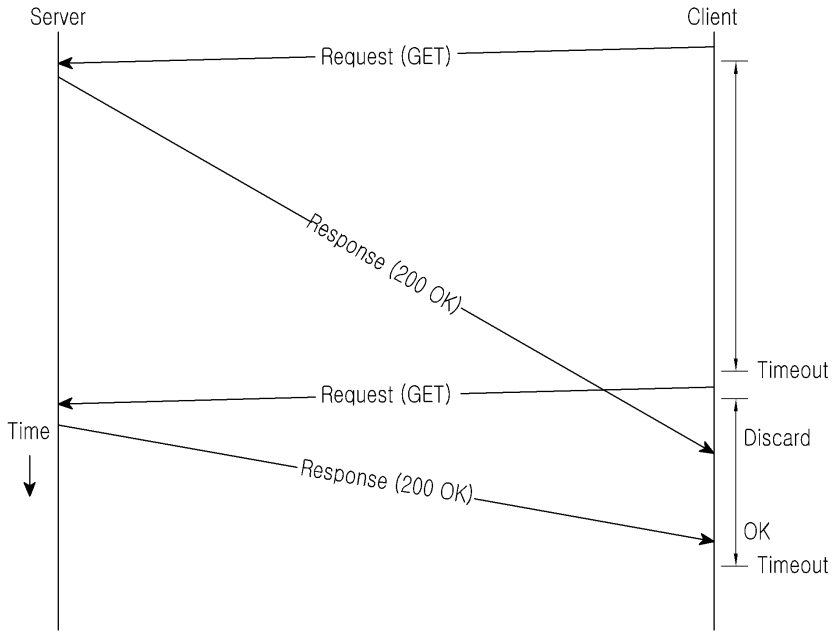
도면1



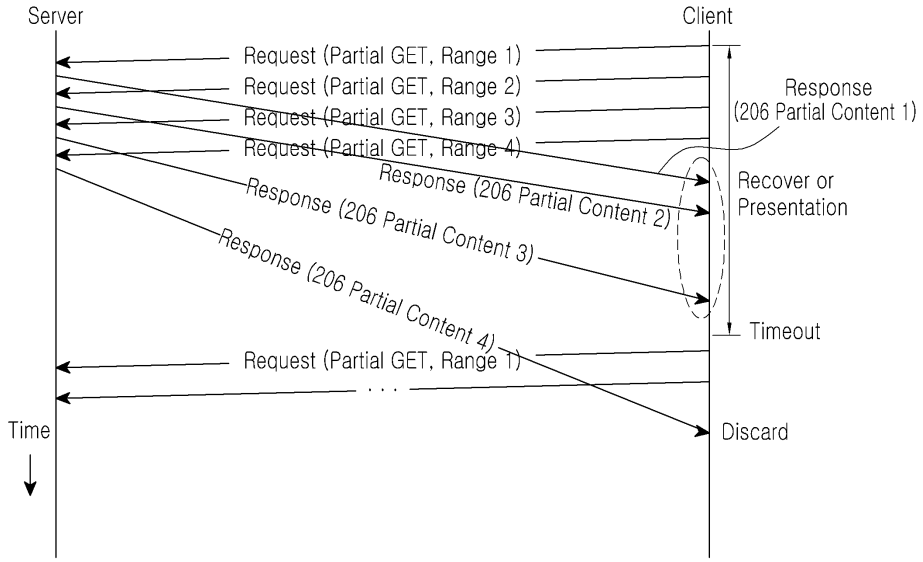
도면2



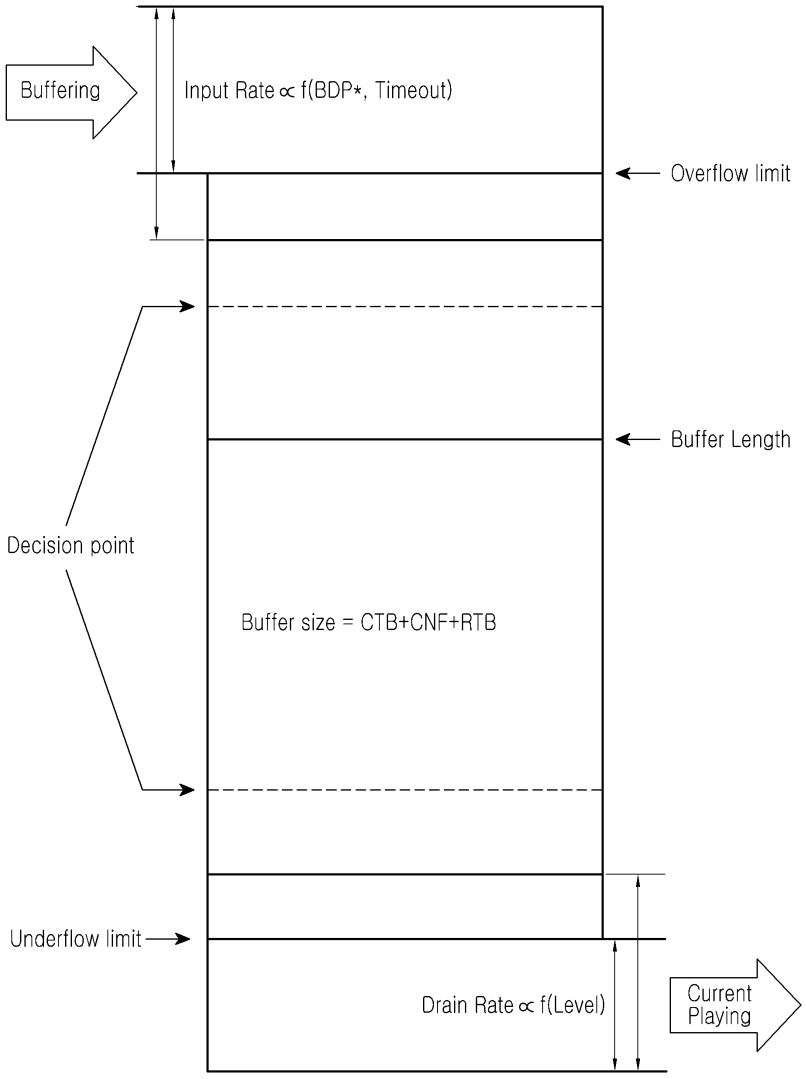
도면3


Server
Client
Request (GET)
Response (200 OK)
Timeout
Request (GET)
Time
Discard
Response (200 OK)
OK
Timeout

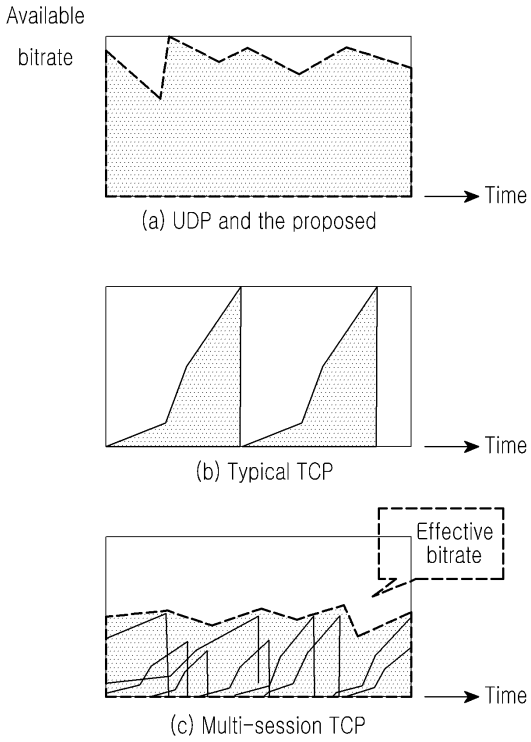
도면4


Server
Client
Request (Partial GET, Range 1)
Request (Partial GET, Range 2)
Request (Partial GET, Range 3)
Request (Partial GET, Range 4)
Response
(206 Partial Content 1)
Response (206 Partial Content 2)
Recover or
Presentation
Response (206 Partial Content 3)
Response (206 Partial Content 4)
Timeout
Request (Partial GET, Range 1)
. . .
Time
Discard

