(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년09월23일
(11) 등록번호 10-2024311
(24) 등록일자 2019년09월17일

(51) 국제특허분류(Int. Cl.)
H04L 29/06 (2006.01) *H04L 29/08* (2006.01)
H04N 21/262 (2011.01) *H04N 21/472* (2011.01)
(52) CPC특허분류
H04L 65/4084 (2013.01)
H04L 65/602 (2013.01)
(21) 출원번호 10-2018-7029384(분할)
(22) 출원일자(국제) 2014년07월11일
심사청구일자 2018년10월11일
(85) 번역문제출일자 2018년10월11일
(65) 공개번호 10-2018-0114964
(43) 공개일자 2018년10월19일
(62) 원출원 특허 10-2016-7003172
원출원일자(국제) 2014년07월11일
심사청구일자 2016년07월28일
(86) 국제출원번호 PCT/EP2014/064949
(87) 국제공개번호 WO 2015/004276
국제공개일자 2015년01월15일
(30) 우선권주장
1312561.2 2013년07월12일 영국(GB)
(뒷면에 계속)

(56) 선행기술조사문헌
US20120023251 A1*
ISO/IEC 23009-1 : Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats (2012.04.01.)*
Thomas Stockhammer, "MPEG's Dynamic Adaptive Streaming over HTTP (DASH) - Enabling Formats for Video Streaming over the Open Internet" (2011. 11. 22.)*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
캐논 가부시끼가이샤
일본 도꾜도 오오따꾸 시모마루꼬 3쪼메 30방 2고
(72) 발명자
파블레 유엔
프랑스 에프-35390 라 도미늘래 라 룰래
벨소르 로맹
프랑스 에프-35000 렌느 뤼 드 로비엉 3
(뒷면에 계속)
(74) 대리인
장수길, 이중희

전체 청구항 수 : 총 20 항 심사관 : 나용수

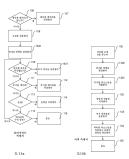
(54) 발명의 명칭 **푸시 메시지 제어를 이용하는 적응적 데이터 스트리밍 방법**

(57) 요 약

통신 네트워크를 통해 스트리밍을 관리하는 방법들이 제공된다. 서버 및 디바이스들은 클라이언트 디바이스가 서버에 의한 데이터 푸시를 예상할 수 있도록 푸시 정책을 공유한다. 예상은 일부 푸시된 데이터의 전송을 초기에 취소할 수 있게 하며, 이에 따라 대역폭 소비를 감소시킨다. 공유된 푸시 정책은 서버 및 클라이언트 양자에

(뒷면에 계속)

대 표 도



내포될 수 있다. 실시예들에서, 이는 예를 들어, 미디어 표현 기술 파일에 삽입되거나 HTTP 헤더에 포함되는 것으로 서버에 의해 클라이언트에게 명확히 특정된다. 클라이언트는 또한 그 자신의 요건을 충족하기 위해 공유된 푸시 정책의 업데이트를 요청할 수 있다.

(52) CPC특허분류
H04L 65/608 (2013.01)
H04L 67/02 (2013.01)
H04N 21/26258 (2013.01)
H04N 21/47202 (2013.01)

(72) 발명자
마즈 프레데릭
프랑스 에프-35850 랑강 뤼 데 띠엘 6
우드라오고 나엘
프랑스 에프-35330 모르 드 브르따뉴 꾸뚜즈
드누알 프랑크
프랑스 에프-35190 생 도미누크 라 빌-에스-레
륄랑 에르베
프랑스 에프-35000 렌느 뤼 뒤아멜 49

(30) 우선권주장
1312547.1 2013년07월12일 영국(GB)
1410540.7 2014년06월12일 영국(GB)

명 세 서

청구범위

청구항 1

서버로부터 클라이언트로 미디어 데이터를 제공하는 방법으로서,

- 상기 클라이언트로부터, 초기화 정보 및 미디어에 대한 상기 클라이언트의 선호를 시그널링하는 파라미터들을 포함하고 MPD(Media Presentation Description)를 요청하기 위한 MPD 요청을 수신하는 단계 - MPD는 MPEG-DASH 표준에서 정의됨 -;

- 상기 클라이언트로부터의 상기 MPD 요청 내의 파라미터들에 따라, 상기 클라이언트로 푸시할 하나 이상의 세그먼트들을 결정하는 단계; 및

- 상기 클라이언트로부터의 상기 MPD 요청에 응답하여 상기 클라이언트로 결정된 상기 하나 이상의 세그먼트들을 푸시하는 단계를 포함하는 미디어 데이터를 제공하는 방법.

청구항 2

제1항에 있어서,

상기 결정하는 단계에서, 미디어 세그먼트들에 캡슐화된 미디어 스트림들을 나타내기 위해 필요한 메타데이터를 포함하는 초기화 세그먼트가 상기 클라이언트로 푸시되도록 결정되는, 미디어 데이터를 제공하는 방법.

청구항 3

제1항에 있어서,

상기 결정하는 단계에서, 일부 미디어 세그먼트들이 상기 클라이언트로 푸시되도록 결정되는, 미디어 데이터를 제공하는 방법.

청구항 4

제1항에 있어서,

상기 결정하는 단계에서, 초기화 세그먼트 및 일부 미디어 세그먼트들이 상기 클라이언트로 푸시되도록 결정되는, 미디어 데이터를 제공하는 방법.

청구항 5

제1항에 있어서,

상기 서버가 상기 파라미터들에 따라 세그먼트들을 푸시하려고 의도한다는 것을 나타내는 통지를 상기 클라이언트로 전송하는 단계를 더 포함하는 미디어 데이터를 제공하는 방법.

청구항 6

제1항에 있어서,

상기 파라미터들은 비디오의 높이에 관한 파라미터를 포함하는, 미디어 데이터를 제공하는 방법.

청구항 7

제1항에 있어서,

상기 파라미터들은 대역폭에 관한 파라미터를 포함하는, 미디어 데이터를 제공하는 방법.

청구항 8

제1항에 있어서,

상기 파라미터들은 비디오 세그먼트를 위한 전송 레이트와 오디오 세그먼트를 위한 언어 정보 중 적어도 하나를 시그널링하는, 미디어 데이터를 제공하는 방법.

청구항 9

제1항에 있어서,

상기 MPD 요청에 응답하여 상기 클라이언트로 상기 MPD를 전송하는 단계를 더 포함하는 미디어 데이터를 제공하는 방법.

청구항 10

클라이언트에 의해 서버로부터 미디어 데이터를 수신하는 방법으로서,

- 초기화 정보 및 미디어에 대한 상기 클라이언트의 선호를 시그널링하는 파라미터들을 포함하고 MPD를 요청하기 위한 MPD 요청을 상기 서버로 전송하는 단계 - MPD는 MPEG-DASH 표준에서 정의됨 -; 및

- 상기 MPD 요청에 응답하여 상기 서버로부터 푸시된 하나 이상의 세그먼트들을 수신하는 단계를 포함하고,

푸시된 상기 하나 이상의 세그먼트들은 상기 MPD 요청에 포함된 상기 파라미터들에 따라 상기 서버에 의해 결정되는, 미디어 데이터를 수신하는 방법.

청구항 11

제10항에 있어서,

상기 수신하는 단계에서, 미디어 세그먼트들에 캡슐화된 미디어 스트림들을 나타내기 위해 필요한 메타데이터를 포함하고, 상기 서버로부터 푸시된 초기화 세그먼트가 수신되는, 미디어 데이터를 수신하는 방법.

청구항 12

제10항에 있어서,

상기 수신하는 단계에서, 상기 서버로부터 푸시된 일부 미디어 세그먼트들이 수신되는, 미디어 데이터를 수신하는 방법.

청구항 13

클라이언트로 미디어 데이터를 제공하는 장치로서,

하드웨어 프로세서; 및

상기 하드웨어 프로세서에 의해 실행되도록 구성된 하나 이상의 프로그램들을 저장하는 메모리를 포함하고,

상기 하나 이상의 프로그램들은,

- 상기 클라이언트로부터, 초기화 정보 및 미디어에 대한 상기 클라이언트의 선호를 시그널링하는 파라미터들을 포함하고 MPD를 요청하기 위한 MPD 요청을 수신하고 - MPD는 MPEG-DASH 표준에서 정의됨 -,

- 상기 클라이언트로부터의 상기 MPD 요청 내의 파라미터들에 따라, 상기 클라이언트로 푸시할 하나 이상의 세그먼트들을 결정하고,

- 상기 클라이언트로부터의 상기 MPD 요청에 응답하여 상기 클라이언트로 결정된 상기 하나 이상의 세그먼트들을 푸시하기 위한

명령어들을 포함하는, 미디어 데이터를 제공하는 장치.

청구항 14

제13항에 있어서,

상기 결정을 할 때, 미디어 세그먼트들에 캡슐화된 미디어 스트림들을 나타내기 위해 필요한 메타데이터를 포함하는 초기화 세그먼트가 상기 클라이언트로 푸시되도록 결정되는, 미디어 데이터를 제공하는 장치.

청구항 15

제13항에 있어서,

상기 결정을 할 때, 일부 미디어 세그먼트들이 상기 클라이언트로 푸시되도록 결정되는, 미디어 데이터를 제공하는 장치.

청구항 16

서버로부터 미디어 데이터를 수신하는 장치로서,

하드웨어 프로세서; 및

상기 하드웨어 프로세서에 의해 실행되도록 구성된 하나 이상의 프로그램들을 저장하는 메모리를 포함하고,

상기 하나 이상의 프로그램들은,

- 초기화 정보 및 미디어에 대한 클라이언트의 선호를 시그널링하는 파라미터들을 포함하고 MPD를 요청하기 위한 MPD 요청을 상기 서버로 전송하고 - MPD는 MPEG-DASH 표준에서 정의됨 -,

- 상기 MPD 요청에 응답하여 상기 서버로부터 푸시된 하나 이상의 세그먼트들을 수신하기 위한

명령어들을 포함하고,

푸시된 상기 하나 이상의 세그먼트들은 상기 MPD 요청에 포함된 상기 파라미터들에 따라 상기 서버에 의해 결정되는, 미디어 데이터를 수신하는 장치.

청구항 17

제16항에 있어서,

상기 수신을 할 때, 미디어 세그먼트들에 캡슐화된 미디어 스트림들을 나타내기 위해 필요한 메타데이터를 포함하고, 상기 서버로부터 푸시된 초기화 세그먼트가 수신되는, 미디어 데이터를 수신하는 장치.

청구항 18

제16항에 있어서,

상기 수신을 할 때, 상기 서버로부터 푸시된 일부 미디어 세그먼트들이 수신되는, 미디어 데이터를 수신하는 장치.

청구항 19

컴퓨터로 하여금 미디어 데이터를 클라이언트로 제공하는 방법을 실행하도록 하기 위한 프로그램을 저장하는 비일시적 컴퓨터-판독가능 저장 매체로서,

상기 방법은,

- 상기 클라이언트로부터, 초기화 정보 및 미디어에 대한 상기 클라이언트의 선호를 시그널링하는 파라미터들을 포함하고 MPD를 요청하기 위한 MPD 요청을 수신하는 단계 - MPD는 MPEG-DASH 표준에서 정의됨 -;

- 상기 클라이언트로부터의 상기 MPD 요청 내의 파라미터들에 따라, 상기 클라이언트로 푸시할 하나 이상의 세그먼트들을 결정하는 단계; 및

- 상기 클라이언트로부터의 상기 MPD 요청에 응답하여 상기 클라이언트로 결정된 상기 하나 이상의 세그먼트들을 푸시하는 단계

를 포함하는, 비일시적 컴퓨터-판독가능 저장 매체.

청구항 20

컴퓨터로 하여금 클라이언트로부터 미디어 데이터를 수신하는 방법을 실행하도록 하기 위한 프로그램을 저장하는 비일시적 컴퓨터-판독가능 저장 매체로서,

상기 방법은,

- 초기화 정보 및 미디어에 대한 상기 클라이언트의 선호를 시그널링하는 파라미터들을 포함하고 MPD를 요청하기 위한 MPD 요청을 서버로 전송하는 단계 - MPD는 MPEG-DASH 표준에서 정의됨 -; 및

- 상기 MPD 요청에 응답하여 상기 서버로부터 푸시된 하나 이상의 세그먼트들을 수신하는 단계

를 포함하고,

푸시된 상기 하나 이상의 세그먼트들은 상기 MPD 요청에 포함된 상기 파라미터들에 따라 상기 서버에 의해 결정되는, 비일시적 컴퓨터-판독가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 출원은 다음의 영국 특허 출원들(2013년 7월 12자로 출원된 출원번호 제1312547.1호 및 제1312561.2호, 2014년 6월 12일로 출원된 출원번호 제1410540.7호)로부터 우선권을 주장하며, 이들 모두는 참조로서 본 명세서에 포함된다.

[0002] 본 발명은 HTTP 통신 네트워크들을 통한 데이터 스트리밍에 관한 것이다.

[0003] 보다 상세하게는, 본 발명은 네트워크 제약 조건들을 충족시키기 위한 적응적 데이터 스트리밍에 관한 것이다. 본 발명은 DASH 네트워크들에서 애플리케이션을 가질 수 있다.

[0004] DASH(Dynamic Adaptive Streaming over HTTP의 머리글자)는 HTTP를 통해 미디어 콘텐츠 스트리밍(전형적으로 오디오/비디오 콘텐츠)을 허용하는 통신 표준이다. DASH에 따라, 미디어 표현들은 XML 파일들로 기술되고, 이는 "미디어 표현 기술(media presentation description)" 파일들(이하 MPD)로 불린다. MPD 파일들은 이들이 미디어 콘텐츠들의 전달을 요청하고 제어할 수 있는 정보를 클라이언트 디바이스들에 제공한다.

배 경 기 술

[0005] HTTP를 통한 미디어 스트리밍의 일반적인 원리는 도 3에 예시된다. HTTP를 통한 적응적 미디어 스트리밍을 위한 신규 프로토콜들 및 표준들의 대부분은 이러한 원리에 기초한다.

[0006] 미디어 서버(300)는 데이터를 클라이언트(310)에게 스트리밍한다. 미디어 서버는 미디어 표현들을 저장한다. 예를 들어, 미디어 표현(301)은 오디오와 비디오 데이터를 포함한다. 오디오와 비디오는 동일 파일에 인터리브될 수 있다. 미디어 표현이 구축되는 방법은 도 4a를 참고하여 이하 설명된다. 미디어 표현은 예를 들어, 독립적으로 어드레싱 및 다운로드될 수 있는 MP4 세그먼트들과 같은 작은 독립적이고 연속적인 시간 세그먼트들(302a, 302b, 및 302c)로 시간적으로 분할된다. 이러한 시간 세그먼트들 각각에 대한 미디어 콘텐츠의 어드레스(HTTP URL)들을 다운로드하는 것은 서버에 의해 클라이언트에 대해 설정된다. 오디오/비디오 미디어 콘텐츠의 각각의 시간 세그먼트는 하나의 HTTP 어드레스와 연관된다.

[0007] 미디어 서버는 또한, 미디어 콘텐츠 특성(예를 들어, 미디어의 유형: 오디오, 비디오, 오디오-비디오, 텍스트 등), 인코딩 포맷(예를 들어, 비트레이트, 타이밍 정보 등), 시간 미디어 세그먼트들 및 연관된 URL들의 리스트를 포함하는 미디어 표현의 콘텐츠를 기술하는 매니페스트 파일(manifest file) 문서(304)(도 5를 참고하여 이하 설명된다)을 저장한다. 대안적으로, 문서는 시간 미디어 세그먼트들 및 연관된 URL들의 명시적 리스트를 재건하는 것을 가능하게 하는 템플릿 정보를 포함한다. 이 문서는 XML(eXtensible Markup Language)을 이용하여 기입될 수 있다.

[0008] 매니페스트 파일은 클라이언트에 전송된다. 단계 305 동안 매니페스트 파일이 수신시, 클라이언트는 미디어 콘텐츠들의 시간 세그먼트들과 HTTP 어드레스들 사이의 연관을 통지받는다. 또한, 매니페스트 파일은 미디어 표현(본 예에서는 인터리브된 오디오/비디오)의 콘텐츠에 관한 정보를 클라이언트에 제공한다. 정보는 해상도, 비트 레이트 등을 포함할 수 있다.

[0009] 수신된 정보에 기초하여, 클라이언트의 HTTP 클라이언트 모듈(311)은 매니페스트 파일에 기술된 미디어 콘텐츠의 시간 세그먼트들을 다운로드하기 위한 HTTP 요청들(306)을 낼 수 있다. 서버의 HTTP 응답들(307)은 요청된 시간 세그먼트들을 운반한다. HTTP 클라이언트 모듈(311)은 응답들로부터 시간 미디어 세그먼트들을 추출하고, 그들을 미디어 엔진(312)의 입력 버퍼(307)에 제공한다. 최종적으로, 미디어 세그먼트들은 각각의 단계 308 및

309 동안 디코딩되고 표시된다.

[0010] 미디어 엔진(312)은 다음 시간 세그먼트들에 대한 요청들이 적절한 시간에 발행되게 하기 위해 DASH 제어 엔진(313)과 상호작용한다. 다음 세그먼트는 매니페스트 파일로부터 식별된다. 요청이 발행되는 시간은 수신 버퍼(307)가 가득 찼는지에 달려있다. DASH 제어 엔진(313)은 버퍼가 오버로드 또는 완전히 비어 있는 것을 방지하기 위해 버퍼를 제어한다.

[0011] 미디어 표현과 매니페스트 파일의 발생은 도 4a를 참고하여 설명된다. 단계들 400 및 401 동안, 오디오 및 비디오 데이터가 획득된다. 다음에, 오디오 데이터는 단계 402 동안 압축된다. 예를 들어, MP3 표준이 이용될 수 있다. 또한, 비디오 데이터는 단계 403 동안 동시에 압축된다. 예를 들어, MPEG4, MPEG/AVC, SVC, HEVC, 또는 스케일러블 HEVC와 같은 비디오 압축 알고리즘들이 이용될 수 있다. 오디오 및 비디오 데이터의 압축이 수행될 때, 오디오 및 비디오 기본 스트림들(404, 405)이 이용 가능하다. 기본 스트림들은 단계 406 동안 글로벌 미디어 표현 내에 캡슐화된다. 예를 들어, ISO BMFF 표준(또는 ISO BMFF 표준의 AVC, SVC, HEVC로의 확대, HEVC의 스케일러블 확대 등)은 글로벌 미디어 표현으로서 인코딩된 오디오 및 비디오 기본 스트림들의 콘텐츠를 기술하는데 사용될 수 있다. 그렇게 획득된 캡슐화된 미디어 표현(407)은 단계 408 동안 XML 매니페스트 파일(409)을 생성하기 위해 사용된다. 비디오 데이터(401)와 오디오 데이터(400)의 여러 표현은 미디어 표현(407)에서 획득되고, 압축되고, 캡슐화되고, 기술될 수 있다.

[0012] 도 4b에 예시된 MPEG/DASH 스트리밍 프로토콜의 특정한 경우에 대해서, 매니페스트 파일은 "미디어 표현 기술"(또는 "MPD" 파일)로 불린다. 파일의 기초 요소는 프로필 또는 스키마와 같은 DASH 정보에 더해진 모든 표현에 적용하는 속성들을 포함하는 MPD 요소이다. 미디어 표현은 Period 요소에 의해 표현되는 시간 기간들로 분할된다. MPD 파일(410)은 각각의 시간 기간과 관련된 모든 데이터를 포함한다. 이런 정보를 수신함으로써, 클라이언트는 각각의 시간 기간에 대한 콘텐츠를 알게 된다. 각각의 Period(411) 동안, AdaptationSet 요소들이 정의된다.

[0013] 가능한 조직은 표현에 포함되는 미디어 유형당 하나 이상의 AdaptationSet를 갖기 위한 것이다. 비디오와 관련된 AdaptationSet(412)는 서버에서 이용 가능한 인코딩된 비디오들의 상이한 가능한 표현들에 대한 정보를 포함한다. 각각의 표현은 Representation 요소에 기술된다. 예를 들어, 제1 표현은 640x480의 공간 해상도로 인코딩되고 500k비트/s의 비트 레이트로 압축된 비디오일 수 있다. 제2 표현은 동일한 비디오이지만 250k비트/s의 비트 레이트로 압축될 수 있다.

[0014] 클라이언트가 비디오와 관련된 HTTP 어드레스들을 안다면, 각각의 비디오는 이후 HTTP 요청들에 의해 다운로드될 수 있다. 각 표현의 콘텐츠와 HTTP 어드레스들 사이의 연관은 기술의 추가 레벨(시간 세그먼트)을 사용하여 이루어진다. 각각의 비디오 표현은 시간 세그먼트들(413)(전형적으로 수초)로 분할된다. 각각의 시간 세그먼트는 HTTP 어드레스(URL 또는 일 바이트 범위를 가진 URL)을 통해 액세스 가능한 서버에 저장된 콘텐츠를 포함한다. 여러 요소는 MPD 파일에서 시간 세그먼트들을 기술하기 위해 이용될 수 있다: SegmentList, SegmentBase 또는 SegmentTemplate.

[0015] 또한, 특정 세그먼트가 이용 가능하다: 초기화 세그먼트. 초기화 세그먼트는 캡슐화된 비디오 스트림을 기술하는 MP4 초기화 정보를 포함한다(비디오가 ISO BMFF 또는 그 확장들을 사용하여 캡슐화되는 경우). 예를 들어, 이는 클라이언트가 비디오와 관련되는 디코딩 알고리즘들을 예시하는 데 도움을 준다.

[0016] 초기화 세그먼트와 미디어 세그먼트들의 HTTP 어드레스들은 MPD 파일에 나타난다.