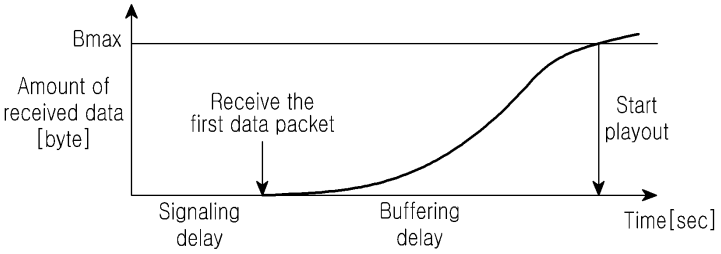
도면5


Buffering
Input Rate ∝ f(BDP*, Timeout)
Overflow limit
Buffer Length
Decision point
Buffer size = CTB+CNF+RTB
Underflow limit
Drain Rate ∝ f(Level)
Current Playing

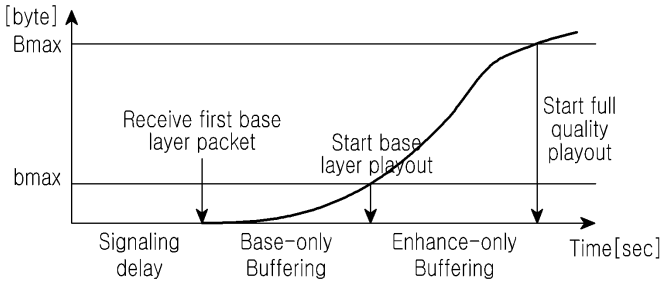
도면6


Available
bitrate
Time
(a) UDP and the proposed
Time
(b) Typical TCP
Effective
bitrate
Time
(c) Multi-session TCP

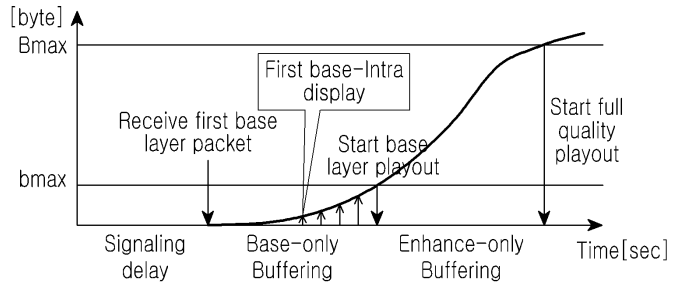
도면7


Bmax
Amount of
received data
[byte]
Receive the
first data packet
Start
playout
Signaling
delay
Buffering
delay
Time[sec]

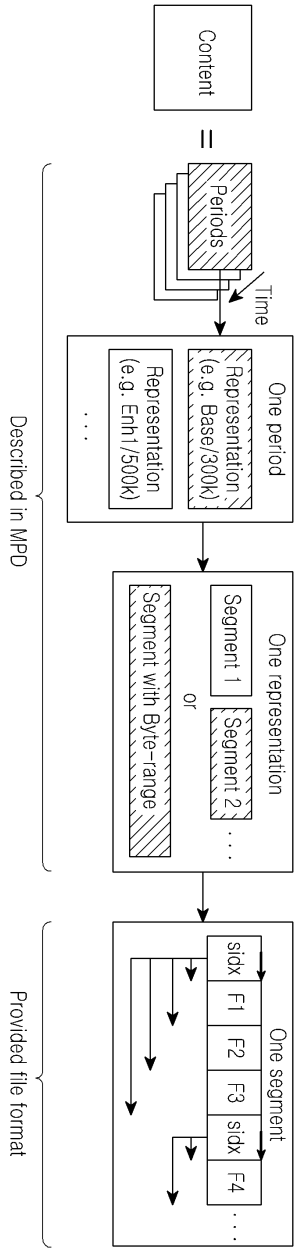
도면8


[byte]
Bmax
bmax
Receive first base
layer packet
Start base
layer playout
Start full
quality
playout
Signaling
delay
Base-only
Buffering
Enhance-only
Buffering
Time[sec]