[0017] 도 5에는 예시적인 MPD 파일이 도시된다. 2개의 미디어는 도시된 MPD 파일에 기술된다. 제1 미디어는 영어 오디오 스트림이고, 제2 미디어는 비디오 스트림이다. 영어 오디오 스트림은 AdaptationSet 태그(500)를 이용하여 도입된다. 2개의 대안적인 표현은 이 오디오 스트림에 이용 가능하다:

[0018] - 제1 표현(501)은 64000비트/초의 비트 레이트을 가진 기본 오디오 스트림으로 캡슐화된 MP4이다. (MP4 파싱 후) 이 기본 스트림을 다루기 위해 이용될 코덱은 값, 'mp4a.0x40'을 갖는 속성 코덱들에 의해 표준에서 정의된다. 이는 세그먼트 계층 구조에서 BaseURL 요소들의 연쇄에 의해 형성되는 어드레스에서 요청을 통해 액세스 가능하다: URI와 관련되는 <BaseURL>7657412348.mp4</BaseURL>. 'http://cdn1.example.com' 또는 'http://cdn2.example.com'에 의해 MPD 요소의 상위 레벨에서 정의되는 <BaseURL>(2개의 서버가 동일한 콘텐츠를 스트리밍하기 위해 이용 가능하다)은 절대 URI이다. 클라이언트는 이후 어드레스 'http://cdn1.example.com/7657412348.mp4' 또는 어드레스 'http://cdn2.example.com/7657412348.mp4'에 대한 요청으로부터 영어 오디오 스트림을 요청할 수 있다.

[0019] - 제2 표현(502)은 32000비트/초의 비트 레이트를 가진 기본 오디오 스트림으로 캡슐화된 MP4이다. 제1 표현(501)에 대한 설명과 동일한 설명들이 이루어질 수 있고, 클라이언트 디바이스는 이에 따라 하기 어드레스들 중 하나에서의 요청에 의해 이런 제2 표현(502)을 요청할 수 있다:

[0020] 'http://cdn1.example.com/3463646346.mp4' 또는

[0021] 'http://cdn2.example.com/3463646346.mp4'.

[0022] 비디오와 관련된 적응 세트(503)는 6개의 표현을 포함한다. 이들 표현은 상이한 공간 해상도들(320x240, 640x480, 1280x720) 및 상이한 비트 레이트들(초당 256000 내지 2048000 비트)을 가진 비디오들을 포함한다. 이러한 표현들 각각에서, 각각의 URL은 BaseURL 요소를 통해 연관된다. 따라서, 클라이언트는 추정된 대역폭, 화면 해상도 등과 같은 상이한 기준에 따라 동일한 비디오의 이러한 대안적인 표현들 중에서 선택할 수 있다(도 5에서는 명료함을 위해, 시간 세그먼트들로의 표현들의 분해가 예시되지 않음에 유의한다).

[0023] 도 5a는 DASH 클라이언트의 표준 거동을 도시한다. 도 5b는 도 4a에 도시된 방법에 사용되는 예시적인 매니페스트 파일(기술 파일 또는 MPD)의 트리 표현을 도시한다.

[0024] 스트리밍 세션을 시작할 때, DASH 클라이언트는 매니페스트 파일을 요청하는 것으로 시작된다(단계 550). 서버의 응답을 대기하고 매니페스트 파일을 수신한 후(단계 551), 클라이언트는 매니페스트 파일을 분석하고(단계 552), 그 환경에 적합한 AdaptationSets의 세트 AS_{ij}를 선택하고(단계 553), 이후 AdaptationSet AS_{ij} 내에서, 예를 들어, 그 대역폭, 디코딩 및 렌더링 능력들에 적절한 표현을 MPD에서 선택한다(단계 554).

[0025] DASH 클라이언트는 이후, 미디어 디코더들을 위한 초기화 정보로 시작할 것을 요청하기 위해 세그먼트들의 리스트를 미리 구축할 수 있다. 이 초기화 세그먼트는, 이것이 다수의 표현, 적응 세트들 및 기간들에 공통이거나 각각의 Representation에 특정하거나 심지어 제1 미디어 세그먼트에 포함될 수 있기 때문에, MPD에서 식별되어야 한다(단계 555).

[0026] 클라이언트는 이후 초기화 세그먼트를 요청한다(단계 556). 초기화 세그먼트가 수신될 때(단계 557), 디코더들은 개시된다(단계 558).

[0027] 클라이언트는 이후 세그먼트 기준으로 제1 미디어 데이터를 요청하고(단계 560), 실제로 디코딩하고 표시하기(단계 563) 전에 최소 데이터량을 버퍼링한다(단계 559에서의 조건에 기임함). MPD 다운로드와 제1 표시 프레임들 간의 이러한 다수의 요청/응답은 스트리밍 세션에 기동 지연을 도입한다. 이들 초기 단계 후에, DASH 스트리밍 세션은 표준 방식으로 계속된다, 즉, DASH 클라이언트는 미디어 세그먼트들을 번갈아 적응하고 요청한다.

[0028] 현재 DASH 버전은 매니페스트 파일들 내의 관심 영역(Region-Of-Interest)의 기술을 제공하지 않는다. 여러 접근법들이 이런 기술을 제안했다.

[0029] 특히, 미디어 콘텐츠들의 컴포넌트들은 SubRepresentation 요소들을 이용하여 기술될 수 있다. 이들 요소는 Representation에 삽입된 하나 또는 여러 컴포넌트의 특성들을 기술한다. 도 6에는 비디오의 컴포넌트로서 타일 트랙들을 기술하는 DASH 매니페스트 파일의 예를 도시한다. 간결성과 명료성을 위하여, 하나의 Period(600)만이 표현된다. 그러나, 후속 기간 요소들은 동일 방식으로 조직화될 것이다. 부분 601에서, 제1 적응 세트 요소는 스케일러블 비디오의 기저 계층을 기술하기 위해 이용된다. 예를 들어, 비디오는 스케일러블 SVC 또는 HEVC에 따라 인코딩된다. 부분 602에서, 제2 적응 세트는 스케일러블 비디오의 최상위 해상도 계층을 기술하기 위해 이용된다. 논-스케일러블(non-scalable) 비디오에서는, 제2 적응 세트(602)만이 기저 계층의 종속성, 즉 dependencyId 속성 없이 존재할 것이다. 이런 제2 적응 세트(602)에서는, 단일 표현(603), 즉 표시 가능한 비디오에 대응하는 표현이 기술된다. 표현은 클라이언트 요구들에 대한 각각의 URL들을 가진 세그먼트들(610)의 리스트로서 기술된다.

[0030] 따라서, 표현은 'R1'(dependencyId 속성)에 의해 식별된 다른 표현에 의존하고, 실제로 제1 적응 세트(601)로부터의 기저 계층 표현에 의존한다. 종속성은 스트리밍 클라이언트가 향상 계층에 대한 현재 세그먼트를 얻기 전에 기저 계층에 대한 현재 세그먼트를 먼저 요청하도록 강요한다. 이는 이런 방식으로 참고될 트랙들이 자동으로 클라이언트에 의해 로딩될 것이기 때문에 타일 트랙들에 대한 종속성들을 나타내는데 사용될 수 없다. 이는 미디어 표현 동안 언제든지 사용자에 대한 관심 있는 타일들을 선택하는 것이 사용자에게 달렸기 때문에, 회피되어야 할 것이다. 따라서, 복합 트랙과 타일 트랙 사이에 종속성들을 나타내기 위해 SubRepresentation 요소가 이용된다. 표시 가능한 비디오는 서브 표현들(604 내지 608)의 리스트로서 기술된다. 각각의 서브 표현은

실제로 캡슐화된 MP4 파일에서 트랙을 표현한다. 따라서, 타일당(본 예에서는 4개의 타일) 하나의 서브 표현에 복합 트랙(608)을 위한 하나의 서브 표현이 더해진다. 각각의 서브 표현은 이것이 타일 트랙(614, 615, 616 및 617) 또는 복합 트랙(618)에 대응하는지 나타내기 위해 콘텐츠 컴포넌트 요소(614 내지 618)에 의해 기술된다. DASH/MPD에서 이용 가능한 Role 기술자 유형은 타일링을 위한 특정 방식과 함께 사용된다. Role 기술자는 또한 풀-프레임 비디오에서 타일의 위치를 나타낸다. 예를 들어, 컴포넌트(614)는 비디오의 좌측 상부에 위치한 타일을 기술한다(행의 첫 번째와 열의 첫 번째를 위한 1:1). 타일들의 치수, 폭 및 높이는 MPD에 의해 가능하게 만들어진 바와 같은 서브 표현의 속성들로서 특정된다. 대역폭 정보는 또한 그 대역폭에 따라, 타일들의 수의 결정과 타일들의 선택에서 DASH 클라이언트에게 도움이 되기 위해 여기에 둘 수 있다. 복합 트랙과 관련하여, 이는 디코딩될 수 있는 비디오 스트림을 다운로드의 끝에서 구축할 수 있는 것이 필요하기 때문에, 타일 트랙들과는 상이한 방식으로 시그널링되어야 한다. 이런 목적을 위해, 2개의 요소가 기술에 추가된다. 첫째로, 관련된 콘텐츠 컴포넌트(618)의 기술자는 이것이 모든 컴포넌트들 중 주 컴포넌트인 것을 나타낸다. 둘째로, 서브 표현에서, '요구되는' 새로운 속성은 대응하는 데이터가 요청되어야 하는 클라이언트에게 나타내기 위해 추가된다. 복합 트랙 또는 타일 트랙들 중 하나 이상에 대한 모든 요청들은 세그먼트 리스트(610)에 제공된 URL으로부터 계산된다(시간 간격당 하나). 본 예에서, MPD의 시작에서 "BaseURL"과 결합된 "URL_X"는 클라이언트가 HTTP GET 요청을 수행하는데 사용할 수 있는 완전한 URL을 제공한다. 이 요청으로, 클라이언트는 복합 트랙을 위한 데이터와 모든 타일 트랙들을 위한 모든 데이터를 얻을 것이다. 전송을 최적화하기 위해, 요청 대신에 클라이언트는 먼저 index_range 속성(620)으로부터 이용 가능한 데이터를 이용하여, 세그먼트 인덱스 정보(전형적으로 통상의 기술자에게 잘 알려진, ISO BMFF에서의 "ssix" 및/또는 "sidx" 정보)를 요청할 수 있다. 이 인덱스 정보는 각각의 컴포넌트에 대한 바이트 범위들을 결정할 수 있게 한다. DASH 클라이언트는 이후 선택된 트랙들(요구된 복합 트랙을 포함함)만큼, 적당한 바이트 범위를 갖는 많은 수의 HTTP GET 요청들을 전송할 수 있다.

[0031] 스트리밍 세션을 시작할 때, DASH 클라이언트는 매니페스트 파일을 요청한다. 수신될 때, 클라이언트는 매니페스트 파일을 분석하고, 그 환경에 적합한 AdaptationSets의 세트를 선택한다. 다음에, 클라이언트는 AdaptationSet 내의 MPD에서 그 대역폭, 디코딩 및 렌더링 능력들과 호환 가능한 Representation을 선택한다. 다음에, 이는 미디어 디코더들을 위한 초기화 정보로 시작하는, 요청될 세그먼트들의 리스트를 미리 구축한다. 초기화 정보가 디코더들에 의해 수신될 때, 이들은 초기화되고, 클라이언트는 제1 미디어 데이터를 요청하고, 실제로 표시를 시작하기 전에 최소 데이터량을 버퍼링한다.

[0032] 이들 다수의 요청/응답은 스트리밍 세션의 기동시 지연을 도입할 수 있다. 서비스 제공자들이 비디오의 시청을 시작하지 않고 서비스를 떠나는 이들의 클라이언트를 보게 되는 위험이 있다. 이 때를, 클라이언트에 의해 수행되는, 제1 미디어 데이터 청크에 대한 초기 HTTP 요청과, 미디어 데이터 청크가 실제로 기동 지연 역할을 하기 시작할 때의 사이에 지정하는 것이 일반적이다. 이는 네트워크 라운드 트립 시간뿐만 아니라 미디어 세그먼트들의 사이즈에 의존한다.

[0033] 서버 푸시는 웹 리소스 로딩 시간을 줄이기 위한 유용한 특징이다. 이런 서버들은 도 1a 내지 1e를 참고하여 논의된다.

[0034] 도 1b에는, HTTP/2 교환들에서 필요한 모든 리소스들, 즉 리소스들 R1 내지 R4와 서브 리소스들 A 내지 I(도 1a에 도시된 바와 같이)에 대한 요청이 전송되어야 한다. 그러나 서버들에 의해 푸시 특징을 이용할 때, 도 1c에 예시된 바와 같이, 요청들의 수는 요소들 R1 내지 R4로 제한된다. 요소들 A 내지 I는 도 1a에 도시된 종속성에 기초하여 서버에 의해 클라이언트에게 "푸시되고", 그에 의해 연관된 요청들을 불필요하게 만든다.

[0035] 따라서, 도 1b 및 1c에 예시된 바와 같이, 서버들이 푸시 특징을 이용할 때, 리소스를 그 서브 리소스로 로딩하는데 필요한 HTTP 라운드 트립들(요청 + 응답)의 수는 감소된다. 이는 특히, 예를 들어 이동 통신 네트워크들과 같은 높은 대기 시간 네트워크들에 대해 관심을 둔다.

[0036] HTTP는 웹 리소스, 전형적으로는 웹 페이지들을 전송하는데 사용되는 프로토콜이다. HTTP는 클라이언트와 서버를 암시한다:

[0037] • 클라이언트는 요청을 서버로 전송한다;

[0038] • 서버는 웹 리소스의 표현을 포함하는 응답으로 클라이언트의 요청에 응답한다.

[0039] 요청들과 응답들은 다양한 부분들, 특히 HTTP 헤더들을 포함하는 메시지들이다. HTTP 헤더는 값과 함께 명칭을

포함한다. 예를 들어, "Host: en.wikipedia.org"은 "Host" 헤더이고, 그 값은 "en.wikipedia.org"이다. 이는 질의받은 리소스의 호스트를 나타내기 위해 사용된다(예를 들어, HTTP를 기술하는 Wikipedia 페이지는 http://en.wikipedia.org/wiki/HTTP에서 이용 가능하다). HTTP 헤더들은 클라이언트 요청들과 서버 응답들에 나타난다.

[0040] HTTP/2는 스트림들을 통해 요청/응답들을 교환하는 것을 가능하게 한다. 스트림은 모든 HTTP 요청과 응답을 위해 HTTP/2 연결 내에서 생성된다. 프레임들은 요청들과 응답들의 콘텐츠와 헤더들을 전달하기 위해 스트림 내에서 교환된다.

[0041] HTTP/2는, 예를 들어 다음과 같은 상이한 의미들로 프레임들의 제한적 세트를 정의한다:

[0042] - HEADERS: HTTP 헤더들의 전송을 위해 제공되는 것

[0043] - DATA: HTTP 메시지 콘텐츠의 전송을 위해 제공되는 것

[0044] - PUSH_PROMISE: 푸시된 콘텐츠를 알리기 위해 제공되는 것

[0045] - PRIORITY: 스트림의 우선순위를 설정하기 위해 제공되는 것

[0046] - WINDOW_UPDATE: 제어 흐름 윈도의 값을 업데이트하기 위해 제공되는 것

[0047] - SETTINGS: 구성 파라미터들을 전달하기 위해 제공되는 것

[0048] - CONTINUATION: 헤더 블록 단편들의 시퀀스를 계속하기 위해 제공되는 것

[0049] - RST_STREAM: 스트림을 종료 또는 취소시키기 위해 제공되는 것.

[0050] 서버들에 의한 푸시는 서버들이 비요청 웹 리소스 표현들을 클라이언트들로 전송하는 것을 허용하기 위해 HTTP/2에 도입되었다. 웹 페이지들과 같은 웹 리소스는 일반적으로 다른 리소스과의 링크들을 포함하고, 다른 리소스들은 그들 자신이 다른 리소스과의 링크들을 포함할 수 있다. 웹 페이지를 완전히 표시하기 위해, 모든 링크된 리소스 및 서브링크된 리소스는 일반적으로 클라이언트에 의해 검색될 필요가 있다. 이런 발견의 증가는, 특히 이동 통신 네트워크들과 같은 높은 대기 시간 네트워크들에서 웹 페이지의 느린 표시를 야기할 수 있다.

[0051] 주어진 웹 페이지에 대한 요청을 수신할 때, 서버는 어느 다른 리소스가 요청된 리소스의 완전한 처리를 위해 필요한지 알 수 있다. 요청된 리소스과 링크된 리소스들을 동시에 전송함으로써, 서버는 웹 페이지의 로드 시간을 줄일 수 있다. 따라서, 푸시 특징을 이용하여, 서버는 소정 리소스가 요청되는 시간에 추가 리소스 표현들을 전송할 수 있다.

[0052] 도 1e의 흐름도를 참조하여, 푸시 특징을 구현하는 서버의 동작의 예시적 모드가 설명된다.

[0053] 단계 100 동안, 서버는 초기 요청을 수신한다. 다음에, 서버는 단계 101 동안 응답의 일부로서 푸시하기 위한 리소스들을 식별하고, 단계 102 동안 콘텐츠 응답을 전송하기 시작한다. 동시에, 서버는 단계 103 동안 푸시 약속 메시지들을 클라이언트로 전송한다. 이들 메시지는, 예를 들어 도 1a에 도시된 종속성들에 기초하여 서버가 푸시하고자 계획하고 있는 다른 리소스들을 식별한다. 이들 메시지들은 클라이언트가 미리, 어느 푸시된 리소스가 전송될지 알도록 하기 위해 전송된다. 특히, 이는 클라이언트가 동시에 푸시되거나 막 푸시되려고 하는 리소스에 대한 요청을 전송하는 위험이 감소된다. 이런 위험을 더 감소시키기 위해, 서버는 푸시 약속에 기술된 리소스를 참고하여 응답의 임의의 부분을 전송하기 전에 푸시 약속 메시지를 전송해야 한다. 이는 또한, 클라이언트들이 이들 리소스를 원하지 않는 경우 클라이언트들이 약속된 리소스들의 푸시의 취소를 요청할 수 있게 한다. 다음에, 서버는 단계 104 동안 응답과 모든 약속된 리소스를 전송한다. 단계 105 동안 프로세스는 종료된다.

[0054] 도 1d의 흐름도는 클라이언트 측에 대한 프로세스를 예시한다.

[0055] 클라이언트가 서버로부터 검색하기 위한 리소스를 식별했을 때, 클라이언트는 먼저 단계 106 동안 대응하는 데이터가 그 캐시 메모리에 이미 존재하는지 체크한다. 리소스가 캐시 메모리에 이미 존재하는 경우(예), 리소스는 단계 107 동안 캐시 메모리에서 검색된다. 캐시된 데이터는 이전 요청들로부터 검색된 데이터 또는 서버에 의해 이전에 푸시된 데이터 중 어느 하나일 수 있다. 캐시 메모리에 존재하지 않는 경우(아니오), 클라이언트는 단계 108 동안 요청을 전송하고, 서버의 응답을 대기한다. 서버로부터의 프레임의 수신시, 클라이언트는 단계 109 동안 프레임이 푸시 약속에 대응하는지 체크한다. 단계 110 동안 데이터 프레임이 푸시 약속에 대응하

는 경우(예), 클라이언트는 푸시 약속을 처리한다. 클라이언트는 푸시될 리소스를 식별한다. 클라이언트가 리소스를 수신하고 싶지 않다면, 클라이언트는 서버가 해당 리소스를 푸시하지 않도록 에러 메시지를 서버로 전송할 수 있다. 그렇지 않으면, 클라이언트는 대응하는 푸시 콘텐츠를 수신할 때까지 푸시 약속을 저장한다. 푸시 약속은 서버가 약속된 리소스를 푸시하는 동안 클라이언트가 이를 요청하지 않도록 이용된다. 데이터 프레임이 푸시 약속에 대응하지 않는 경우(아니오), 이는 단계 111 동안 프레임이 데이터를 푸시하는데 관련된 데이터 프레임인지 체크된다. 이것이 데이터를 푸시하는데 관련된 경우(예), 클라이언트는 단계 112 동안 푸시된 데이터를 처리한다. 푸시된 데이터는 클라이언트 캐시 내에 저장된다. 프레임은 데이터를 푸시하는데 관련된 데이터 프레임이 아닌 경우(아니오), 단계 113 동안 이것이 서버로부터 수신되는 응답에 대응하는지 체크된다. 프레임이 서버로부터의 응답에 대응하는 경우(예), 응답은 단계 114 동안 처리된다(예를 들어, 애플리케이션에 전송된다). 그렇지 않으면(아니오), 단계 115 동안 프레임이 응답(예)의 끝을 식별하는지 체크된다. 이 경우, 프로세스는 단계 116 동안 종료된다. 그렇지 않으면, 프로세스는 단계 109로 다시 돌아간다.

[0056] 따라서, 클라이언트가 응답 및 약속된 리소스들을 수신하는 것처럼 보인다. 이에 따라 약속된 리소스는 응답이 예를 들어, 검색된 웹 페이지를 표시하는 브라우저와 같은 애플리케이션에 의해 이용되는 동안 일반적으로 클라이언트 캐시에 저장된다. 클라이언트 애플리케이션이 푸시된 리소스들 중 하나를 요청할 때, 리소스는 어떠한 네트워크 지연을 초래하는 일 없이, 클라이언트 캐시에서 바로 검색된다.

[0057] 캐시에 푸시된 리소스의 저장은 캐시 제어 지령(directive)들을 이용하여 제어된다. 캐시 제어 지령들은 또한 응답들의 제어를 위해 사용된다. 이들 지령은 특히, 프록시들(임의의 푸시되거나 푸시되지 않은 리소스)에 적용 가능하고, 프록시 또는 클라이언트에 의해서만 저장될 수 있다.

[0058] 도 1a는 서버가 소유한 리소스들의 세트와 이들의 관계의 그래프이다. 리소스들의 세트는 서로 얽혀있다: R_1, R_2, R_3 및 R_4는 클라이언트에 의해 적절히 처리되도록 함께 다운로드될 필요가 있는 리소스이다. 게다가 서브 리소스들 A 내지 H가 정의된다. 이들 서브 리소스들은 1, 2 또는 3 리소스와 관련된다. 예를 들어, A는 R_1에 링크되고 C는 R_1, R_2 및 R_4에 링크된다.

[0059] 이미 상술한 바와 같이, 도 1b는 서버 PUSH 특징을 이용하지 않는 HTTP 교환을 도시한다: 클라이언트는 R_1을 요청하고, 다음에 R_2, A, B, C 및 D를 발견하고, 이들을 요청한다. 이들을 수신한 후, 클라이언트는 R_3, R_4, F 및 G를 요청한다. 결국 클라이언트는 H 및 I 서브 리소스들을 요청한다. 이는 리소스들의 전체 세트를 검색하기 위해 4번의 라운드 트립을 요청한다.

[0060] 이미 상술한 바와 같이, 도 1c는 서버에 의한 직접 연결된 서브 리소스들을 푸시하는 특징을 이용하는 HTTP 교환을 예시한다. R1를 요청한 후, 서버는 R_1을 전송하고, A, B, C 및 D를 푸시한다. 클라이언트는 R_2를 식별하고, 이를 요청한다. 서버는 R_2를 전송하고, F 및 G를 푸시한다. 결국 클라이언트는 R_3, R_4를 식별하고, 이들 리소스를 요청한다. 서버는 R_3, R_4를 전송하고, H 및 I를 푸시한다. 이 프로세스는 리소스들의 전체 세트를 검색하기 위해 3번의 라운드 트립을 요청한다.

[0061] 한 세트의 리소스들, 전형적으로 웹 페이지와 그 서브 리소스들의 로딩 시간을 줄이기 위해, HTTP/2는 다수의 요청과 응답 우선순위들을 동시에 교환할 수 있다. 도 2에 예시된 바와 같이, 웹 페이지는 자바스크립트, 이미지 등과 같은 여러 리소스의 다운로드를 요청할 수 있다. 초기 HTTP 교환(200) 동안, 클라이언트는 HTML 파일을 검색한다. 이 HTML 파일은 2개의 자바스크립트 파일(JS1, JS2), 2개의 이미지(IMG1, IMG2), 하나의 CSS 파일, 및 하나의 HTML 파일과의 링크들을 포함한다. 교환(201) 동안, 클라이언트는 각각의 파일에 대한 요청을 전송한다. 도 2의 교환(201)에 주어진 순서는 웹 페이지 순서에 기초한다: 클라이언트는 링크가 발견되자마자 요청을 전송한다. 서버는 이후 JS1, CSS, IMG1, HTML, IMG2 및 JS2에 대한 요청들을 수신하고, 그 순서를 따라 이들 요청을 처리한다. 클라이언트는 이후 그 순서에서 이들 리소스를 검색한다.

[0062] HTTP 우선순위들은 클라이언트가, 어느 요청들이 더 중요하고, 다른 요청들보다 더 빨리 처리되어야 하는지를 명시하는 것을 가능하게 한다. 우선순위들의 특정한 사용은 교환(202)에서 예시된다. 자바스크립트 파일들은 가장 높은 우선순위가 할당된다. CSS와 HTML 파일들은 중간 우선순위가 할당되고, 이미지들은 낮은 우선순위가 할당된다. 이 접근법은 블로킹 파일들, 또는 다른 파일들보다 더 빨리 다른 리소스에 대한 참조를 포함할 수 있는 파일들의 수신을 허용한다. 응답시, 서버는 교환(202)에서 설명된 바와 같이, 자바스크립트 파일들을 더 빨리, 이후에 CSS 및 HTML 파일들을 그 후에, 그리고 이미지들을 마지막에 전송할려고 시도할 것으로 예상된다.

서버들은 클라이언트 우선순위들을 따르는 것이 요구되지 않는다.

[0063] 우선순위들뿐만 아니라, HTTP/2는 동시에 교환될 데이터의 양이 제어될 수 있다고 규정한다. 클라이언트와 서버는 이들이 연결당 기준 및 스트링당 기준으로 버퍼링할 수 있는 데이터의 양이 어느 정도인지 특정할 수 있다. 이는 TCP 혼잡 제어와 유사하다: 이용 가능한 버퍼 사이즈를 특정하는 윈도 사이즈는 주어진 값으로 초기화되고; 이미터가 데이터를 전송할 때, 윈도 사이즈는 감소되고; 이미터는 윈도 사이즈가 결코 0 아래로 가지 않도록 데이터의 전송을 중단해야만 한다. 수신기는 데이터를 수신하고, 데이터가 버퍼로부터 수신되고 제거되었다는 것을 확인 응답하기 위해 데이터를 전송하고; 메시지는 버퍼로부터 제거된 데이터의 양을 포함하고; 윈도 사이즈는 이후 주어진 값으로부터 증가되고, 이미터는 데이터의 전송을 재시작할 수 있다.

[0064] 상기의 관점에서, DASH는 클라이언트가 일반적으로 이것이 수행하고 있는 애플리케이션의 목적을 위해 콘텐츠의 최상의 표현을 선택할 수 있기 때문에, 클라이언트가 스트리밍을 리드(lead)한다는 가정에 기초하는 것으로 보인다. 예를 들어, 클라이언트는 그 폼팩터와 화면 해상도에 기초하여 고화질 또는 저화질 콘텐츠를 요청하는지 알 수 있다.

[0065] 서버 기반 스트리밍은 전형적으로 RTP를 이용하여 행해진다. DASH와 대조적으로, RTP는 HTTP를 사용하지 않고, 웹 인프라, 특히 프록시들 및 캐시들로부터 직접 이익을 얻을 수 없다. 웹 소켓 기반 미디어 스트리밍은 동일한 단점들을 갖는다. HTTP/1.1에서, 서버 기반 스트리밍은 서버가 일반적으로 클라이언트 요청들에만 응답할 수 있기 때문에 쉽게 구현될 수 없다. HTTP/2에서, 특히 푸시 특징의 도입으로, DASH 기반 서버들은 스트리밍을 리드할 수 있다. 따라서, 서버들은 이들이 사용자 경험을 최적화하기 위해 스트리밍하고 있는 콘텐츠의 특성들의 지식을 이용할 수 있다. 예를 들어, 서버는 SD로서 영화를 푸시할 수 있지만(제한된 대역폭 때문에), 광고들이 추가적인 제한된 양의 대역폭을 취하기 때문에 광고들을 HD로서 푸시할 수 있다. 또 다른 예는, 저-해상도 비디오로 고속 시작을 하기 시작하고 대역폭이 완전히 추정될 때 가능한 최고 표현으로 스위칭하는 서버의 경우이다.

[0066] 서버가 스트리밍을 리드할 수 있게 하기 위해, 하나의 접근법은 서버가 선호하는 데이터(특히, DASH 데이터)를 푸시하게 하는 것이다. 클라이언트는 이후 비디오를 표시하는데 이용 가능한 데이터는 무엇이든지 이용한다. 서버는 전형적으로 여러 세그먼트의 푸시를 동시에 알린다. 서버는 이후 세그먼트들을 동시에 또는 연속해서 전송한다.

[0067] 문제점은, 약속된 데이터가 원하는 시간에서 전송되고 수신되는지를 클라이언트 및 서버가 알 수 없고; 클라이언트가 비디오 세그먼트가 전송될 때와 그 순서를 알 수 없는데서 일어난다.

[0068] 또한, 서버에 의해 푸시되거나 알려진 약속된 데이터는 클라이언트 요구들과 매칭하지 않을 수 있고, 그에 따라 특히 서버 측에서 리소스 소모를 초래한다.

[0069] 따라서, 특히 DASH 기반 통신들의 상황에서 데이터 스트리밍을 향상시킬 필요가 있다.

발명의 내용

[0070] 본 발명은 이러한 상황 안에 있다.

[0071] 서버의 관점에 대응하는 본 발명의 제1 양태에 따르면, 서버 디바이스에 의해 클라이언트 디바이스에 미디어 데이터를 스트리밍하는 방법은:

[0072] - 상기 클라이언트 디바이스로부터, 제1 미디어 데이터에 관한 요청을 수신하는 단계,

[0073] - 요청받지 않고 상기 클라이언트 디바이스에 전송될 제2 미디어 데이터를 식별하는 단계, 및

[0074] - 상기 요청에 응답하여, 상기 제1 미디어 데이터에 관한 데이터를, 상기 클라이언트 디바이스에 전송하고, 상기 제2 미디어 데이터를 각각 식별하는 적어도 하나의 알림 메시지(announcement message)를 상기 클라이언트 디바이스에 전송할 목적으로 상기 알림 메시지 또는 메시지들을 준비하는 단계를 포함하고,

[0075] 상기 방법은 상기 서버 디바이스가 상기 제2 비요청 미디어 데이터의 식별 또는 상기 클라이언트 디바이스로의 전송을 추진하기 위해 상기 클라이언트 디바이스와 공유되는 푸시 정책을 이용하는 단계를 더 포함한다.

[0076] 클라이언트의 관점에 대응하는 본 발명의 제2 양태에 따르면, 클라이언트 디바이스에 의해, 서버 디바이스에 의해 스트리밍되는 미디어 데이터에 액세스하는 방법은:

[0077] - 제1 미디어 데이터에 관한 요청을 상기 서버 디바이스에 전송하는 단계,

[0078] - 상기 요청에 응답하여, 상기 제1 미디어 데이터에 관한 데이터를 상기 서버 디바이스로부터 수신하는 단계를 포함하고,

[0079] 상기 방법은 상기 클라이언트 디바이스가 상기 클라이언트 디바이스에 의해 요청받지 않고 상기 서버에 의해 전송될 제2 미디어 데이터를 결정하거나 상기 서버 디바이스에 의한 그것의 전송 순서를 결정하기 위해 상기 서버 디바이스와 공유되는 푸시 정책을 이용하는 단계를 더 포함한다.

[0080] 특히, 상기 공유 푸시 정책은, 상기 디바이스들이 상기 서버 디바이스에 의해 상기 클라이언트 디바이스에 전송될 상기 제2 비요청 미디어 데이터를 결정하기 위해, 제2 미디어 데이터를 어떻게 결정하는지를 정의할 수 있다.

[0081] 이러한 접근법 덕분에, 푸시될 미디어 데이터에 관한 서버의 결정과 클라이언트의 필요 사이의 불일치가 감소될 수 있고 따라서 리소스들이 절약될 수 있다.

[0082] 이것은 클라이언트가 서버의 거동과, 따라서 막 푸시되려고 하는 제2 미디어 데이터를 예상하는 것을 가능하게 하는 공유 푸시 정책을 이용하는 것에 의해 달성된다. 몇몇 클라이언트의 후속 요청들을 위해 이용될 수 있는 상기 공유 푸시 정책 덕분에, 클라이언트는 상기 요청들이 서버에 보내지기 전에도 서버의 거동을 예상할 수 있다.

[0083] 상기 예상의 결과로, 클라이언트는, 서버에 의한 알림에 관하여 예상되는 방식으로, 필요하지 않은 그러한 제2 미디어 데이터의 취소를 준비하고 요청할 수 있다.

[0084] 제1 미디어 데이터에 관한 상기 요청은 제1 미디어 데이터 및/또는 이 제1 미디어 데이터와 관련된 다른 데이터에 관한 것일 수 있다.

[0085] 상기 제2 미디어 데이터는, 예를 들어 상기 서버 디바이스에 의해, 상기 제1 미디어 데이터와 연관될 수 있다.

[0086] 본 발명의 실시예들은 서버-유도 스트리밍(server-guided streaming)을 위한 경량의 메커니즘을 제공한다. 실시예들은 DASH 네트워크들의 상황에서 구현될 수 있다.

[0087] 서버 디바이스들은 클라이언트 디바이스들에 대해 콘텐츠 추천들을 할 수 있다. 또한, 그것들은 네트워크 사용량을 최적화할 수 있다.

[0088] 본 발명의 실시예들은 기존의 HTTP/2 특징들과 호환된다. 이러한 특징들은 유리하게도 본 발명의 실시예들을 구현하기 위해 이용될 수 있다.

[0089] 네트워크 성능들이 일반적으로 증가된다.

[0090] 대응하여, 본 발명은 또한 클라이언트 디바이스에 미디어 데이터를 스트리밍하기 위한 서버 디바이스에 관한 것이고, 그 디바이스는:

[0091] - 클라이언트 디바이스로부터, 제1 미디어 데이터에 관한 요청을 수신하도록 구성된 수신기,

[0092] - 요청받지 않고 상기 클라이언트 디바이스에 전송될 제2 미디어 데이터를 식별하도록 구성된 제어 유닛, 및

[0093] - 상기 요청에 응답하여, 상기 제1 미디어 데이터에 관한 데이터를, 상기 클라이언트 디바이스에 전송하고, 상기 제2 미디어 데이터를 각각 식별하는 적어도 하나의 알림 메시지를 상기 클라이언트 디바이스에 전송할 목적으로 상기 알림 메시지 또는 메시지들을 준비하도록 구성된 전송기를 포함하고,

[0094] 상기 제어 유닛은 상기 제2 비요청 미디어 데이터의 식별 또는 상기 클라이언트 디바이스로의 전송을 추진하기 위해 상기 클라이언트 디바이스와 공유되는 푸시 정책을 이용하도록 더 구성된다.

[0095] 본 발명은 또한 서버 디바이스에 의해 스트리밍되는 미디어 데이터에 액세스하기 위한 클라이언트 디바이스와 관련되고, 이 디바이스는:

[0096] - 제1 미디어 데이터에 관한 요청을 상기 서버 디바이스에 전송하도록 구성된 전송기, 및

[0097] - 상기 요청에 응답하여, 상기 제1 미디어 데이터에 관한 데이터를 상기 서버 디바이스로부터 수신하도록 구성된 수신기를 포함하고,

[0098] 상기 클라이언트 디바이스는

[0099] 상기 클라이언트 디바이스가 상기 클라이언트 디바이스에 의해 요청받지 않고 상기 서버에 의해 전송될 제2 미디어 데이터를 결정하거나 상기 서버 디바이스에 의한 그것의 전송 순서를 결정하기 위해 상기 서버 디바이스와 공유되는 푸시 정책을 이용하도록 구성된다.

[0100] 서버와 클라이언트 디바이스들은 전술한 대응하는 방법들과 동일한 이점들을 가진다.

[0101] 상기 방법들과 디바이스들의 옵션의 특징들이 종속 청구항들에서 정의된다. 이들 중 일부가 상기 방법들에 관하여 후술된다. 그러나 그것들은 대응하는 디바이스에도 적용될 수 있다.

[0102] 아래에서 명시적 접근법이라고 불리는 일부 실시예들에서, 서버의 관점으로부터의 방법은:

[0103] 상기 서버 디바이스에 의해 푸시 정책을 결정하는 단계, 및

[0104] 상기 결정된 푸시 정책을 기술하는 푸시 정책 정보를 상기 클라이언트 디바이스와 공유하기 위해 상기 푸시 정책 정보를 상기 서버 디바이스로부터 상기 클라이언트 디바이스로 전송하는 단계를 더 포함한다.

[0105] 대응하여, 클라이언트 측에서, 상기 방법은 상기 공유 푸시 정책을 기술하는 푸시 정책 정보를 상기 서버 디바이스로부터 수신하는 단계를 더 포함한다.

[0106] 아래의 일부 예들에 기술된 바와 같이, 상기 공유 푸시 정책을 기술하는 상기 푸시 정책 정보는 상기 서버 디바이스로부터 상기 클라이언트 디바이스로 전송되는 기술 파일(description file)에 삽입되고, 상기 기술 파일은 상기 제1 미디어 데이터를 포함하는 미디어에 관한 기술 정보를 포함하고, 상기 방법은 상기 공유 푸시 정책을 이용하여 상기 기술 파일에 기초하여 상기 제2 비요청 미디어 데이터를 결정하는 단계를 더 포함한다.

[0107] 특정한 실시예에서, 상기 기술 파일은 복수의 미디어 데이터 속성 레벨을 이용하여 상기 미디어 데이터를 기술하고, 상기 기술 파일의 다양한 각각의 레벨들에서 다양한 공유 푸시 정책들이 정의된다.

[0108] 다른 예들에서, 상기 공유 푸시 정책을 기술하는 상기 푸시 정책 정보는 상기 서버 디바이스로부터 상기 클라이언트 디바이스로 전송되는 HTTP 프레임의 헤더에 삽입된다.

[0109] 특정한 특징들에 따르면, 상기 방법은, 상기 서버 디바이스에서, 상기 클라이언트 디바이스로부터 HTTP 프레임의 헤더에 삽입된 푸시 정책 업데이트 정보를 수신하는 단계, 및 그에 따라 상기 클라이언트 디바이스에 의해 요청된 다른 미디어 데이터로부터 비요청 미디어 데이터를 결정하기 전에 상기 공유 푸시 정책을 업데이트하는 단계를 더 포함할 수 있다.

[0110] 대응하여, 상기 방법은, 상기 클라이언트 디바이스에서, HTTP 프레임의 헤더에 삽입된 푸시 정책 업데이트 정보를 상기 서버 디바이스에 전송하는 단계를 더 포함할 수 있다.

[0111] 하이브리드 접근법에 따르면, 상기 공유 푸시 정책을 기술하는 상기 푸시 정책 정보는 제1 푸시 정책 부분과 제2 푸시 정책 부분에 의해 정의되고, 상기 제1 푸시 정책 부분은 상기 서버 디바이스로부터 상기 클라이언트 디바이스로 전송되는 기술 파일에 삽입되고, 상기 기술 파일은 상기 제1 미디어 데이터를 포함하는 미디어 데이터에 관한 기술 정보를 포함하고, 상기 방법은 상기 공유 푸시 정책을 이용하여 상기 기술 파일에 기초하여 상기 제2 비요청 미디어 데이터를 결정하는 단계를 더 포함하고,

[0112] 상기 제2 푸시 정책 부분은 상기 서버 디바이스로부터 상기 클라이언트 디바이스로 전송되는 HTTP 프레임의 헤더에 삽입된다.

[0113] 예를 들어, 상기 제2 푸시 정책 부분은 상기 제1 푸시 정책 부분에 정의된 하나 이상의 연관된 변수에 대한 하나 이상의 값을 포함할 수 있다.

[0114] 또한, 상기 기술 파일은 복수의 후보 푸시 정책의 기술을 포함할 수 있고, 상기 제2 푸시 정책 부분은 따라서 상기 복수로부터 후보 푸시 정책의 식별자를 포함할 수 있고, 그 식별된 후보 푸시 정책은 따라서 상기 제1 푸시 정책 부분을 형성한다.

[0115] 다른 실시예들에서, 상기 푸시 정책 정보는 상기 서버 디바이스로부터 상기 클라이언트 디바이스로 전송되는 웹 페이지에 삽입된 자바스크립트(JavaScript) 프로그램을 포함한다.

[0116] 또 다른 실시예들에서, 상기 방법은 (전술한 기술 파일 또는 아래 예들에 소개된 HTML 페이지와 같은) 구조화된 문서에 기초하여 상기 제2 비요청 미디어 데이터를 결정하는 단계를 더 포함하고, 상기 구조화된 문서는 상기 제1 미디어 데이터를 포함하는 미디어 데이터에 관한 기술 정보를 포함하고,

[0117] 상기 푸시 정책 정보는 상기 제2 비요청 미디어 데이터를 식별하기 위해 상기 구조화된 문서의 트리 표현에 대해 평가될 XPath 표현식을 포함한다.

[0118] 상기 푸시 정책 정보의 구문에 관하여, 실시예들은 상기 푸시 정책 정보는 기술 파일에서 식별될 제2 비요청 미디어 데이터의 양을 정의하는 제1 푸시 속성을 포함하고,

[0119] 상기 기술 파일은 상기 제1 미디어 데이터를 포함하는 미디어 데이터에 관한 기술 정보를 포함하고, 상기 방법은 상기 공유 푸시 정책을 이용하여 상기 기술 파일에 기초하여 상기 제2 비요청 미디어 데이터를 결정하는 단계를 더 포함하는 것을 규정한다.

[0120] 특정한 특징들에 따르면, 상기 제1 푸시 속성은 상기 기술 파일 내에서 요청된 상기 제1 미디어 데이터에 관련하여 상기 제2 비요청 미디어 데이터를 식별한다. 이것은 후술하는 연산자들을 이용하여 행해질 수 있다.

[0121] 변형예에서, 상기 제1 푸시 속성은 상기 기술 파일 내의 특정한 미디어 데이터의 식별자이다.

[0122] 특정한 특징들에 따르면, 상기 기술 파일 내의 상기 기술 정보는 미디어 데이터가 속하는 시간 기간을 정의하는 기간 속성, 상기 미디어 데이터의 미디어 유형을 정의하는 적응 속성, 상기 미디어 데이터의 인코딩 버전(예를 들어, 비트레이트, 프레임 레이트, 프레임 해상도, 타이밍 정보, 등등)을 정의하는 표현 속성 및 세그먼트 속성 정의 중에서 적어도 하나의 미디어 데이터 속성에 따라 미디어 데이터를 기술하고,

[0123] 상기 푸시 정책 정보는, 상기 제2 비요청 미디어 데이터를 식별하기 위해, 상기 미디어 데이터 속성 또는 속성들에 대한 제약 조건(constraint)을 정의하는 적어도 제2 푸시 속성을 포함한다.

[0124] 이것은 상기 기술 파일의 전역에서 매우 선택적인 푸시 정책들을 갖는 것을 가능하게 한다.

[0125] 특히, 상기 푸시 속성 또는 속성들은 상기 기술 파일 내의 상기 제1 미디어 데이터의 대응하는 미디어 데이터 속성 또는 속성들에 관련하여 상기 제2 비요청 미디어 데이터의 미디어 데이터 속성 또는 속성들을 정의할 수 있다.

[0126] 대안으로, 상기 푸시 속성 또는 속성들은 상기 제2 비요청 미디어 데이터가 검색되어야 하는 상기 기술 파일 내의 노드를 식별할 수 있다.

[0127] 일부 실시예들에서, 상기 기술 파일 내의 상기 기술 정보는, 각각의 미디어 데이터에 대해 하나씩, 상기 미디어 데이터와 연관된 우선순위 속성들을 포함하고, 상기 제2 미디어 데이터의 전송 순서가 상기 연관된 우선순위 속성들에 기초한다. 이것은 상기 푸시 데이터의 전송 순서를 정의하는 것이다.