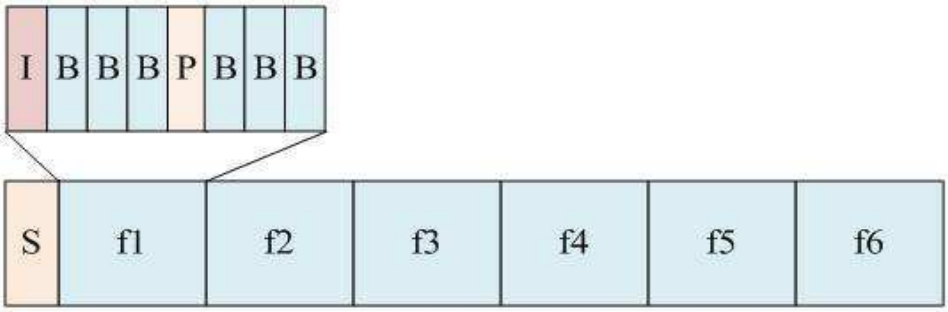
도면9


[byte]
Bmax
bmax
First base-Intra display
Receive first base layer packet
Start base layer playout
Start full quality playout
Signaling delay
Base-only Buffering
Enhance-only Buffering
Time[sec]

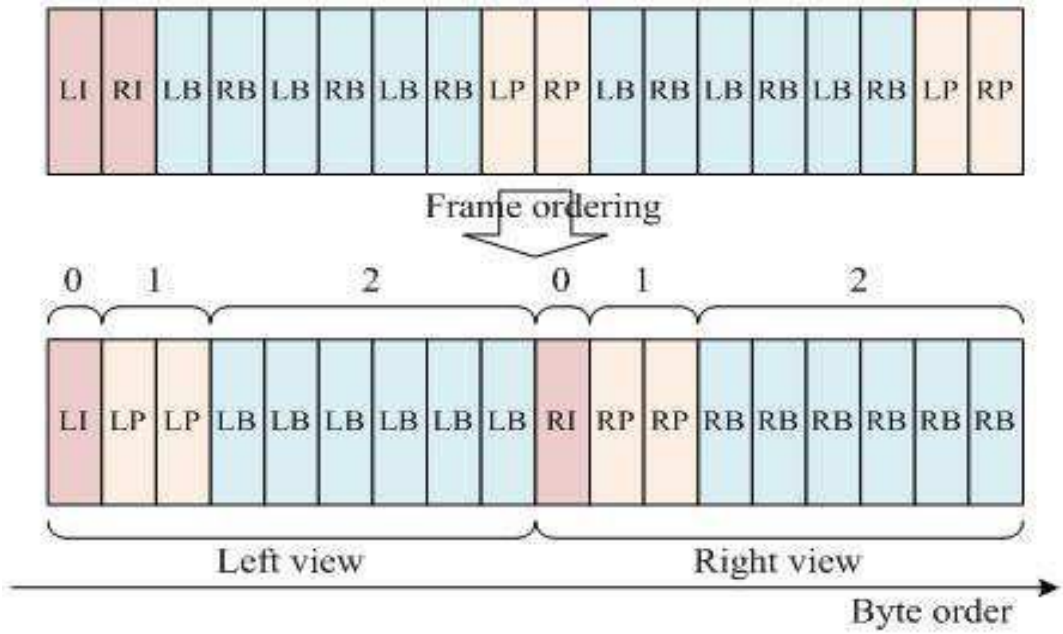
도면10


Content
Periods
Time
One period
Representation (e.g. Base/300k)
Representation (e.g. Enh1/500k)
One representation
Segment 1
Segment 2
or
Segment with Byte-range
One segment
sidx
F1
F2
F3
sidx
F4
Described in MPD
Provided file format

도면11



도면12



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 1항, 4항, 8항, 11항

【변경전】

상기 소정의 레인지에

【변경후】

상기 소정의 바이트 레인지에