[0128] 실시예들에서, 상기 공유 푸시 정책은 요청된 상기 제1 미디어 데이터로부터 상기 제2 미디어 데이터를 식별한다.

[0129] 아래에서 묵시적 접근법이라고 불리는 실시예들에서, 상기 공유 푸시 정책은 서버 디바이스와 클라이언트 디바이스 양쪽 모두에서 알고리즘을 이용하여 동일한 제2 미디어 데이터를 결정하는 것으로 구현되고, 상기 알고리즘은 상기 서버 디바이스와 상기 클라이언트 디바이스가 요청된 상기 제1 미디어 데이터로부터 동일한 제2 미디어 데이터를 결정하는 것을 가능하게 한다.

[0130] 상기 묵시적 접근법과 명시적 접근법 양쪽 모두에 적응된 일부 실시예들에서, 상기 식별된 제2 미디어 데이터가, 각각이 알림 메시지를 요구하는, 복수의 미디어 세그먼트를 포함한다면, 상기 방법은 대응하는 복수의 알림 메시지를 상기 클라이언트 디바이스에 전송될 단일 알림 메시지로 병합하는 단계를 더 포함할 수 있다. 이것은 더 적은 알림 메시지들이 전송될 것이므로 대역폭 소비를 줄이는 것이다.

[0131] 상기 공유 푸시 정책을 실제로 이용하고 그 결과에 따라 상기 클라이언트 디바이스에 의한 푸시들을 예상하기 위해, 상기 방법은 상기 서버 디바이스가 대응하는 준비된 알림 메시지를 전송하지 않도록 상기 제2 비요청 미디어 데이터의 일부의 전송을 취소할 것을 요청하는 취소 요청을 상기 클라이언트 디바이스로부터 수신하는 단계를 더 포함할 수 있다.

[0132] 대응하여 상기 클라이언트에서, 상기 방법은, 상기 서버 디바이스가 상기 제2 비요청 미디어 데이터의 일부를 식별하는 알림 메시지를 전송하지 않도록 만들기 위하여, 상기 제2 비요청 미디어 데이터의 상기 일부의 전송을 취소할 것을 요청하는 취소 요청을 상기 서버 디바이스에 전송하는 단계를 더 포함할 수 있다.

[0133] 본 발명의 실시예들에서, 상기 제2 비요청 미디어 데이터는 상기 서버 디바이스에 의해 준비된(그리고 어쩌면 상기 서버 디바이스로부터 수신된) 그리고 상기 서버 디바이스가 요청받지 않고 상기 클라이언트 디바이스에 전

송하고자 하는 상기 제2 비요청 미디어 데이터를 식별하는 적어도 하나의 알림 메시지와 독립적으로 상기 클라이언트 디바이스에 의해 결정된다. 여기서, "독립적으로(independently)"는 상기 클라이언트 디바이스가 상기 제2 비요청 데이터의 미래의 전송에 대해 상기 클라이언트 디바이스에 알리기 위해 전용되는 그러한 알림 메시지(즉, PUSH_PROMISE)를 인지하지 않고 그러한 비요청 데이터의 결정을 할 수 있다는 것을 의미한다.

[0134] 본 발명의 다른 실시예들에서, 각각의 제1 미디어 데이터에 관한 복수의 요청으로부터 각각의 비요청 미디어 데이터를 결정하기 위해 동일한 공유 푸시 정책이 이용된다. 시간 및 연속적인 요청들에 대해 동일한 푸시 정책을 이용함으로써, 클라이언트는 서버에 의한 무익한 데이터의 전송을 효율적으로 예측하는 한층 더 좋은 위치에 있고, 따라서 그것들의 전송 및 대응하는 알림 메시지들의 전송을 효율적으로 취소하는 위치에 있다.

[0135] 서버로부터 클라이언트로의 푸시 데이터의 전송 순서의 통지에 관하여, 서버 디바이스로부터 클라이언트 디바이스로 미디어 데이터를 스트리밍하는 방법은:

[0136] - 제1 미디어 데이터에 관한 요청을 클라이언트 디바이스로부터 수신하는 단계,

[0137] - 요청받지 않고 상기 클라이언트 디바이스에 전송될 제2 미디어 데이터를 식별하는 단계,

[0138] - 상기 요청에 응답하여, 상기 제1 미디어 데이터에 관한 데이터, 및 상기 제2 미디어 데이터를 각각 식별하는 적어도 하나의 알림 메시지를 상기 클라이언트 디바이스에 전송하는 단계를 포함할 수 있고,

[0139] 상기 방법은:

[0140] - 상기 서버 디바이스에 의한 상기 제2 미디어 데이터의 전송 순서를 정의하는 단계(이것이 공유 푸시 정책의 전부 또는 일부를 형성함),

[0141] - 상기 알림 메시지들과 함께 상기 전송 순서와 관련된 정보를 전송하는 단계 - 상기 정보는 상기 클라이언트 디바이스가 상기 서버에 의해 정의된 상기 전송 순서를 결정하는 것을 가능하게 함 - 를 더 포함한다.

[0142] 예를 들어, 상기 제2 미디어 데이터의 전송 순서는 상기 클라이언트 디바이스에 따라 우선순위 값들에 따라 정의되고, 최고 우선순위 값을 가진 미디어 데이터가 먼저 전송된다.

[0143] 상기 우선순위 값들은 HTTP/2 프로토콜에 따라 정의될 수 있다.

[0144] 실시예들에 따르면, 적어도 하나의 우선순위 값이 네트워크 대역폭 추정 메커니즘과 연관되고, 상기 방법은:

[0145] - 상기 메커니즘과 연관된 우선순위 값을 가진 제2 미디어 데이터를 상기 클라이언트 디바이스에 전송하는 단계,

[0146] - 상기 제2 미디어 데이터에 응답하여, 상기 클라이언트 디바이스로부터 적어도 하나의 제어 흐름 메시지를 수신하는 단계, 및

[0147] - 수신된 상기 적어도 하나의 제어 흐름 메시지에 기초하여 이용 가능한 대역폭을 추정하는 단계를 더 포함한다.

[0148] 예를 들어, 서버 디바이스는 각각의 그리고 상이한 사이즈들을 가진 복수의 데이터 프레임에 따라 상기 제2 미디어 데이터를 전송한다.

[0149] 상기 방법은 서버 디바이스에 의해, 상기 대역폭 추정에 기초하여, 상기 제2 미디어 데이터의 업데이트된 전송 순서를 정의하는 단계를 더 포함할 수 있다.

[0150] 실시예들에 따르면 클라이언트 디바이스로부터의 상기 요청은 상기 제1 미디어 데이터를 포함하는 미디어 데이터와 관련된 기술 파일을 수신하기 위한 요청을 포함하고, 상기 기술 파일은 상기 제1 미디어 데이터에 관한 기술 정보를 포함하고, 상기 방법은 상기 기술 파일에 기초하여 상기 제2 비요청 미디어 데이터를 결정하는 단계를 더 포함한다.

[0151] 예를 들어, 요청된 제1 미디어 데이터는 비디오 세그먼트들이다.

[0152] 상기 스트리밍은 DASH 표준에 따라 수행될 수 있다. 예를 들어, 상기 방법은:

[0153] - 클라이언트 디바이스로부터, 순서 업데이트 요청(ordering update request)을 수신하는 단계,

[0154] - 상기 순서 업데이트 요청에 기초하여, 상기 제2 미디어 데이터의 새로운 전송 순서를 정의하고 상기 제2 미디어 데이터의 상기 새로운 전송 순서와 관련된 상기 정보를 업데이트하는 단계, 및

[0155] - 상기 전송 순서와 관련된 상기 업데이트된 정보에 따라 상기 제2 미디어 데이터를 상기 클라이언트에 전송하는 단계를 더 포함한다.

[0156] 상기 방법은 순서 업데이트 확인 메시지를 상기 클라이언트 디바이스에 전송하는 단계를 더 포함할 수 있다.

[0157] 예를 들어, 상기 업데이트된 순서는 상기 순서 업데이트 요청을 수신했을 때 상기 클라이언트 디바이스로의 전송이 개시되지 않은 상기 제2 미디어 데이터에 대하여 정의된다.

[0158] 예를 들어, 상기 순서 업데이트 요청은 제2 미디어 데이터의 적어도 일부에 대한 순서 값을 포함한다.

[0159] 실시예들에 따르면, 상기 제2 미디어의 전송 순서는 우선순위 값들에 따라 정의되고, 제1 미디어 데이터의 적어도 일부에 대한 우선순위 값이 업데이트될 때, 요청받지 않고 상기 클라이언트 디바이스에 전송될 그리고 제1 미디어 데이터의 상기 적어도 일부와 연관된 제2 미디어 데이터의 적어도 일부에 대한 우선순위 값들이 그에 따라 업데이트된다.

[0160] 예를 들어, 상기 제1 및 제2 미디어는 시간적 관계, 공간적 관계 및 품질 관계 중 적어도 하나에 따라 연관된다.

[0161] 실시예들에 따르면:

[0162] - 상기 제2 미디어 데이터는 상기 제1 미디어 데이터의 품질을 향상시키기 위한 향상 데이터(enhancement data)를 포함하고,

[0163] - 향상 계층의 미디어 데이터에 대한 우선순위 값이 업데이트될 때, 상기 향상 계층의 모든 미디어 데이터에 대한 우선순위 값들이 업데이트된다.

[0164] 예를 들어, 상기 제1 및 제2 미디어 데이터는 비디오 시간 세그먼트들을 포함하고, 상기 향상 미디어 데이터의 시작 시간은 상기 제1 미디어 데이터의 비디오 콘텐츠와 관련된 정보에 기초한다.

[0165] 예를 들어, 상기 제1 미디어 데이터의 비디오 콘텐츠와 관련된 상기 정보는 상기 기술 파일에 저장된다.

[0166] 예를 들어, 상기 전송 순서는 제1 미디어 데이터와 제2 미디어 데이터 간의 디코딩 관계들에 적어도 기초한다.

[0167] 예를 들어, 상기 전송 순서는 상기 미디어 데이터의 통계적 인기들에 적어도 기초한다.

[0168] 예를 들어, 상기 전송 순서는 상기 클라이언트 디바이스 측에서의 상기 미디어 데이터의 재생 시간에 적어도 기초한다.

[0169] 예를 들어, 상기 전송 순서는 상기 미디어 데이터의 추정 전송 시간에 적어도 기초한다.

[0170] 예를 들어, 상기 전송 순서는 상기 미디어 데이터에 대한 사용자 정의 관심들(user-defined interests)에 적어도 기초한다.

[0171] 상기 방법은:

[0172] - 상기 클라이언트 디바이스로부터 제어 메시지들을 수신하는 단계 - 상기 제어 메시지들은 상기 서버 디바이스가 현재 재생되고 있는 미디어 데이터를 식별하는 것을 가능하게 함 -,

[0173] - 상기 서버에 의해, 상기 제어 메시지들에 기초하여, 상기 제2 미디어 데이터의 업데이트된 전송 순서를 정의하는 단계, 및

[0174] - 상기 업데이트된 전송 순서에 따라 상기 제2 미디어 데이터를 상기 클라이언트에 전송하는 단계를 더 포함할 수 있다.

[0175] 상기 방법은 순서 업데이트 확인 메시지를 상기 클라이언트 디바이스에 전송하는 단계를 더 포함할 수 있다.

[0176] 예를 들어, 상기 제어 메시지들은 상기 클라이언트 디바이스의 버퍼 메모리의 사용과 관련되고, 상기 버퍼 메모리는 상기 클라이언트에 의해 재생되도록 미디어 데이터를 저장한다.

[0177] 예를 들어, 서버 디바이스는 전송된 제1 요청된 미디어 데이터의 레코드를 유지하고, 상기 레코드와 상기 버퍼 메모리의 상기 사용에 기초하여 상기 제2 미디어 데이터의 식별이 수행된다.

[0178] 예를 들어, 상기 전송 순서 정보는 상기 알림 메시지들 내에서 전송된다.

[0179] 예를 들어, 상기 전송 순서 정보는 상기 알림 메시지들 뒤에 전용 메시지들 내에서 전송된다.

[0180] 클라이언트의 관점으로부터, 클라이언트 디바이스에 의해, 서버 디바이스에 의해 스트리밍된 미디어 데이터에 액세스하는 방법은:

[0181] - 제1 미디어 데이터에 관한 요청을 상기 서버 디바이스에 전송하는 단계,

[0182] - 상기 요청에 응답하여, 상기 제1 미디어 데이터에 관한 데이터, 및 요청받지 않고 상기 클라이언트 디바이스에 전송될 제2 미디어를 각각 식별하는 적어도 하나의 알림 메시지를 상기 서버 디바이스로부터 수신하는 단계를 포함할 수 있고,

[0183] 상기 방법은:

[0184] - 상기 알림 메시지들과 함께 상기 제2 미디어 데이터의 전송 순서와 관련된 정보를 수신하는 단계 - 상기 정보(즉, 공유 푸시 정책)는 상기 클라이언트 디바이스가 상기 서버에 의해 정의된 상기 제2 미디어 데이터의 전송 순서를 결정하는 것을 가능하게 함 - 를 더 포함한다.

[0185] 상기 방법은 상기 클라이언트 디바이스에 의해, 상기 서버 디바이스에 의해 정의된 상기 제2 미디어 데이터의 전송 순서가 상기 클라이언트 디바이스 측에서의 스트리밍 제약 조건들을 만족시키는지를 결정하는 단계, 및 상기 제약 조건들이 만족되지 않는다면, 순서 업데이트 요청을 상기 서버 디바이스에 전송하는 단계를 더 포함할 수 있다.

[0186] 예를 들어, 상기 제2 미디어 데이터의 상기 전송 순서는 상기 클라이언트 디바이스에 따라 우선순위 값들에 따라 정의되고, 최고 우선순위 값을 가진 미디어 데이터가 먼저 전송된다.

[0187] 예를 들어, 상기 우선순위 값들은 HTTP/2 프로토콜에 따라 정의된다.

[0188] 실시예들에 따르면, 적어도 하나의 우선순위 값이 네트워크 대역폭 추정 메커니즘과 연관되고, 상기 방법은:

[0189] - 상기 메커니즘과 연관된 우선순위 값을 가진 제2 미디어 데이터를 상기 서버 디바이스로부터 수신하는 단계,

[0190] - 상기 제2 미디어 데이터에 응답하여, 적어도 하나의 제어 흐름 메시지를 상기 서버 디바이스에 전송하여, 상기 서버 디바이스가 전송된 상기 적어도 하나의 제어 흐름 메시지에 기초하여 이용 가능한 대역폭을 추정할 수 있게 하는 단계를 더 포함한다.

[0191] 예를 들어, 클라이언트 디바이스는 각각의 그리고 상이한 사이즈들을 가진 복수의 데이터 프레임에 따라 상기 제2 미디어 데이터를 수신한다.

[0192] 예를 들어, 서버 디바이스에 의해, 상기 대역폭 추정에 기초하여, 상기 제2 미디어 데이터의 업데이트된 전송 순서가 정의된다.

[0193] 예를 들어, 클라이언트 디바이스로부터의 상기 요청은 상기 제1 미디어 데이터를 포함하는 미디어 데이터와 관련된 기술 파일을 수신하기 위한 요청을 포함하고, 상기 기술 파일은 상기 제1 미디어 데이터에 관한 기술 정보를 포함하고, 상기 방법은 상기 기술 파일에 기초하여 상기 제2 비요청 미디어 데이터를 결정하는 단계를 더 포함한다.

[0194] 예를 들어, 요청된 제1 미디어 데이터는 비디오 세그먼트들이다.

[0195] 예를 들어, 상기 스트리밍은 DASH 표준에 따라 수행될 수 있다.

[0196] 상기 방법은 상기 서버 디바이스에 의해 정의된 상기 제2 미디어 데이터의 새로운 전송 순서와 관련된 업데이트된 정보에 따라 상기 서버 디바이스로부터 상기 제2 미디어 데이터를 수신하는 단계를 더 포함할 수 있다.

[0197] 상기 방법은 상기 서버 디바이스로부터 순서 업데이트 확인 메시지를 수신하는 단계를 더 포함할 수 있다.

[0198] 실시예들에 따르면, 상기 업데이트된 순서는 상기 서버 디바이스가 상기 순서 업데이트 요청을 수신했을 때 상기 서버 디바이스로부터의 전송이 개시되지 않은 상기 제2 미디어 데이터에 대하여 정의된다.

[0199] 실시예들에 따르면, 상기 순서 업데이트 요청은 제2 미디어 데이터의 적어도 일부에 대한 순서 값을 포함한다.

[0200] 실시예들에 따르면, 상기 제2 미디어의 전송 순서는 우선순위 값들에 따라 정의되고, 제1 미디어 데이터의 적어도 일부에 대한 우선순위 값이 업데이트될 때, 요청받지 않고 상기 클라이언트 디바이스에 전송될 그리고 제1 미디어 데이터의 상기 적어도 일부와 연관된 제2 미디어 데이터의 적어도 일부에 대한 우선순위 값들이 그에 따

라 업데이트된다.

[0201] 예를 들어, 상기 제1 및 제2 미디어 데이터는 시간적 관계, 공간적 관계 및 품질 관계 중 적어도 하나에 따라 관련된다.

[0202] 실시예들에 따르면:

[0203] - 상기 제2 미디어 데이터는 상기 제1 미디어 데이터의 품질을 향상시키기 위한 향상 데이터를 포함하고,

[0204] - 향상 계층의 제1 미디어 데이터의 적어도 일부에 대한 우선순위 값이 업데이트될 때, 상기 향상 계층의 모든 미디어 데이터에 대한 우선순위 값들이 업데이트된다.

[0205] 예를 들어, 상기 제1 및 제2 미디어 데이터는 비디오 시간 세그먼트들을 포함하고, 상기 향상 미디어 데이터의 시작 시간은 상기 제1 미디어 데이터의 비디오 콘텐츠와 관련된 정보에 기초한다.

[0206] 실시예들에 따르면, 상기 제1 미디어 데이터의 비디오 콘텐츠와 관련된 상기 정보는 상기 기술 파일에 저장된다.

[0207] 실시예들에 따르면, 상기 전송 순서는 제1 미디어 데이터와 제2 미디어 데이터 간의 디코딩 관계들에 적어도 기초한다.

[0208] 실시예들에 따르면, 상기 전송 순서는 상기 미디어 데이터의 통계적 인기들에 적어도 기초한다.

[0209] 실시예들에 따르면, 상기 전송 순서는 상기 클라이언트 디바이스 측에서의 상기 미디어 데이터의 재생 시간에 적어도 기초한다.

[0210] 실시예들에 따르면, 상기 전송 순서는 상기 미디어 데이터의 추정 전송 시간에 적어도 기초한다.

[0211] 실시예들에 따르면, 상기 전송 순서는 상기 미디어 데이터에 대한 사용자 정의 관심들에 적어도 기초한다.

[0212] 상기 방법은:

[0213] - 상기 서버 디바이스에 제어 메시지들을 전송하는 단계 - 상기 제어 메시지는 상기 서버 디바이스가 현재 재생되고 있는 미디어 데이터를 식별하는 것을 가능하게 함 -, 및

[0214] - 상기 서버 디바이스에 의해, 상기 제어 메시지들에 기초하여, 정의된 업데이트된 전송 순서에 따라 상기 서버 디바이스로부터 상기 제2 미디어 데이터를 수신하는 단계를 포함할 수 있다.

[0215] 상기 방법은 순서 업데이트 확인 메시지를 상기 서버 디바이스로부터 수신하는 단계를 포함할 수 있다.

[0216] 예를 들어, 상기 제어 메시지들은 상기 클라이언트 디바이스의 버퍼 메모리의 사용과 관련되고, 상기 버퍼 메모리는 상기 클라이언트에 의해 재생되도록 미디어 데이터를 저장한다.

[0217] 예를 들어, 서버 디바이스는 전송된 제1 미디어 데이터의 레코드를 유지하고, 상기 레코드와 상기 버퍼 메모리의 상기 사용에 기초하여 현재 재생되고 있는 미디어의 식별이 수행된다.

[0218] 예를 들어, 상기 전송 순서 정보는 상기 알림 메시지들 내에서 수신된다.

[0219] 예를 들어, 상기 전송 순서 정보는 상기 알림 메시지들 뒤에 전용 메시지들 내에서 수신된다.

[0220] 계속해서 상기 전송 순서를 참고하여, 프록시 서버에 의해, 클라이언트 디바이스들과 서버 디바이스들 사이의 데이터 교환들을 관리하는 방법은:

[0221] - 전송 순서의 통지에 관하여 위에 정의된 방법을 구현하는 서버로부터, 클라이언트 디바이스에 재전송될 미디어 데이터를 수신하는 단계,

[0222] - 상기 미디어 데이터의 전송 순서에 기초하여, 상기 미디어 데이터에 대한 재전송 우선순위를 결정하는 단계, 및

[0223] - 결정된 상기 전송 우선순위에 기초하여, 상기 클라이언트 디바이스에 수신된 상기 미디어 데이터의 재전송을 수행하는 단계를 포함할 수 있다.

[0224] 상기 방법은 결정된 상기 재전송 우선순위에 기초하여, 수신된 상기 미디어 데이터를 저장하는 단계를 더 포함할 수 있다.

[0225] 상기 방법은:

[0226] - 상기 제2 양태에 따른 방법을 구현하는 클라이언트 디바이스로부터, 순서 업데이트 요청을 수신하는 단계,

[0227] - 상기 순서 업데이트 요청이 재전송될 미디어 데이터와 관련된다면, 상기 요청에 따라 상기 재전송 우선순위를 업데이트하는 단계, 및

[0228] - 상기 업데이트된 재전송 우선순위에 따라 상기 미디어 데이터의 재전송을 수행하는 단계를 더 포함할 수 있다.

[0229] 상기 방법은:

[0230] - 미디어 데이터에 대한, 제1 서버 디바이스로의 요청을 제1 클라이언트 디바이스로부터 수신하는 단계 - 상기 미디어 데이터는 제2 서버 디바이스로부터 제2 클라이언트 디바이스로의 재전송을 위해 상기 프록시 서버에 의해 저장됨 -,

[0231] - 상기 제1 및 제2 서버 디바이스들에 의해 상기 미디어 데이터와 각각 연관된 우선순위 값들을 결정하는 단계,

[0232] - 상기 제1 및 제2 클라이언트 디바이스들에 대한 각각의 스트리밍 제약 조건들에 따라 상기 우선순위 값들을 업데이트하는 단계, 및

[0233] - 상기 업데이트된 우선순위 값들에 따라 상기 제1 및 제2 클라이언트 디바이스들에 상기 미디어 데이터를 재전송하는 단계를 더 포함할 수 있고,

[0234] 상기 제1 및 제2 서버 디바이스들은 상기 제1 양태에 따른 방법을 구현하고 상기 제1 및 제2 클라이언트 디바이스들은 상기 제2 양태에 따른 방법을 구현한다.

[0235] 상기 방법은 상기 업데이트된 우선순위 값들에 관한 업데이트 통지들을 상기 제1 및 제2 서버 디바이스들에 전송하는 단계를 더 포함할 수 있다.

[0236] 본 발명의 다른 양태에 따르면 서버 디바이스와 클라이언트 디바이스 사이에 데이터를 스트리밍하는 방법이 제공되고, 이 방법은:

[0237] - 서버 디바이스에 의해 상기 제1 양태에 따른 방법을 수행하는 단계, 및

[0238] - 클라이언트 디바이스에 의해 상기 제2 양태에 따른 방법을 수행하는 단계를 포함한다.

[0239] 본 발명의 또 다른 양태에 따르면 프로그램 가능한 디바이스의 컴퓨터 수단에 로딩되어 실행될 때, 위에 정의된 방법들을 구현하기 위한 명령어들을 포함하는 컴퓨터 프로그램들 및 컴퓨터 프로그램 제품들이 제공된다.

[0240] 본 발명의 또 다른 양태에 따르면, 상기 제1 양태에 따른 방법들을 구현하도록 구성된 서버 디바이스가 제공된다.

[0241] 본 발명의 또 다른 양태에 따르면, 상기 제2 양태에 따른 방법들을 구현하도록 구성된 클라이언트 디바이스가 제공된다.

[0242] 특히 클라이언트 디바이스에 전송되는 데이터의 유형과 양을 관련된 클라이언트 디바이스의 특징들에 그리고 서버와 클라이언트 디바이스 간의 연결을 제공하는 네트워크들의 특성들에 적응시키기 위하여, 서버로부터 클라이언트 디바이스로의 미디어 데이터의 적응적 스트리밍을 위한 솔루션들이 제안되었다.

[0243] 이와 관련해서, DASH(Dynamic Adaptive Streaming over HTTP) 표준과 같은 일부 솔루션들은 분배될 리소스(또는 콘텐츠)의 복수의 버전을 저장하고 리소스를 요청하는 클라이언트 디바이스에게, 리소스를 나타내는 다양한 버전들과 이 버전들에 대한 각각의 포인터들(예를 들어, URL들)의 기술을 포함하는 기술 파일을 전송하는 것을 제안한다.

[0244] 그 후 클라이언트 디바이스는, 상기 기술 파일에 기초하여, 그것의 요구에 가장 잘 부합하는 리소스의 버전을 선택하고 대응하는 포인터를 이용하여 이 버전을 요청할 수 있다.

[0245] 이 솔루션은 상기 기술 파일이 미디어 데이터를 포함하지 않으므로(미디어 데이터에 대한 포인터들만 포함하므로) 경량이라는 점에서 유리하다. 그것은 클라이언트 디바이스가 관련 버전들을 그것의 사용을 위해 선택하게 함으로써 클라이언트 디바이스에게는 적합하지 않을 미디어 데이터의 교환을 회피한다. 게다가 그것은 HTTP에 기초한 현재의 웹 아키텍처에 적합하고 이미 배치된 캐싱 메커니즘들을 활용할 수 있다.

[0246] 그러나, 대신에, 이 솔루션은 미디어 데이터가 클라이언트 디바이스에서 수신되고 그 후 디코딩되어 표시될 수 있기 전에 클라이언트 디바이스와 서버 간에 몇몇 교환(또는 라운드트립)을 필요로 하고, 이는 시동 지연(start-up delay)을 야기한다.

[0247] 실시예들에서, 본 발명은, 적어도 미디어 아이템의 시간 세그먼트가 복수의 버전에 의해 표현되는, 미디어 아이템(예를 들어, 비디오)을 나타내는 데이터를 저장하는 서버로부터 그 미디어 아이템을 나타내는 미디어 데이터를 제공하는 방법을 제공하며, 그 방법은 상기 서버에 의해 구현되는 다음의 단계들:

[0248] - 상기 시간 세그먼트를 나타내는 버전들 및 상기 시간 세그먼트를 나타내는 버전들에 대한 각각의 포인터들의 기술을 포함하는 기술 파일에 대한 요청을 클라이언트 디바이스로부터 수신하는 단계;

[0249] - 상기 기술 파일에서 포인팅되는 데이터의 세트들 중에서 데이터를 선택하는 단계;

[0250] - 상기 기술 파일을 상기 클라이언트 디바이스에 전송하는 단계;

[0251] - 상기 선택된 데이터를 상기 클라이언트 디바이스에 푸시하는 단계

[0252] 를 포함한다.

[0253] 적절한 방식으로 선택된 데이터를 푸시하는 것(즉, 클라이언트 디바이스에 의해 요청받은 것은 아니지만, 아래 더 설명된 바와 같이 서버에 의해 선택된 데이터를 전송하는 것)에 의해, 하나 또는 몇몇 라운드트립(들)이 회피될 수 있고 따라서 미디어 데이터의 디코딩 및 표시가 더 빨리 시작될 수 있다.

[0254] 미디어 아이템은 예를 들어 비디오, 또는, 예를 들어, 오디오 트랙과 같은 오디오 아이템일 수 있다.

[0255] 위에 언급한 데이터의 세트들은 상기 시간 세그먼트들을 나타내는 버전들을 포함하지만, 아래 설명된 바와 같이, 초기화 데이터와 같은 다른 데이터도 포함할 수 있다는 것이 언급될 수 있다.

[0256] 방금 언급한 바와 같이, 상기 선택된 데이터는 클라이언트 디바이스의 디코더를 위한 초기화 데이터를 포함할 수 있다. 따라서 디코더는 클라이언트 디바이스가 초기화 데이터를 구체적으로 요청할 필요 없이, 그리고 따라서 더 빨리 초기화될 수 있다.

[0257] 위에 언급한 바와 같이, 상기 선택된 데이터는 또한 시간 세그먼트를 나타내는 상기 버전들 중 하나의 적어도 일부를 포함할 수 있다.

[0258] 데이터를 선택하는 상기 단계는 푸시될 데이터(예를 들어, 비디오 데이터)의 양을 추정하는 단계를 포함할 수 있고, 그 양은 어느 데이터가 선택되어야 할지를 결정할 때 이용될 수 있다. 상기 양은 상기 기술 파일에서 정의된 버퍼 시간에 기초하여 그리고/또는 상기 서버에 의해 결정된 대역폭 추정에 기초하여 추정될 수 있다.

[0259] 데이터를 선택하는 상기 단계는 상기 요청에 포함된 적어도 하나의 선호에 기초하여 그리고/또는 상기 서버와 상기 클라이언트 디바이스 사이의 이전 교환들로부터 얻어진 사용량 데이터에 기초하여 그리고/또는 상기 서버에 의한 상기 기술 파일의 분석에 기초하여 그리고/또는 상기 서버에 저장된 그리고 상기 기술 파일과 연관된 테이블에 기초하여 수행될 수 있다.

[0260] 가능한 실시예에 따르면, 상기 선택된 데이터를 푸시하는 단계 이전에 그와 관련된 푸시 약속을 전송하는 단계가 제공될 수 있다. 따라서 상기 클라이언트 디바이스는 푸시될 데이터에 관해, 이 데이터를 실제로 수신하기 전에, 알게 될 수 있다.

[0261] 상기 푸시 약속을 전송하는 상기 단계는 상기 기술 파일을 전송하는 상기 단계 이전에 수행될 수 있으므로, 초기 스테이지에서 상기 클라이언트 디바이스에 알리는 것이 가능하다.

[0262] 상기 푸시 약속은 예를 들어 상기 선택된 데이터의 식별을 포함한다.

[0263] 제안된 실시예에 따르면, 상기 서버는 상기 선택된 데이터와 연관된 신뢰 레벨을 결정하고 상기 푸시 약속은 상기 결정된 신뢰 레벨을 포함한다.

[0264] 아래에 제공된 상세 설명에서 설명된 가능한 구현에 따르면, 상기 서버는 상기 선택된 데이터를 형성하는 데이터의 블록들의 계층적 표현을 저장할 수 있다. 그러한 경우에, 다음의 단계들이 제공될 수 있다:

[0265] - 상기 클라이언트 디바이스로부터, 데이터의 블록을 푸시하지 말라는 명령을 수신하는 단계;

[0266] - 상기 계층적 표현에서 상기 데이터의 블록과 상기 데이터의 블록에 연결된 데이터의 블록들의 푸시를 취소하

는 단계.

[0267] 상기 제안된 방법은 상기 선택된 데이터와 연관된 신뢰 레벨을 결정하는 단계를 포함할 수 있고; 이때:

[0268] - 상기 결정된 신뢰 레벨이 미리 결정된 임계치보다 낮으면, 상기 선택된 데이터를 푸시하는 단계는 상기 클라이언트 디바이스의 디코더를 위한 초기화 데이터만을 푸시하는 단계를 포함하고;

[0269] - 상기 결정된 신뢰 레벨이 상기 미리 결정된 임계치보다 높으면, 상기 선택된 데이터를 푸시하는 단계는 상기 클라이언트 디바이스의 디코더를 위한 초기화 데이터와 상기 시간 세그먼트를 나타내는 상기 버전들 중 하나의 적어도 일부를 푸시하는 단계를 포함한다.

[0270] 본 발명의 실시예들은 또한, 적어도 미디어 아이템의 시간 세그먼트가 복수의 버전에 의해 표현되는, 미디어 아이템(예를 들어, 비디오)을 나타내는 데이터를 저장하는 서버로부터 그 미디어 아이템을 나타내는 미디어 데이터를 수신하는 방법을 제공하며, 그 방법은 상기 클라이언트 디바이스에 의해 구현되는 다음의 단계들:

[0271] - 상기 시간 세그먼트를 나타내는 버전들 및 상기 시간 세그먼트를 나타내는 버전들에 대한 각각의 포인터들의 기술을 포함하는 기술 파일에 대한 요청을 상기 서버에 전송하는 단계;

[0272] - 상기 기술 파일을 상기 서버로부터 수신하는 단계 - 상기 기술 파일은 데이터의 세트들에 대한 포인터들을 포함함 -;

[0273] - 비요청 데이터를 상기 서버로부터 수신하는 단계 - 상기 비요청 데이터는 상기 데이터의 세트들에 속함 - 를 포함한다.

[0274] 전술한 바와 같이, 비요청 데이터는 상기 클라이언트 디바이스의 디코더를 위한 초기화 데이터(그 경우 상기 비요청 데이터로 디코더를 초기화하는 단계가 제공될 수 있다) 및/또는 시간 세그먼트를 나타내는 상기 버전들 중 하나의 적어도 일부(그 경우 상기 비요청 데이터의 적어도 일부를 디코딩하는 단계가 제공될 수 있다)를 포함할 수 있다.

[0275] 상기 요청은 상기 클라이언트 디바이스에서의 디코딩을 정의하는 적어도 하나의 선호를 포함할 수 있고, 이는 상기 서버가 푸시될 미디어 데이터를 결정하는 데 도움을 줄 수 있다.

[0276] 상기 요청은 또한 상기 클라이언트 디바이스가 푸시된 데이터를 수락한다는 표시자를 포함할 수 있고, 그에 기초하여 서버는 효과적으로 데이터를 푸시하기로 결정할 수 있다.

[0277] 전술한 바와 같이, 비요청 데이터를 수신하는 단계 전에 그와 관련된 푸시 약속을 수신하는 단계가 제공될 수 있다. 푸시 약속을 수신하는 이 단계는 상기 기술 파일을 수신하는 단계 전에 일어날 수 있다.

[0278] 상기 푸시 약속은 비요청 데이터의 식별 및/또는 비요청 데이터와 연관된 신뢰 레벨을 포함할 수 있다

[0279] 다음과 같은 단계들이 클라이언트 디바이스에서 제공될 수 있다:

[0280] - 푸시 약속에 포함된 데이터에 기초하여 푸시 약속의 수락 또는 거절을 결정하는 단계;

[0281] - 거절의 경우에 상기 비요청 데이터를 푸시하지 말라는 명령을 전송하는 단계.

[0282] 다음과 같은 단계들이 이용될 수도 있다:

[0283] - 푸시 약속에 포함되고 비요청 데이터와 연관된 신뢰 레벨에 기초하여 푸시 약속의 수락 또는 거절을 결정하는 단계;

[0284] - 거절의 경우에 상기 비요청 데이터를 푸시하지 말하는 명령을 전송하는 단계.

[0285] 상기 비요청 데이터의 수신시에, 이들 데이터를 디코딩하기 전에 버퍼링하는 단계가 이용될 수 있다.

[0286] 푸시된 데이터는 초기화 데이터 및/또는 초기 미디어 데이터에만 대응하게 되어 있으므로, 다음과 같은 단계들이 구현될 수 있다:

[0287] - 상기 기술 파일에 기초하여 그리고 상기 푸시 약속에 포함된 데이터에 기초하여 요청될(즉, 푸시되도록 계획되지 않은) 데이터(예를 들어, 비디오 데이터)를 결정하는 단계;

[0288] - 상기 결정된 데이터에 대한 요청을 상기 서버에 전송하는 단계.

[0289] 본 발명의 실시예들은 또한 미디어 아이템(예를 들어, 비디오)을 나타내는 데이터를 저장하는 서버로부터 클라

이언트 디바이스로 상기 미디어 아이템을 나타내는 미디어 데이터를 스트리밍하는 방법을 제안하는데, 적어도 상기 미디어 아이템의 시간 세그먼트는 복수의 버전에 의해 표현되며, 이 방법은 다음의 단계들:

[0290] - 상기 클라이언트 디바이스가 상기 시간 세그먼트를 나타내는 상기 버전들 및 상기 시간 세그먼트를 나타내는 상기 버전들에 대한 각각의 포인터들의 기술을 포함하는 기술 파일에 대한 요청을 상기 서버에 전송하는 단계;

[0291] - 상기 서버가 상기 클라이언트 디바이스로부터 상기 요청을 수신하는 단계;

[0292] - 상기 서버가 상기 기술 파일에서 포인팅되는 데이터의 세트들 중에서 데이터를 선택하는 단계;

[0293] - 상기 서버가 상기 기술 파일을 상기 클라이언트 디바이스에 전송하는 단계;

[0294] - 상기 서버가 상기 선택된 데이터를 상기 클라이언트 디바이스에 푸시하는 단계;

[0295] - 상기 클라이언트 디바이스가 상기 서버로부터 상기 기술 파일을 수신하는 단계;

[0296] - 상기 클라이언트 디바이스가 상기 서버로부터 상기 선택된 데이터를 수신하는 단계를 포함한다.

[0297] 본 발명의 실시예들은 또한, 적어도 미디어 아이템의 시간 세그먼트가 복수의 버전에 의해 표현되는, 미디어 아이템(예를 들어, 비디오)을 나타내는 데이터를 저장하는 서버로부터 상기 미디어 아이템을 나타내는 미디어 데이터를 제공하는 디바이스를 제공하며, 상기 디바이스는:

[0298] - 상기 시간 세그먼트를 나타내는 상기 버전들 및 상기 시간 세그먼트를 나타내는 상기 버전들에 대한 각각의 포인터들의 기술을 포함하는 기술 파일에 대한 요청을 클라이언트 디바이스로부터 수신하도록 구성된 수신기;

[0299] - 상기 기술 파일에서 포인팅되는 데이터의 세트들 중에서 데이터를 선택하도록 구성된 선택 모듈;

[0300] - 상기 기술 파일을 상기 클라이언트 디바이스에 전송하도록 구성된 모듈;

[0301] - 상기 선택된 데이터를 상기 클라이언트 디바이스에 푸시하도록 구성된 모듈을 포함한다.

[0302] 본 발명의 실시예들은 또한 적어도 미디어 아이템의 시간 세그먼트가 복수의 버전에 의해 표현되는, 미디어 아이템(예를 들어, 비디오)을 나타내는 데이터를 저장하는 서버로부터 상기 미디어 아이템을 나타내는 미디어 데이터를 수신하는 디바이스를 제공하며, 상기 디바이스는:

[0303] - 상기 시간 세그먼트를 나타내는 상기 버전들 및 상기 시간 세그먼트를 나타내는 상기 버전들에 대한 각각의 포인터들의 기술을 포함하는 기술 파일에 대한 요청을 상기 서버에 전송하도록 구성된 모듈;

[0304] - 상기 서버로부터 상기 기술 파일을 수신하도록 구성된 모듈 - 상기 기술 파일은 데이터의 세트들에 대한 포인터들을 포함함 -;

[0305] - 상기 서버로부터 비요청 데이터를 수신하도록 구성된 모듈을 포함하고, 상기 비요청 데이터는 상기 데이터의 세트들에 속한다.

[0306] 마지막으로, 본 발명의 실시예들은 미디어 아이템(예를 들어, 비디오)을 나타내는 데이터를 저장하는 서버로부터 클라이언트 디바이스로 상기 미디어 아이템을 나타내는 미디어 데이터를 스트리밍하기 위해 서버와 클라이언트 디바이스를 포함하는 시스템을 제공하는데, 적어도 상기 미디어 아이템의 시간 세그먼트는 복수의 버전에 의해 표현되며,

[0307] - 상기 클라이언트 디바이스는 상기 시간 세그먼트를 나타내는 상기 버전들 및 상기 시간 세그먼트를 나타내는 상기 버전들에 대한 각각의 포인터들의 기술을 포함하는 기술 파일에 대한 요청을 상기 서버에 전송하도록 구성된 모듈을 포함하고;

[0308] - 상기 서버는 상기 클라이언트 디바이스로부터 상기 요청을 수신하도록 구성된 모듈, 상기 기술 파일에서 포인팅되는 데이터의 세트들 중에서 데이터를 선택하도록 구성된 선택 모듈, 상기 기술 파일을 상기 클라이언트 디바이스에 전송하도록 구성된 모듈 및 상기 선택된 데이터를 상기 클라이언트 디바이스에 푸시하도록 구성된 모듈을 포함하고;

[0309] - 상기 클라이언트 디바이스는 상기 서버로부터 상기 기술 파일을 수신하도록 구성된 모듈 및 상기 서버로부터 상기 선택된 데이터를 수신하도록 구성된 모듈을 포함한다.

[0310] 미디어 데이터를 제공하는 상기 방법 및 미디어 데이터를 수신하는 상기 방법에 대해 위에 제안된 옵션의 특징들은 미디어 데이터를 스트리밍하는 상기 방법 및 방금 언급한 다양한 디바이스들 및 시스템에도 적용된다.

도면의 간단한 설명

[0311] 본 발명의 다른 특징들 및 이점들은, 도 1a 내지 6과 더불어, 첨부된 도면들을 참조하여 보면 다음의 비제한적 예시적 실시예들의 설명으로부터 분명해질 것이다:

도 7a 및 7b는 실시예들에 따른 미디어 세그먼트 재순서화를 예시한다;

도 8은 실시예들에 따른 서버들에 의해 수행되는 예시적 단계들의 흐름도이다;

도 9는 실시예들에 따른 클라이언트들에 의해 수행되는 예시적 단계들의 흐름도이다;

도 10은 실시예들에 따른 프록시들에 의해 수행되는 예시적 단계들의 흐름도이다;

도 11은 실시예들에 따른 대역폭 측정을 예시한다.

도 12은 실시예들에 따른 비디오 재생 초기화를 예시한다.

도 13은 실시예들에 따른 디바이스들의 개요도이다;

도 14a는 클라이언트 측에서 본 발명의 일반적 단계들을 흐름도를 이용하여 예시한다;

도 14b는 서버 측에서 본 발명의 일반적 단계들을 흐름도를 이용하여 예시한다;

도 15a는 명시적 접근법에 기초하여 클라이언트 측에서 공유된 푸시 정책을 측정하는 단계들을 흐름도를 이용하여 예시한다;

도 15b는, 명시적 접근법이 이용될 때 서버 측에서 푸시 정책을 결정하는 단계들을 흐름도를 이용하여 예시한다;

도 16은 PushPolicy 노드가 서버에 의해 적용된 푸시 정책을 특정하는데 사용되는 MPD 문서를 도시한다;

도 17은, 공유된 푸시 정책 "PushPolicy"에 따르는, 푸시될 준비가 된 일부 세그먼트들을 식별하고 마킹하는 단계들을 흐름도를 이용하여 예시한다;

도 18a는 HTTP "푸시-정책" 헤더에서 전송된 푸시 정책을 이용하는 서버와 클라이언트 간의 통신의 예를 예시한다;

도 18b은 푸시 정책을 변경하기 위한 클라이언트의 요청을 이용하는 동일한 예를 예시한다;

도 20은 알림 메시지들을 통합하는 실시예들에 따르는, 서버 측의 프로세스의 단계들을 흐름도를 이용하여 예시한다;

도 21은 푸시 정책을 선언하기 위해 HTTP 헤더들을 이용할 때 서버 측의 프로세스의 단계들을 흐름도를 이용하여 예시한다;

도 22는 푸시 정책을 선언하고 공유하기 위해 HTTP 요청을 이용할 때 클라이언트 측에서 프로세스의 단계들을 흐름도를 이용하여 예시한다;

도 23은 SupplementalProperty 요소가 문서의 계층적 레벨에서 서버에 의해 적용된 푸시 정책을 특정하는데 사용되는 MPD 문서를 도시한다;

도 24는 XPath 기반 푸시 정책을 위한 예로서 이용된 MPD 문서를 도시한다;

도 25는 푸시 정책을 적용하기 전에, 예를 들어 웹 페이지와 같은 우선순위 트리에서 요소들의 재순서화를 예시한다;

도 26은 본 발명의 실시예들에 따른, DASH 고속 시작을 획득하기 위해 서버 및 클라이언트 디바이스에 의해 각각 구현되는 예시적 방법들을 도시한다;

도 27은 DASH 고속 시작을 위해 서버에 의해 구현되는 예시적 방법을 도시한다; 및

도 28은 DASH 고속 시작을 위해 클라이언트 디바이스에 의해 구현되는 가능한 방법을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0312] 이하, HTTP 2.0 프로토콜을 구현하는 DASH 기반 네트워크들과 관련하여 본 발명의 실시예들이 설명된다. 스트

리밍되는 데이터는, 예를 들어 비디오 데이터이다. 본 발명의 실시예들은 DASH 네트워크들로 한정되지 않는다.

[0313] 데이터를 클라이언트 디바이스로 스트리밍하는 통신 네트워크의 서버 디바이스는 푸시 특징을 구현하며, 이 푸시 특징에 따라 서버 디바이스는 데이터 요소들의 전송에 대한 클라이언트로부터의 명시적인 요청들 없이도 데이터 요소들을 클라이언트로 전송할 수 있다.

[0314] 서버와 클라이언트는 서버로 하여금 푸시 약속들을 결정하고 대응하는 데이터를 실제로 전송하게 하는 푸시 정책들을 공유할 수 있다. 이러한 공유로 인해, 클라이언트는 일부 쓸모없는 데이터의 푸시를 예상하여 그러한 푸시를 취소할 수 있다. 이것은 서버의 처리는 물론, 네트워크 사용을 줄이는데, 그 이유는 PUSH_PROMISE 프레임들이 전송 전에 취소될 수 있기 때문이다.

[0315] 특정 실시예들에서, 서버는 명시적 비요청 데이터 요소들의 전송을 알리는 그의 푸시 약속들 내에서 서버가 데이터 요소들을 전송하기를 의도하는 순서에 관한 순서 정보를 지시할 수 있다. 데이터 요소들의 순서는 우선순위 값들, 예로서 HTTP/2에 따른 우선순위 값들을 이용하여 정의될 수 있다.

[0316] 푸시 약속들의 수신시에, 클라이언트 디바이스는 서버에 의해 의도된 전송의 순서를 미리 결정할 수 있으며, 따라서 클라이언트는 제안된 순서가 그 자신의 원하는 순서와 매칭되지 않는 경우에는 제안된 순서에 반대할 수 있다. 예를 들어, 클라이언트 디바이스는 우선순위 값들을 업데이트하고, 업데이트된 우선순위 값들을 서버로 전송할 수 있다. 따라서, 서버는 클라이언트의 요구와 더 잘 매칭시키기 위해 새로운 우선순위 값들에 기초하여 전송 순서를 변경할 수 있다. 서버는 업데이트된 우선순위들을 이용하여 추가 데이터 전송들을 처리할 수 있다.

[0317] 실시예들에 따르면, 클라이언트는 서버에 데이터 요소들의 전송의 전체 재순서화 또는 부분 재순서화를 요청할 수 있다.

[0318] 도 7a를 참조하여 전체 재순서화가 설명된다. 클라이언트는 단계 700 동안 서버에 미디어 표현 기술(Media Presentation Description)(이하, MPD)을 요청한다. 서버는 단계 701 동안 클라이언트로 전송할 MPD를 검색하고, 푸시할 대응하는 데이터 요소들을 식별한다. 도 7a의 예에서, 서버는 "Data 1.1", "Data 1.2" 및 "Data 1.3"을 푸시할 데이터 요소들로서 식별한다. 이러한 요소들은 예를 들어 데이터 세그먼트들이다. 요소 "Data X.1"은 데이터 X에 대한 기본 계층을 나타내고, 요소 "Data X.2"는 데이터 X에 대한 향상 계층을 나타내고, "Data X.3"은 데이터 X에 대한 추가 향상 계층을 나타낸다. 서버는 데이터 요소들에 대한 특정 전송 순서를 정의한다. 서버는 다가오는 푸시 데이터 요소들을 알리기 위해 각각의 우선순위 값을 클라이언트로 전송될 PUSH_PROMISE 프레임들과 관련시킨다. 이어서, 서버는 단계 702 동안 관련 우선순위들을 갖는 PUSH_PROMISE 프레임 "P1.1", "P1.2" 및 "P1.3"와, MPD를 전송한다. 이어서, MPD 및 푸시 약속을 전송한 직후에, 단계 703 동안, 서버는 "Data 1.1" 요소에 대응하는 데이터 프레임 및 정의된 전송 순서에서 "Data 1.1", "Data 1.2" 및 "Data 1.3"에 이어지는 세그먼트들인 요소 "Data 2.1", "Data 2.2" 및 "Data 2.3"에 각각 대응하는 PUSH_PROMISE 메시지 "P2.1", "P2.2" 및 "P2.3"을 클라이언트로 전송한다. 단계 703의 데이터 프레임 및 푸시 약속의 수신과 동시에, 클라이언트는 MPD 및 "P1.1", "P1.2" 및 "P1.3" PUSH_PROMISE 프레임들의 수신 후에 향상 계층 "Data 1.2"가 추가 향상 계층 "Data 1.3"에 비해 더 낮은 우선순위를 갖는 것으로 결정한다. 따라서, 클라이언트는 단계 704 동안 "Data 1.2" 우선순위를 낮추기 위해 우선순위 업데이트 프레임을 전송한다. 우선순위 업데이트 요청의 수신시에, 서버는 단계 705 동안 전송 스케줄을 변경한다. 따라서, "Data 1.2"의 전송은 "Data 1.3"이 전송된 후로 연기된다. 게다가, 서버는 MPD를 이용하여, "Data 1.2"와 관련된 세그먼트들을 링크한다. 서버는 또한 "Data 2.2"를 식별하고, 그 우선순위를 낮춘다.

[0319] 도 7b를 참조하여 부분 재순서화가 설명된다. 도 7b의 단계 710 내지 714는 도 7a의 단계 700 내지 704와 실질적으로 동일하다. 우선순위 업데이트 프레임의 수신 후에, 서버 거동은 전술한 단계 705와 대비하여 다르다. 단계 715 동안, 서버는 이미 "Data 1.2"의 전송을 시작하였으며, 전송을 더 진행한다. 그 세그먼트에 대해서는 어떠한 우선순위 변경도 없다. 그럼에도, 서버는 링크된 세그먼트들, 즉 본 예에서는 "Data 2.2"의 우선순위를 업데이트한다. 우선순위 변경이 고려되었다는 사실을 알리기 위해, 서버는 "Data 2.2"에 대한 우선순위 업데이트 메시지를 전송할 수 있다. 따라서, 클라이언트는 변경을 알 수 있다.

[0320] 본 발명의 실시예들은 서버들이 고품질 비디오 부분들을 사전에 충분히 양호하게 푸시할 수 있어서 비디오의 전체 부분이 고품질로 재생될 수 있는 사용 예들에서 구현될 수 있다. 예를 들어, 비디오는 저품질로 재생되는 부분 1, 고품질로 재생되는 부분 2, 및 저품질로 재생되는 부분 3으로 분할될 수 있다. 클라이언트와 서버 간의 대역폭은 저품질의 실시간 스트리밍을 가능하게 하지만, 고품질의 실시간 스트리밍은 허용하지 않는다. 그

러한 경우, 서버는 부분 1을 부분 2의 향상과 인터리빙할 수 있다. 부분 1이 재생되면, 향상 부분 2도 이용 가능하며, 서버는 고품질로 재생될 부분 2의 기본 계층을 동일 부분 2의 향상과 연계하여 전송한다. 따라서, 서버는 전체 부분 2가 고품질로 재생되는 것을 보증한다. 이어서, 부분 3이 전송된다. 사용자 경험을 방해하는 품질 플리커링이 완화될 수 있으며, 제한된 수의 순간들에서 품질 스위칭만이 발생한다. 서버는 상이한 품질 레벨로 언제 스위칭할지를 알기 위한 최상의 위치에 있는데, 그 이유는 서버가 비디오 콘텐츠를 알기 때문이다.

[0321] 도 8은 실시예들에 따른 푸시 기반 DASH 미디어 스트리밍을 구현하는 서버에 의해 수행되는 단계들의 흐름도이다. 단계 800 내지 812는 일반 원리들을 설명한다. 단계 820 내지 827은 클라이언트로부터 우선순위 피드백의 관리를 더 구체적으로 다룬다.

[0322] 단계 800 동안, 서버는 클라이언트로부터 요청 R을 수신한다. 이 요청은 통상적으로 MPD 파일을 참조함으로써 특정 미디어를 식별한다. 이어서, 서버는 단계 801 내지 810을 포함하는 반복 프로세스를 수행한다. 프로세스는 정의된 순서에 따라 데이터를 전송하는 단계를 포함한다. 전송의 순서는 클라이언트의 피드백에 따라 업데이트된다. 데이터가 전송되면, 데이터는 클라이언트에 의해 수신 및 재생된다. 이어서, 서버는 전송할 새로운 데이터를 식별하고, 프로세스가 계속되고, 기타 등등이다.

[0323] 제1 반복은 단계 801로부터 시작하며, 이 단계 동안, 전송될 데이터가 식별된다. 반복 프로세스의 첫 번째 수행의 경우, 클라이언트가 비디오 재생을 가능한 한 빠르게 시작하는 것을 가능하게 하기 위해 고속 시작 접근법이 이용될 수 있다. 게다가, 서버는 또한 미디어의 챕터들(chapters)로의 세분을 식별할 수 있다. 서버가 클라이언트가 일반적으로 챕터들을 이용하여 내비게이션한다는 것을 아는 경우, 서버는 사실상 미디어의 시작에 대응하는 세그먼트들뿐만 아니라, 미디어 내의 제1 챕터들의 시작에 대응하는 세그먼트들도 선택할 수 있다. 반복의 첫 번째 수행 후에, 서버는 또한 연결이 미디어의 더 높은 품질의 표현의 전송을 지원할 수 있다는 것을 검출할 수 있다. 따라서, 서버는 해상도 또는 품질 스위치가 언제 수행되어야 하는지를 식별할 수 있다.

[0324] 서버가 푸시할 세그먼트들의 리스트를 식별하면, 서버는 이러한 세그먼트들에 대한 전송 순서를 정의한다. 전송 순서는 단계 802 동안 각각의 푸시되는 세그먼트에 대한 초기 우선순위 값들을 계산하는 데 사용된다. 순서는 여러 파라미터에 기초할 수 있다.

[0325] 제1 파라미터는 상이한 세그먼트들 간의 관계들일 수 있으며, 예를 들어 일부 세그먼트들은 다른 세그먼트들을 올바르게 디코딩하는 데 이용될 수 있어야 한다. 따라서, 이용될 수 있어야 하는 세그먼트들은 상기 다른 세그먼트들보다 높은 우선순위들을 할당받는다.

[0326] 제2 파라미터는 과거의 통계로부터 수집될 수 있는 비디오 세그먼트들의 인기도일 수 있다. 일례로서, 유튜브 URL들을 이용하여, 비디오 내의 특정 시간들이 어드레싱될 수 있다. 이러한 URL들과 관련된 링크들을 클릭할 때, 지정된 시간에 비디오 재생을 시작하는 데 필요한 비디오만이 검색된다. 게다가, 비디오가 챕터화되고 있는 경우, 각각의 챕터의 시작은 일반적으로 챕터 시작들 사이의 세그먼트들보다 사용자들로부터 더 자주 검색된다. 따라서, 챕터 시작의 세그먼트들은 챕터 사이의 세그먼트들보다 높은 우선순위들을 할당받는다.

[0327] 제3 파라미터는 타임라인일 수 있으며, 재생에 더 가까운 비디오 세그먼트의 우선순위는 나중에 재생될 비디오 세그먼트의 우선순위보다 높다.

[0328] 제4 파라미터는 세그먼트를 실제로 전송하는 데 소비되는 추정 시간일 수 있다. 비디오 세그먼트가 클 때, 전송에 오랜 시간이 걸리며, 따라서 전송은 가능한 한 빨리, 즉 높은 우선순위로 시작되어야 한다.

[0329] 2개의 세그먼트가 동일한 우선순위들을 갖는 경우, 대응하는 데이터 프레임들은 전송 동안 인터리빙될 수 있다.

[0330] 미디어 콘텐츠 내에서 관심 영역들이 식별되는 경우, 대역폭이 고품질 표현을 위해서는 충분히 크지 않지만, 저품질 표현을 위해서는 충분히 클 경우에, 서버는 관심 영역에 대해서만 향상 계층을 선택할 수 있다.

[0331] 우선순위들이 계산되면, 서버는 단계 803 동안 우선순위 값들을 포함하는 PUSH_PROMISE 프레임들을 전송한다. PUSH_PROMISE 프레임들의 전송을 시작하기 위해 모든 세그먼트들의 식별이 필요한 것은 아니다. MPD가 푸시될 세그먼트들에 대해 전송되어야 하는 경우(단계 804), MPD가 전송된다(단계 805). 세그먼트 전송은 단계 806 동안 동시에 시작된다.

[0332] PUSH_PROMISE 프레임들이 클라이언트에 의해 수신되면, 서버는 우선순위 업데이트 변경들을 수신하고, 이어서 그에 따라 그의 전송 스케줄을 변경할 수 있다(단계 807 내지 808 및 단계 820 내지 828). 세그먼트들을 전송하는 동안, 서버는 우선순위 변경 메시지들의 수신을 기다린다. 우선순위 변경 메시지가 수신되는 경우(단계 807), 서버는 그에 따라 세그먼트들을 재순서화하고, 세그먼트 전송을 계속한다(단계 808). 모든 세그먼트들이

전송되면(단계 809-1), 서버는 미디어의 끝까지 미디어 스트리밍을 계속하기 위해 반복 프로세스를 재개한다. 미디어의 끝에 도달할 때(단계 809-2), 서버는 다른 미디어의 스트리밍을 자동으로 시작해야 하는지 체크한다(단계 810). 다른 미디어가 스트리밍되어야 하는 경우(예), 서버는 스트리밍할 새로운 미디어를 식별하고(단계 811), 단계 801로부터 프로세스를 재개한다. 새로운 데이터가 스트리밍되지 않아야 하는 경우, 프로세스가 종료된다(단계 812).

[0333] 단계 808의 클라이언트로부터의 우선순위 피드백의 관리는 단계 820 동안 우선순위 업데이트 변경 메시지의 수신으로부터 시작된다. 클라이언트가 세그먼트 푸시를 취소하는 경우에 아래의 단계들도 수행될 수 있으며, 이러한 사례는 실제로는 해당 세그먼트에 최저 우선순위를 할당하는 것에 대한 등가물로서 간주될 수 있다.

[0334] 우선순위 업데이트 변경 메시지의 수신시에, 서버는 단계 821 동안 관련 세그먼트를 식별한다. 이어서, 서버는 세그먼트 전송의 재순서화를 진행한다(단계 822, 823). 세그먼트가 이미 전송된 경우, 프로세스가 종료된다. 세그먼트가 전송되고 있는 경우에는, 서버 구현에 따라, 서버는 (예를 들어, 너무 복잡하므로) 전송 변경을 거부될 수 있거나, 전송될 나머지 데이터를 사실상 리스케줄링할 수 있다.

[0335] 데이터의 리스케줄링은 다음과 같이 수행될 수 있다. 서버는 푸시할 비디오 세그먼트들(및/또는 푸시되고 있는 비디오 세그먼트들)의 리스트를 저장한다. 이 리스트는 서버에 의해 설정된 우선순위들에 따라 순서화된다. 이어서, 서버는 세그먼트에 대한 새로운 우선순위 값을 설정한다. 이어서, 리스트가 재순서화되고, 그에 따라 대응하는 비디오 세그먼트의 전송이 더 일찍 또는 더 늦게 행해진다.

[0336] 비디오 세그먼트가 재순서화되면, 서버는 실제로 이러한 우선순위 변경을 다른 관련 비디오 세그먼트들에 적용하기로 결정할 수 있다. 클라이언트가 향상 계층의 일부인 비디오 세그먼트의 우선순위를 상승시키는 경우, 서버는 이 향상 계층의 모든 세그먼트들의 우선순위를 상승시킬 수 있다. 역으로, 클라이언트가 기본 비디오 세그먼트 계층의 우선순위를 낮추는 경우, 이 세그먼트와 시간적으로 관련된 모든 세그먼트들의 우선순위가 낮아질 수 있다. 이러한 프로세스는 단계 824 내지 827에서 설명된다. MPD 및 리스케줄링된 비디오 세그먼트에 기초하여, 서버는 관련 세그먼트들의 리스트를 식별한다(단계 824). 관계는 시간, 공간, 품질 기반 등일 수 있다. MPD는 잠재적 관계들을 더 양호하게 나타내도록 향상될 수 있다. 구체적으로, (둘 이상의 비디오 세그먼트를 재생하는 데 필요한) 초기화 세그먼트의 우선순위가 낮아지거나 높아질 때, 모든 관련 세그먼트들이 리스케줄링될 수 있다. 이것은 기본 계층 세그먼트들 및 향상 세그먼트들에도 해당할 수 있다. 각각의 식별된 관련 세그먼트에 대해, 서버는 관련 세그먼트의 전송이 변경되어야 하는지 테스트한다(단계 825). 변경되어야 하는 경우, 서버는 각각의 세그먼트에 대한 새로운 우선순위 값을 계산하고(단계 826), 그에 따라 세그먼트 전송을 리스케줄링한다(단계 827). 단계 820 동안 수신된 새로운 우선순위 값과 단계 821 동안 식별된 세그먼트의 초기 우선순위 값 간의 차이를 오래된 값에 더함으로써 새로운 우선순위 값이 계산될 수 있다. 각각의 관련 세그먼트가 테스트되었을 때 프로세스가 종료된다(단계 828).

[0337] 서버는 WINDOW_SIZE 프레임들과 같은 제어 흐름 메시지들도 수신할 수 있다. 이러한 메시지들은 서버로 하여금 클라이언트가 현재 무엇을 재생하고 있는지를 식별하는 것을 가능하게 할 수 있다. 일부 추가적인 버퍼 공간이 클라이언트 측에서 이용 가능할 때, 일부 데이터, 통상적으로는 가장 오래된 데이터가 버퍼로부터 제거된 것으로 추정될 수 있다. 서버가 전송된 데이터의 이력을 유지하는 경우, 서버는 어느 데이터가 제거되었는지를 식별할 수 있다. 따라서, 서버가 클라이언트의 캐시 순서를 아는 경우, 서버는 클라이언트가 현재 어느 비디오 세그먼트들을 재생하고 있는지를 알 수 있다. 이러한 순서는 캐싱된 데이터를 타임라인에 따라 순서화하는 것을 가능하게 하는 MPD에 기초할 수 있다. 이어서, 서버는 예를 들어 클라이언트 시간 건너뛰기를 검출할 수 있다. 서버는 클라이언트가 비디오 챕터들의 건너뛰기를 계속할 수 있도록 다음 챕터의 시작을 미리 빠르게 전송함으로써 반응할 수 있다.

[0338] 우선순위들을 갖는 PUSH_PROMISE 프레임들의 전송은 다양한 방식으로 행해질 수 있다는 점에 유의해야 한다. PUSH_PROMISE 프레임은 클라이언트에 의해 개시되는 열린 스트림과 관련되어야 한다. 실시예들에 따르면, 단계 800 동안 클라이언트에 의해 행해지는 초기 스트림은 항상 열린 상태로 유지될 수 있다. 다른 실시예들에 따르면, PUSH_PROMISE 프레임은 서버에 의해 열린 스트림 내에서 전송된다. 이 경우, 클라이언트는 PUSH_PROMISE 프레임이 부모 클라이언트 개시 스트림에 의해 전송되는 것으로 간주한다. 따라서, 클라이언트는 특정 PUSH_PROMISE 프레임에 대응하는 가상 요청의 올바른 헤더들을 계산할 수 있다.

[0339] 다른 실시예들에 따르면, 우선순위 메시지가 PUSH_PROMISE와 연계하여 전송된다. 첫 번째 가능성은 그것을 PUSH_PROMISE 프레임 내의 헤더로서 전송하는 것이다. 다른 가능성은 대응하는 PUSH_PROMISE 프레임에 의해 예약된 스트림 ID를 이용하여 PRIORITY 프레임을 전송하는 것이다. 세 번째 가능성은 PUSH_PROMISE 프레임, 이어

서 (스트림을 열기 위한) 대응하는 HEADERS 프레임, 이어서 PRIORITY 프레임을 이러한 새로 열린 스트림 상에서 전송하는 것이다.

[0340] 클라이언트의 버퍼를 더 제어하기 위해, 서버는 클라이언트에 의해 캐싱된 세그먼트의 새로운 표현을 전송할 수 있다. 이러한 새로운 표현의 일부로서 전송되는 헤더들 내에서, HTTP 캐시 지령들은 세그먼트를 캐싱 가능하지 않게 함으로써 사실상 제거하도록 클라이언트에 요청하는 데 사용될 수 있다. 이것은 클라이언트 측에서 버퍼 공간을 복구하는 것을 가능하게 할 수 있다. HTTP/2 제어 흐름이 사용될 수 있다. 이어서, 서버는 추가 데이터를 푸시할 수 있다.

[0341] 서버는 각각의 비디오 세그먼트에 대한 우선순위 값들을 전송할 수 있다. 서버는 또한 특정 세그먼트들에 대한 우선순위 값들을 전송할 수 있다. 서버가 현재 PUSH_PROMISE 프레임에 대한 우선순위 값을 전송하지 않은 경우, 클라이언트는 서버로부터 전송된 최종 우선순위 값으로부터 우선순위 값을 계산할 수 있다. 예를 들어, 클라이언트는 관련 우선순위 값을 갖지 않는 새로운 PUSH_PROMISE 프레임이 수신될 때마다 우선순위 값을 증가시킬 수 있다. 따라서, PUSH_PROMISE 프레임들은 그룹화될 수 있으며, 따라서 특정 세그먼트의 우선순위를 업데이트하는 것은 그룹의 모든 세그먼트들의 우선순위들도 업데이트할 것이다.

[0342] 클라이언트 측의 프로세스가 도 9를 참조하여 설명된다.

[0343] 클라이언트는 주어진 시간에 이용 가능한 콘텐츠를 재생할 수 있어야 한다. 그러나, 클라이언트는 잠재적 버퍼 제한들 및 처리 시간을 해결해야 한다. 클라이언트는 서버에 의해 제안되는 전송 순서가 클라이언트의 버퍼에서 이용 가능한 메모리 공간과 매칭되고 클라이언트에 의해 현재 재생되는 콘텐츠와 매칭되는지를 체크해야 한다.

[0344] 제1 단계 900 동안, 클라이언트는 서버에 연결하고, MPD 파일을 요청한다. 이어서, 클라이언트는 단계 901 동안 MPD 파일을 검색하고, 데이터의 수신을 대기한다(단계 902). 데이터가 수신될 때, 클라이언트는 데이터가 푸시 약속인지를 체크한다(단계 903). 푸시 약속이 수신된 경우, 이것은 새로운 비디오 세그먼트가 서버에 의해 전송되고 있다는 것을 의미한다. 클라이언트는 푸시 약속을 처리한다. 구체적으로, 클라이언트는 단계 904 동안 서버에 의해 제안된 우선순위 값들을 확인할 수 있다. 클라이언트가 현재 세그먼트 또는 다른 약속된 세그먼트에 대한 우선순위 값들을 변경하기를 원할 경우(단계 905), 클라이언트는 새로운 우선순위 값을 계산하여 서버로 전송한다(단계 906).

[0345] 클라이언트가 비디오 데이터를 수신하는 경우(단계 907), 클라이언트는 비디오 세그먼트를 MPD 파일에 링크하고(단계 908), 비디오 데이터를 저장한다(단계 909). 비디오 데이터를 MPD 파일에 링크하는 것은 비디오 세그먼트가 비디오를 디코딩하기 위해 더 사용될 때 클라이언트가 비디오 세그먼트를 검색하는 것을 가능하게 한다(단계 911). 이것은 또한, 예를 들어 연속 비디오 세그먼트들이 그룹화될 경우에 비디오 데이터의 효율적인 저장을 제공할 수 있다(단계 909).

[0346] 버퍼 저장 제약 조건들은 우선순위를 더 변경할 수 있다. 따라서, 클라이언트는 우선순위 값이 변경되어야 하는지를 다시 체크할 수 있으며, 필요한 경우에 서버와 통신할 수 있다(단계 905, 906).

[0347] 클라이언트가 비디오 재생을 시작하거나 계속할 준비가 되면(단계 910), 클라이언트는 그의 캐시로부터 다음 시간 슬롯 비디오 세그먼트들을 검색하고(단계 911), 비디오를 디코딩 및 재생한다(단계 912). 단계 911의 일부로서, 클라이언트는 어느 비디오 세그먼트들이 이용 가능한지를 알기 위해 그의 캐시에 질의할 수 있다. 디폴트로, 클라이언트는 이용 가능한 모든 비디오 세그먼트들, 구체적으로는 존재할 경우에 모든 향상 세그먼트들을 사용할 수 있다. 클라이언트는 서버가 콘텐츠를 선택하게 할 수 있으며, 일반적으로 모든 세그먼트들이 클라이언트에 의해 사용되어야 한다. 일부 세그먼트들이 (오디오 영어 트랙들 및 불어 트랙들과 같이) 연계하여 사용될 수 없는 경우, 클라이언트는 우선 미사용 세그먼트들을 버려야 한다. 모든 클라이언트들이 캐시 상태에 액세스할 수 있는 것은 아니라는 점에 유의해야 하며, 구체적으로 웹 애플리케이션들은 통상적으로 웹 브라우저 캐시에 액세스하지 못한다. 그러한 경우, 서버는 푸시되는 세그먼트들의 리스트를 웹 애플리케이션 클라이언트로 직접 전송할 수 있다. 예를 들어, 이러한 정보는 웹 소켓 연결을 이용하여 서버로부터 클라이언트로 교환될 수 있다.

[0348] 비디오가 재생 및 디코딩됨에 따라, 대응하는 비디오 세그먼트들이 버퍼로부터 제거될 수 있다. 따라서, 클라이언트는 WINDOW_SIZE 프레임을 이용하여 그의 이용 가능 버퍼 사이즈를 업데이트한다. 클라이언트는 최근에 재생된 비디오 세그먼트들을 유지하여, 사용자가 제한된 기간 동안 비디오를 되감기하는 것을 가능하게 할 수 있다. 흐름 제어 업데이트 메커니즘은 또한 사용자가 빨리 감기/시간 건너뛰기를 행할 때 사용될 수 있다. 클

라이언트는 오래된 저장 비디오 콘텐츠를 제거하여, 새로운 콘텐츠를 위한 공간을 만들 수 있으며, 이러한 변화를 WINDOW_SIZE 프레임을 이용하여 서버에 알린다. 서버가 WINDOW_SIZE 프레임을 수신할 때, 서버는 어느 비디오 세그먼트들이 제거되었는지를 계산하고, 이어서 전술한 바와 같이 클라이언트가 실제로 무엇을 재생하고 있는지를 식별할 수 있다.

[0349] 이하, 단계 904가 더 상세히 설명된다.

[0350] 클라이언트는 모든 푸시 약속된 비디오 세그먼트들의 리스트를 유지한다. 이 리스트는 푸시 약속 프레임들에서 발견되는 우선순위 정보에 따라 순서화된다. 먼저, 잠재적 동결 비디오 문제들에 대해 체크한다. 이용 가능 대역폭의 추정 및 순서화된 비디오 세그먼트 리스트에 기초하여, 각각의 세그먼트의 전송 시작 및 종료 시간들이 추정될 수 있다. 이러한 시간들에 기초하여, 각각의 비디오 세그먼트가 비디오 재생을 위해 사용되어야 하는 시간에 이용 가능한지를 테스트할 수 있다. 약속된 비디오 세그먼트가 그의 대응하는 비디오 재생 사용 후에 전달될 것으로 예상되는 경우, 그의 우선순위는 상승되어야 한다. 따라서, 비디오 세그먼트는 푸시 약속된 비디오 세그먼트 리스트 순서에서 위로 이동된다. 정확한 우선순위 값을 계산하기 위해, 비디오 세그먼트가 정시에 전달되는 것을 가능하게 하고 현재 비디오 세그먼트 위치에 가장 가까운 비디오 세그먼트 내의 위치가 검색된다. 이어서, 우선순위는 비디오 세그먼트 새 위치 전후에 있는 리스트 내의 비디오 세그먼트들의 우선순위들 사이의 값으로 설정된다.

[0351] 비디오 세그먼트 우선순위들을 변경하기 위해 클라이언트에 의해 다른 팩터들도 사용될 수 있다. 예를 들어, 클라이언트가 소정의 챕터 스위칭을 행할 것을 예상하고 있는 경우, 클라이언트는 사실상 챕터들을 시작하는 모든 비디오 세그먼트들, 구체적으로는 대응하는 초기화 세그먼트들의 우선순위를 상승시킬 수 있다.

[0352] 실시예들에 따르면, 클라이언트 측 흐름 제어는 스트림별 흐름 제어를 불능화하고 연결별 흐름 제어만을 유지하는 것을 포함한다. 연결별 윈도 사이즈는 클라이언트가 임의의 주어진 시간에 실제로 저장할 수 있는 비디오의 최대 양을 정의한다. 클라이언트 및 서버는 이러한 윈도 사이즈를 감소 또는 증가시키기 위해 초기화 시간에 그리고 연결 동안 협상할 수 있다. 서버가 소정의 HD 콘텐츠를 푸시하기를 원하는 경우, 서버는 클라이언트에게 윈도 사이즈를 증가시킬 것을 요청할 수 있다. 연결 대역폭이 낮은 경우, 서버는 비디오의 특정 부분에 대한 HD 콘텐츠의 전송을 사전에 잘 예상할 필요가 있으며, 이 경우에 버퍼 사이즈는 더 커져야 한다.

[0353] 전송 순서는 버퍼가 단일 사이즈를 가질 때 중요한 문제일 수 있다. 구체적으로, 버퍼가 데이터로 채워짐에 따라, 우선순위 순서는 더욱더 중요해진다. 중요한 제약 조건은 비디오가 결코 동결되지 않는 것이다. 버퍼가 충분히 비어 있는 한, 서버는 큰 세그먼트들과 같은 다양한 비디오 세그먼트들을 사전에 푸시하여, 효율적인 빨리 감기 및 챕터 건너뛰기를 제공할 수 있다. 버퍼가 거의 완전히 채워지면, 푸시할 비디오 세그먼트들은 재생되고 있는 비디오 세그먼트들에 가능한 한 가까워야 한다. 이러한 푸시 거동은 서버가 클라이언트 버퍼에 관한 정확한 정보를 가질 경우에 서버에 의해 행해질 수 있다. 이것은 우선순위 업데이트 메커니즘을 이용하여 클라이언트에 의해 구현될 수도 있다.

[0354] 자동 비디오 스위칭의 경우, 도 9의 흐름도는 푸시 약속 체크의 일부로서 새로운 MPD의 푸시를 검출함으로써 확장될 수 있다(단계 903). MPD 푸시가 검출될 때, 클라이언트는 단계 908의 일부로서 새로운 비디오의 세그먼트들의 수신을 시작할 수 있다. 따라서, 클라이언트는 비디오 데이터와 관련된 MPD를 식별해야 한다. 주어진 MPD에 대해 비디오 재생이 종료되면(단계 902), 비디오 재생을 계속하기 위해 새로운 MPD가 사용될 수 있다. 클라이언트는 실제로 이전 MPD에 링크된 모든 비디오 세그먼트들을 분출할 수 있다.

[0355] 도 10을 참조하여, DASH 인식 프록시의 거동이 설명된다. 서버로부터 푸시된 세그먼트를 수신할 때, 그것을 최종 클라이언트로 푸시하기 위해 프록시가 요구되지는 않는다. 그러나, DASH 스트리밍의 경우에는, 그렇게 하는 것이 양호한 수단(또는 디폴트 거동)으로서 간주될 수 있다.

[0356] 프록시는 우선순위 처리는 물론, 전송될 푸시 데이터와 관련하여 서버 및 클라이언트 거동들을 조정할 수 있다. 프록시는 사실상 클라이언트와 관련된 우선순위들과 서버와 관련된 우선순위들을 독립적으로 처리할 수 있다. 게다가, 서버는 주어진 클라이언트에 대해 필요한 것보다 많은 데이터를 푸시할 수 있으며, 프록시는 추가적인 푸시 데이터를 검색하여 다른 클라이언트들로부터의 요청들을 충족시킬 수 있다.

[0357] 서버는 여러 이유로 비디오 세그먼트를 푸시할 수 있다. 예를 들어, 비디오 세그먼트는 최종 클라이언트에게 유용할 것으로 생각되는 경우에 푸시될 수 있다. 비디오 세그먼트는 비디오 세그먼트가 여러 번 사용될 수 있고 프록시들로 푸시할 가치가 있는 것으로 생각되는 경우에도 푸시될 수 있다.

[0358] 첫 번째 경우, 프록시들은 일반적으로 비디오 세그먼트를 클라이언트로 전송한다. 프록시들은 클라이언트 또는

프록시 네트워크 상태, 예를 들어 클라이언트 무선 상태를 최적화하기 위해 그의 전송을 연기할 수 있다. 하나의 예시적인 경우는 고속 시작 비디오 재생 및 대역폭 추정을 위한 세그먼트 푸시일 수 있으며, 이 경우에 데이터는 가능한 한 빠르게 클라이언트로 전송되어야 한다. 서버가 데이터를 프록시들로 푸시하는 것에 관심이 있는 경우, 프록시들은 그들이 비디오 세그먼트가 클라이언트에게 유용할 것이라는 것을 알기 위한 수단을 갖는 경우를 제외하고는 비디오 세그먼트를 클라이언트로 자동 전송하지 않을 수 있다. 클라이언트들로 전송되지 않을 수 있는 비디오 세그먼트들의 식별을 가능하게 하기 위해, 특정 우선순위 값이 사용될 수 있다. 우선순위 값의 사용은 프록시가 도달하는 다양한 프레임들의 처리를 최적화하기 위해 우선순위 값을 항상 체크하는 것을 가능하게 한다.

[0359] 도 10은 3개의 흐름도를 포함한다. 하나의 흐름도는 푸시된 세그먼트들을 필터링하는 프로세스(단계 1000 내지 1008)와 관련된다. 다른 흐름도는 세그먼트가 이미 한 클라이언트에게 약속되어 있는 동안에 다른 클라이언트에 의해 요청될 때 수행되는 프로세스(단계 1010 내지 1015)와 관련된다. 또 다른 흐름도는 우선순위 변경들의 관리(단계 1020 내지 1026)와 관련된다.

[0360] 푸시 세그먼트들을 필터링하는 프로세스는 통상적으로 PUSH_PROMISE 프레임 또는 관련 DATA 프레임을 수신할 때 푸시 데이터 이벤트의 수신(단계 1000)으로부터 시작된다. 프록시는 데이터가 높은 우선순위를 갖는지를 체크한다(단계 1001). 데이터는 그들의 우선순위 값이 전송되고 있는 다른 세그먼트들의 우선순위 값들보다 훨씬 클 경우에 높은 우선순위를 갖는 것으로 간주될 수 있다. 데이터는 그의 우선순위 값이 고속 시작 또는 대역폭 추정과 같은 특별한 의미를 갖는 경우에도 높은 우선순위를 갖는 것으로 간주될 수 있다. 데이터가 높은 우선순위를 갖는 경우, 그들은 가능한 한 빠르게 클라이언트로 전송된다(단계 1002). 이어서, 프록시는 데이터를 저장할지를 결정한다(단계 1003, 1004). 이러한 결정은 푸시 데이터 스트림을 여는 대응하는 PUSH_PROMISE 프레임 또는 대응하는 HEADERS 프레임을 수신할 때 한 번 행해질 수 있다. 이러한 결정은 또한 프록시 캐시 상태, 비디오의 사용 구상, 비디오 소스의 인기도 또는 다른 기준들에 기초할 수 있다. 프록시는 세그먼트가 하나 이상의 클라이언트에 의해 동시에 요청되는 동안 푸시되는 경우에 비디오 세그먼트를 저장한다. 비디오 세그먼트들은 세그먼트들이 고속 시작으로서 식별되는 경우에도 저장될 수 있다.

[0361] 데이터가 높은 우선순위를 갖지 않는 경우, 프록시는 데이터가 낮은 우선순위를 갖는지를 체크한다(단계 1005). 낮은 우선순위의 데이터는 클라이언트로의 전송이 생략되지만 서버에 의해 프록시들과 같은 네트워크 중개자들과 관련된 것으로 간주되는 데이터일 수 있다. 프록시는 먼저 데이터를 클라이언트로 전송할지를 결정한다(단계 1006). 이러한 결정은 푸시 데이터 스트림을 여는 대응하는 PUSH_PROMISE 프레임 또는 대응하는 HEADERS 프레임을 수신할 때 한 번 행해질 수 있다. 그러한 것으로 결정되는 경우, 프록시는 대응하는 프레임을 클라이언트로 전송한다(단계 1002). 이어서, 프로세스는 데이터를 저장할지를 결정한 후에 종료된다.

[0362] 서버와 프록시 사이에 협상된 우선순위 값은 클라이언트와 프록시 사이에 협상된 우선순위 값과 다를 수 있다. 따라서, 데이터가 통상의 우선순위를 갖는 경우(즉, 낮은 우선순위도 높은 우선순위도 갖지 않는 경우), 프록시는 세그먼트 우선순위 값이 프록시에 의해 관리되는지를 체크한다. 도 10(단계 1020 내지 1026)에 도시된 바와 같이, 프록시는 데이터가 전송되어야 하는 시간을 스케줄링하기 위해 클라이언트-프록시 값을 사용하며, 프록시는 모든 전송될 비디오 관련 프레임들의 리스트를 유지한다. 이러한 프레임들은 해당 순서에 따라 전송되기 전에 우선순위 값들에 따라 순서화된다.

[0363] 프록시가 우선순위 업데이트 프레임을 수신하는 경우(단계 1010), 프록시는 관련 비디오 세그먼트를 식별한다(단계 1011). 우선순위 값이 프록시에 의해 관리되지 않는 경우(단계 1012), 프록시는 우선순위 업데이트 프레임을 서버로 전송한다(단계 1013). 그렇지 않은 경우, 프록시는 이러한 새로운 우선순위 값을 저장하고, 그에 따라 비디오 세그먼트 전송을 재순서화한다(단계 1014). 잠재적 충돌이 발생하는 경우, 특히 서버로부터의 비디오 세그먼트 전달이 클라이언트 요구에 대해 너무 늦을 것으로 예상되는 경우, 프록시는 우선순위 값을 서버로 전송할 수 있다.

[0364] 단계 1020 내지 1026은 서버에 의해 다른 클라이언트에 대해 이미 약속된(단계 1021) 비디오 세그먼트에 대한 한 클라이언트로부터의 요청을 수신(단계 1020)하는 프록시의 사례와 관련된다. 그러한 요청에 주어지는 우선순위에 따라, 프록시는 클라이언트의 요청을 충족시키는 최소 프록시-서버 우선순위를 계산한다(단계 1022). 이러한 계산은 서버-프록시 전달 시간이 프록시-클라이언트 예상 전달 시간보다 이른 것을 보증하는 프록시-서버 우선순위 값을 계산함으로써 행해진다. 계산된 우선순위가 현재 설정된 우선순위보다 낮은 경우에 우선순위가 변경되며(단계 1023), 이 경우에 프록시는 우선순위 업데이트 메시지를 서버로 전송할 것이며(단계 1024), 프록시는 이러한 비디오 세그먼트 우선순위를 프록시에 의해 관리되는 것으로서 마킹할 것이고, 따라서 프록시

는 비디오 세그먼트를 그의 2개의 클라이언트로 그들의 요구에 대한 최상의 시간에 전송할 것이다. 이 프로세스와 유사하게, 프록시는 여러 클라이언트로부터 동일 세그먼트에 대한 여러 우선순위 업데이트를 수신할 수 있으며, 이 경우에 프록시는 실제로 모든 클라이언트들을 충족시키는 최저 우선순위 값을 전송할 수 있다.

[0365] 도 11을 참조하여, 클라이언트가 푸시 데이터 이벤트를 수신하고, 그의 우선순위 값이 서버가 대역폭을 측정하기 위해 그를 사용하기를 원한다는 것을 지시하는 실시예가 설명된다. 대역폭 측정은 라운드 트립 시간들을 계산하기 위한 능동 또는 수동 측정들을 통해 TCP/IP 패킷들을 이용하여 행해질 수 있다. 문헌 Saubhasik et al. "Bandwidth Estimation and Rate Control in BitVampire"에서 발견되는 바와 같이, 라운드 트립 시간들에 기초하여 이용가능 대역폭이 계산될 수 있다. 이러한 계산은 HTTP/2 제어 흐름의 효과들을 잠재적으로 고려할 수 있다. 일부 데이터 프레임들이 대역폭 추정을 위해 사용된다는 통지를 가능하게 함으로써, HTTP/2 제어 흐름 없이 이용 가능한 대역폭이 추정될 수 있다.

[0366] 프로세스는 단계 1100으로부터 시작되며, 이 단계 동안 서버로부터 푸시 데이터 프레임이 수신된다. 이어서, 스트림의 관련 우선순위가 서버가 대역폭을 측정하고 있다는 것을 지시하는지를 체크한다(단계 1101). 그러한 경우, 전용 버퍼가 최대화된다(단계 1102). 대안으로서, 스트림 흐름 제어가 불능화될 수 있다. 수신 노드가 프록시인 경우(단계 1103), 프록시는 세그먼트 데이터를 전송할 수 있다. 그렇지 않은 경우, 클라이언트는 세그먼트를 저장할지를 결정한다(단계 1104). 클라이언트는 푸시 세그먼트를 저장한다(단계 1105). 어느 경우에나, 클라이언트는 연결별 윈도에 대한 WINDOWS_UPDATE의 형태로 수신 확인 통지를 서버로 전송한다(단계 1106). 이어서, 이러한 수신 확인 통지는 연결 대역폭을 추정하기 위해 서버에 의해 사용될 것이다. 클라이언트가 프록시인 경우, 프록시는 푸시 데이터를 가능한 한 빠르게 전송한다(단계 1108). 최종 클라이언트로부터 수신 확인 통지가 수신될 때, 프록시는 그것을 또한 서버로 전송한다(단계 1109, 1110).

[0367] 이용 가능 대역폭을 추정하기 위해, 서버는 데이터 프레임의 전송 시간과 수신 확인 통지 메시지의 수신 시간 사이의 차이로서 계산되는 전송 데이터 프레임의 라운드 트립 시간을 이용할 수 있으며, 이들 2개의 페어링은 예를 들어, 윈도 사이즈 업데이트와 동일해야 하는 데이터 프레임 사이즈에 기초한다. 라운드 트립 시간들은 하나 이상의 비디오 세그먼트의 다양한 데이터 프레임들로부터 계산될 수 있다. 정확도를 향상시키기 위해, 데이터 프레임들은 다양한 사이즈들을 가질 수 있다. 비디오 세그먼트를 상이한 사이즈의 여러 DATA 프레임으로 분할하는 것은 서버에 의해 수행될 수 있다. 서버는 네트워크 계층이 DATA 프레임들을 여러 개의 TCP/IP 패킷(따라서, 더 작은 DATA 프레임)으로 분할하지 않거나 전송될 콘텐츠를 버퍼링하지 않고 여러 DATA 프레임들을 TCP/IP 패킷으로 병합하는 것을 보증하는 것만이 필요하다. 그러한 측정들에 기초하여, 서버가 어떤 비디오 표현을 사용할지를 실제로 결정하는 데 사용할 수 있는 이용 가능 대역폭을 계산하기 위해 표준 기술들이 이용될 수 있다(전술한 문헌에서 일례가 발견될 수 있다).

[0368] 도 12를 참조하여, 초기 비디오 재생의 예가 설명된다. 서버는 고속 시작 우선순위를 이용하여 데이터를 푸시한다. 데이터는 아마도 낮은 비디오 세그먼트 레이트를 갖고, 클라이언트는 그러한 데이터를 수신하고, 수신 확인 통지를 서버로 전송하며, 따라서 서버는 대역폭을 추정하고, 최적 표현으로 스위칭할 수 있는 것이 고려된다. 클라이언트측 프로세스는 단계 1200 내지 1207에서 설명된다. 서버측 프로세스는 단계 1210 내지 1215에서 설명된다.

[0369] 클라이언트 프로세스는 푸시 데이터를 수신하는 단계 1200으로부터 시작된다. 이어서, 클라이언트는 우선순위가 고속 시작 값을 갖는지를 체크한다(단계 1201). 그러한 경우, 클라이언트는 통상적으로 전용 버퍼를 최대화한다(단계 1202). 이러한 최대화는 푸시 데이터의 PUSH_PROMISE를 수신할 때 수행된다. 이어서, 데이터가 저장되고(단계 1203), 클라이언트는 수신 확인 통지를 WINDOW_UPDATE 프레임을 이용하여 서버로 전송한다(단계 1204). 이어서, 클라이언트는 비디오 재생을 시작하기에 충분한 데이터가 이용 가능한지를 체크한다(단계 1205). 그러한 경우, 비디오 재생이 시작된다(단계 1206). 그렇지 않은 경우, 클라이언트는 데이터 재생을 시작하기에 충분한 데이터가 이용 가능할 때까지 더 많은 데이터를 대기한다(단계 1207).

[0370] 서버 프로세스는 고속 시작 우선순위를 갖는 세그먼트 데이터 프레임들을 전송하는 단계 1211로부터 시작된다(단계 1210). 이어서, 서버는 이용 가능 대역폭의 계산(단계 1212)을 가능하게 하는 수신 확인 통지를 수신한다(단계 1211). 충분한 측정들이 획득되면, 서버는 최적 표현을 선택하고(단계 1213), 최적 표현 세그먼트들의 푸시를 시작한다(단계 1214). 서버는 표현을 언제 스위칭할지를 결정한다. 이것은 적어도 두 가지 이익을 갖는다. 첫째, 서버는 측정들이 충분히 정확할 때 이를 알 수 있으며, 그러한 경우에 바로 하나의 해상도로부터 다른 해상도로 스위칭할 수 있으며, 클라이언트는 소정의 지연을 처리하는 것이 필요할 것이다. 둘째, 서버는 사용자 경험을 덜 방해하는 시간에 하나의 해상도로부터 다른 해상도로 스위칭하기로 결정할 수 있다. 사실상,

서버는 비디오 콘텐츠에 대한 지식을 갖는다. 특히, 해상도 스위치가 최상으로 구상될 수 있는 시간들에 대한 정보를 이용하여 MPD가 증대될 수 있다.

[0371] 본 발명은 향상된 스트리밍 방법과 관련되며, 이러한 방법에서는 서버측에서 클라이언트 디바이스로부터 제1 미디어 데이터와 관련된 요청이 수신되고; 요청 없이 클라이언트 디바이스로 전송될 제2 미디어 데이터가 식별되고; 이어서 상기 요청에 응답하여 상기 제1 미디어 데이터와 관련된 데이터가 상기 클라이언트 디바이스로 전송되며, 상기 제2 미디어 데이터를 각각 식별하는 적어도 하나의 알림 메시지가 알림 메시지 또는 메시지들을 클라이언트 디바이스로 전송하기 위한 뷰와 함께 준비된다.

[0372] 클라이언트 측에서, 제1 미디어 데이터와 관련된 요청이 서버 디바이스에 전송되고; 그리고 상기 요청에 응답하여 상기 제1 미디어 데이터와 관련된 데이터가 상기 서버 디바이스로부터 수신된다.

[0373] 향상된 스트리밍 방법은 일부 미디어 데이터를 푸시하기 위한 서버의 결정들과 그러한 데이터에 대한 클라이언트의 요구 간의 불일치들을 감소시킨다. 하기로부터 명백한 바와 같이, 서버와 클라이언트는 푸시 정책을 공유하여, 그들 둘 다는 클라이언트에 의해 요청된 임의의 미디어 데이터로부터 동일한 미디어 데이터가 푸시되도록 결정한다. 푸시 정책은 푸시할 데이터를 결정하는 방법을 정의하고, 요청된 데이터에 링크된 어느 리소스들이 푸시될지(GET 요청 후에), 그리고 어떻게 (예를 들어, 어느 순서로) 그들이 푸시되는지 결정하기 위한 규칙으로서 알려질 수 있다. 보통, 링크된 리소스는 하나의 문서, 예를 들어 MPD 파일(멀티미디어 데이터를 위한 DASH 콘텍스트에서) 또는 HTML 문서와 같은 매니페스트 파일을 이용하여 결정된다.

[0374] 그 결과, 공유된 푸시 정책에 기초하여, 클라이언트는 서버로부터의 쓸모없는 미디어 데이터의 전송을 회피하기 위해, 더 정확하게는 취소하기 위해 서버의 거동을 예측할 수 있다. 따라서 클라이언트와 서버 사이에 통신 네트워크의 대역폭 사용이 감소된다. 게다가, HTTP 요청들 및 PUSH_PROMISE 취소의 수가 감소되고, 이것은 특히 낮은 대기 시간 라이브 비디오 스트리밍을 위해 애플리케이션의 대기 시간을 낮춘다.

[0375] 본 발명에 따르면, 서버는 클라이언트 디바이스로의 제2 비요청 미디어 데이터의 전송 및 식별을 서버 디바이스가 추진하도록 클라이언트 디바이스와 공유된 푸시 정책을 이용할 수 있다. 특히, 서버는 클라이언트 디바이스에 전송될 제2 비요청 미디어 데이터를 서버 디바이스가 결정하도록 하기 위해, 제2 미디어 데이터를 결정하는 방법을 정의하는, 그리고 클라이언트 디바이스와 공유되는 푸시 정책을 이용할 수 있다. 이에 따라, 클라이언트는 클라이언트 디바이스에 의해 요청되지 않은 서버 디바이스에 의해 전송될 제2 미디어 데이터를 클라이언트 디바이스가 결정하도록 하기 위해, 제2 미디어 데이터를 결정하는 방법을 정의하는, 그리고 서버 디바이스와 공유되는 푸시 정책을 이용할 수 있다.

[0376] 도 14a는 클라이언트 측에서의 본 발명의 일반적인 단계들을 흐름도를 이용하여 도시하는 반면에, 도 14b는 서버 측에서의 본 발명의 일반적인 단계들을 흐름도를 이용하여 도시한다.

[0377] 도 1d 및 도 1e를 참조하여 설명된 프로세스와 비교하여, 부가적 단계들 1400 및 1402는 서버와 클라이언트 각각이 서로 공유되고, 그에 따라 이용될 푸시 정책을 결정하는 것을 가능하게 만든다.

[0378] 제1 실시예들에 따르면, 공유된 푸시 정책은 묵시적 푸시 정책이며, 이것은 클라이언트와 서버가 무엇이 공유될 푸시 정책인지를 서로 알리기 위해 (명시적) 정책 데이터를 교환하지 않음을 의미한다. 공유된 푸시 정책에 대한 묵시적 접근법의 구현은 서버 디바이스 및 클라이언트 디바이스 양쪽에서 "제2 미디어 데이터 결정 알고리즘"이라고 지칭되는, 동일한 알고리즘을 이용하는 것을 포함하고, 이 알고리즘은 서버 디바이스 및 클라이언트 디바이스가 요청된 제1 미디어 데이터로부터 동일한 제2 미디어 데이터를 결정할 수 있게 한다.

[0379] 예를 들어, 알고리즘은 클라이언트와 서버의 셋-업 동안 또는 특정 표준에 관련해서 미리 결정된다. 알고리즘의 전형적인 예는 매니페스트 파일의 파싱 순서에서 요청된 리소스 다음에 N 개의 리소스들을 푸시하는 것에 존재할 수 있으며, 여기서 N은 미리 정해진 수, 예를 들어 4이다.

[0380] 도면들을 참조하면, 단계들 1400 및 1402는 묵시적 푸시 정책의 경우에, 푸시될 리소스들을 식별하기 위한 미리 정해진 알고리즘을 메모리에 로딩시에 존재한다(서버 측에서의 단계 1403).

[0381] 클라이언트는 예상되는 PUSH_PROMISE의 수를 추정하기 위한 그리고 원하지 않는 푸시 데이터에 대한 취소 메시지들을 준비하기 위해 그렇게 결정된 푸시 정책을 예를 들어, 단계 1401에서 효율적으로 이용할 수 있다.

[0382] 예를 들어, 이것은 결과적으로 서버가 제2 비요청 미디어 데이터의 일부의 전송을 취소하도록 요청하는 취소 요청을 클라이언트 디바이스로부터 수신하도록 할 것이어서, 서버 디바이스는 대응하는 준비된 알림 메시지를 전송하지 않는다. 그것의 일부에 관해서, 서버 디바이스가 제2 비요청 미디어 데이터의 일부를 식별하는 알림 메

시지를 전송하지 않도록 추진하기 위해, 클라이언트는 따라서 서버 디바이스에, 제2 비요청 미디어 데이터의 일부의 전송을 취소하도록 요청하는 취소 요청을 전송할 것이다. 혹자는 알림 메시지가 서버 디바이스로부터 전송되거나 또는 클라이언트 디바이스에 의해 수신되기 전에 그러한 취소가 발생할 수 있다는 것을 이해할 수 있다. 이 접근법은 예를 들어, 클라이언트가 다른 버전의 매체로 스위칭하기로 결정할 때 유용할 수 있다. 그러한 상황에서, 그것은 이전 버전에 대해 푸시된 세그먼트들을 취소하도록 결정할 수 있다.

[0383] 이 알고리즘을 이용하여 푸시될 리소스들의 지식 덕분에, 클라이언트는 서버로부터의 대응하는 PUSH_PROMISE를 대기할 필요 없이 후속의 리소스들을 검색하기 위해, 서버에 제2 요청을 동시에 행할 수 있다. DASH의 경우에, 클라이언트에 대한 이러한 가능성은, 나중에 수신될 PUSH_PROMISE와 제2 요청이 간섭하지 않을 것을 보장하면서, 클라이언트의 지연을 감소시키는 것을 가능하게 한다.

[0384] 클라이언트는 또한 필요한 다른 리소스를 요청할 수 있는데, 이는 그것이 알고리즘의 결과로부터, 이들 다른 필요한 리소스들이 곧 푸시되지 않을 것이라고 결정할 경우이다.

[0385] 제2 실시예들에 따르면, 공유된 푸시 정책은 클라이언트와 서버 간의 교환들에서, 명시적으로 전체 규칙(즉, 알고리즘 또는 알고리즘의 파라미터들)을 정의함으로써, 또는 양측에 미리 정의된 푸시 정책들에 대한 참조를 이용하여 정의된다. 이것은 서버가 서버의 푸시 정책을 기술하는 푸시 정책 정보를 우선 결정하도록 요구한다. 그 후 푸시 정책 정보는 푸시 정책을 클라이언트와 공유하기 위해 클라이언트에 전송된다. 이에 따라, 클라이언트는 따라서 서버 디바이스로부터, 공유되는 푸시 정책을 기술하는 푸시 정책 정보를 수신한다.

[0386] 명시적 접근법의 한가지 이점은, 각각의 클라이언트에 대해 또는 각각의 멀티미디어 표현(예를 들어, 각각의 MPD)에 대해, 그들의 처리 특성을 더 잘 충족시키기 위해서, 상이한 푸시 정책이 서버에 의해 이용될 수 있다는 사실에 의존한다. 도 15a는 명시적 접근법에 기초하여 클라이언트 측에서의 공유된 푸시 정책을 결정하는 단계 1400을 흐름도를 이용하여 도시하는 반면에, 도 15b는 명시적 접근법이 이용될 때 서버 측에서 푸시 정책을 결정하는 단계 1402를 흐름도를 이용하여 도시한다.

[0387] 도 15b에 도시된 바와 같이, 서버는 단계 1504에서 푸시 정책을 선언하기 위한 메시지를 생성하고, 그것을 공유하기 위해, 그 후 단계 1505에서 그것을 클라이언트에 전송한다. 선언 메시지에서 푸시 정책을 기술하는 정보는 푸시 정책 정보로서 지칭된다.

[0388] 아래 기술되는 도 16 내지 도 18은 어떻게 푸시 정책이 선언되고 클라이언트에 전송되는지에 대한 예시적인 상세 사항들을 나타낸다.

[0389] 단계 1402에서 결정된 바와 같은 푸시 정책을 이용하여 푸시될 리소스들은 이후 단계 1504에서 생성된 푸시 정책 선언 메시지에 정의된 선택 알고리즘(또는 제2 미디어 데이터 결정 알고리즘)에 의해 단계 1403에서 식별된다.

[0390] 도 15a에 도시되는 바와 같이 클라이언트 측에서, 클라이언트는 동일한 선택 알고리즘을 적용함으로써 소정의 리소스 요청에 대해 푸시될 리소스들을 미리 식별할 수 있다. 이것은 클라이언트가 서버에 의해 푸시될 데이터를 미리 결정하는 것을 가능하게 하고, 그러므로 푸시 데이터의 효율적인 관리 및 적절한 경우 GET 요청들의 수의 감소를 보장한다.

[0391] 동일한 선택 알고리즘을 적용하기 위해, 클라이언트는 서버에 의해 적용되는 푸시 정책을 기술하는 푸시 정책 정보를 수신한다.

[0392] 다양한 푸시 정책 선언 방법들이 이용될 수 있다.

[0393] 일 실시예에서, 푸시 정책 선언은, 입력 파라미터들로서, 푸시될 리소스들을 포함하는 문서(일반적으로 DASH에 대한 매니페스트 파일)에 대응하는 DOM 트리 및 요청 R을 취하고, 푸시될 리소스들의 순서화된 리스트를 출력하는 JavaScript 프로그램 덕분에 공유된다. 이 실시예에서, 푸시 정책 정보는 서버 디바이스로부터 클라이언트 디바이스에 전송된 웹 페이지에 삽입된 JavaScript 프로그램을 포함한다.

[0394] 다른 실시예들에서, 푸시 정책은 매니페스트 파일 내에 기술된다. 즉, 공유된 푸시 정책을 이용하여 서버 디바이스로부터 클라이언트 디바이스에 전송되는 기술 파일 내에, 공유된 푸시 정책을 기술하는 푸시 정책 정보가 삽입된다. 기술 파일은 제1 미디어 데이터를 포함하는 미디어 데이터와 관련되고, 푸시될 제2 비요청 미디어 데이터를 결정하기 위해 양측에 의해 이용되는 기술 정보를 포함한다.

[0395] DASH에서, 기술 파일은 예를 들어, MPD 파일이다. 하기에서의 기술은 주로 DASH와 MPD 파일들에 기초한다. 그

러나, 동일한 접근법이 스무스(Smooth) 스트리밍 또는 HTTP 라이브 스트리밍과 같은 다른 매니페스트 기반 스트리밍 방법들에 적용된다.

[0396] 특정 실시예들에 따라, 푸시 정책 정보는 기술 파일에서 식별될 제2 비요청 미디어 데이터의 양을 정의하는 제1 푸시 속성을 포함한다. 이것은 하나의 요청 R이 클라이언트로부터 수신된 후에 푸시될 세그먼트들의 수를 지정하는 것을 가능하게 한다.

[0397] 이것은 PushPolicy 노드(1600)가 서버에 의해 적용되는 푸시 정책을 지정하는 데 사용되는 MPD 문서를 도시하는 도 16에 의해 예시된다.

[0398] 이 예에서, PushPolicy 노드(1600)는 푸시 속성 즉, GET 요청이 수신된 후에 푸시될 세그먼트들의 수를 선언하기 위한 "SegmentIdx"를 포함한다. 예를 들어, 클라이언트가 그것의 GET 요청에서 세그먼트(1601)를 요청하면, 그것은 응답으로서, MPD 문서의 파싱 순서에서 다음 2개의 세그먼트들에 대한 PUSH_PROMISE 프레임을 수신할 것이다. 이 예에서, 제1 푸시 속성은 상대적으로 기술 파일 내에 요청된 제1 미디어 데이터와 관련해서 제2 비요청 미디어 데이터를 식별한다. 더 일반적으로, 푸시될 K 개의 세그먼트들의 미리 정해진 개수가 푸시 정책 값을 정의하기 위해 이용된다. 따라서, 클라이언트에 의해 요청된 각각의 세그먼트에 대해, 서버는 K 개의 다음 세그먼트들을 푸시할 것이다.

[0399] 도 16의 예(1600)는 단일 푸시 속성을 나타내지만, 여러 개의 푸시 속성들이 있을 수 있다. 각각의 푸시 속성은 푸시될 세그먼트들을 선택하기 위한 매니페스트를 나타내는 DOM(Document Object Model) 트리의 노드들 상의 제약 조건을 나타낼 수 있다. 도 4b의 이전 예를 참조하면, 푸시 정책 노드(1600)는 미디어 데이터가 속하는 기간 요소를 지칭하는 기간 속성 "PeriodIdx", 미디어 데이터의 AdaptationSet 요소를 지칭하는 적응 속성 "AdaptationSetIdx", 미디어 데이터의 표현 요소, 즉 인코딩 버전(특정 코덱, 해상도, 또는 비트레이트...)을 지칭하는 표현 속성 "RepresentationIdx", 및 소정의 표현 내의 세그먼트를 지칭하는 세그먼트 속성 "SegmentIdx"를 포함하는 미디어 데이터 속성들(MPD 요소들 및/또는 속성들)을 이용하여 기술 파일(MPD 파일) 내에 기술되는 미디어 데이터를 지칭할 수 있다.

[0400] 이러한 기존 미디어 데이터 속성들에 기초하여, 푸시 정책 정보는 제2 비요청 미디어 데이터를 식별하기 위한, 미디어 데이터 속성 또는 속성들의 제약 조건을 정의하는 제2 푸시 속성을 적어도 포함할 수 있다.

[0401] 예를 들어, 푸시 속성은 푸시할 세그먼트를 선택하기 위한 기간의 제약 조건을 지정하기 위한 PeriodIdx 속성과 관련될 수 있다; 다른 하나는 적응의 제약 조건을 지정하기 위한 AdaptationSetIdx 속성과 관련될 수 있다; 다른 하나는 표현의 제약 조건을 지정하기 위한 RepresentationIdx 속성과 관련될 수 있다; 게다가, 상기 제1 푸시 속성은 SegmentIdx 속성과 관련된다.

[0402] 푸시 속성이 존재하지 않거나 비어있을 때, 관련된 미디어 데이터 속성은 제약되지 않는 것으로서 간주되어야 한다.

[0403] 푸시 속성들의 값은 다음의 구문을 이용할 수 있다:

[0404] 푸시 속성 = [연산자] 피연산자

[0405] 여기서 "연산자"는 선택적이고, 요청된 세그먼트와 관련해서 푸시될 세그먼트들을 정의하기 위해 값 '+' 또는 '-'를 취하고("+"는 후를 의미하고 "-"는 전을 의미함), 여기서 "피연산자"는 0 이상의 정수 값 또는 와일드카드 파라미터로서 '*'이다.

[0406] 도 17은 공유된 푸시 정책 "PushPolicy"에 따라 푸시될 준비가 된 것으로서의 일부 세그먼트들을 식별하고 마킹하기 위한 단계들을 흐름도를 이용하여 도시한다. 이 흐름도는 단계 1403을 예시한다.

[0407] 우선, 서버는 단계 1700에서 매니페스트 파일에서 요청된 세그먼트를 식별한다. 요청은 이 세그먼트의 식별된 "reqSegIdx"를 포함한다.

[0408] 매니페스트 파일 MPD 내의 각각의 노드 타입에서, 인덱스 값은 각각의 노드에 속성이 있다고 생각한다. 그 값은 매니페스트 파일에서 출현의 순서로 각각의 노드에 대해 증분된다.

[0409] 다음에, 요청된 세그먼트(즉, GET 요청에 특정된 세그먼트)에 대응하는 Period, AdaptationSet, Representation, 및 SegmentURL의 인덱스들은 요청된 세그먼트가 도달될 때까지 전체 MPD를 파싱함으로써 검색된다.

[0410] 푸시 정책에 정의된 푸시 속성들의 연산자 및 피연산자 값들은 푸시될 세그먼트들이 어느 노드들에서 정의되는지를 식별하는데 이용된다("+" 또는 "-" 연산자와 연관될 때, 푸시될 세그먼트들의 양을 정의하는 SegmentIdx 속성을 제외함).

[0411] 연산자가 특정되지 않을 때, 피연산자 값은 푸시될 데이터가 검색되어야 하는 노드의 인덱스를 식별한다. 예를 들어, 제1 푸시 속성 "SegmentIdx"가 연산자를 갖지 않을 경우에, 푸시될 특정 세그먼트의 식별자는 기술 파일 내에 있다. 하나의 대안에서, 연산자가 특정되지 않을 때, 피연산자 값은 범위 값들을 식별할 수 있고, 예를 들어 "SegmentIdx=2-5"는 인덱스가 2, 3, 4, 및 5와 동일한 세그먼트들을 리턴할 것이다.

[0412] 그렇지 않으면(연산자가 특정되면), 피연산자 값은 요청된 세그먼트의 인덱스(단계 1700에서 획득된 "reqSegIdx")에 적용하기 위한 오프셋 값(즉, "idxOffset"으로 명명됨)을 나타낸다. 그러한 경우에, 푸시될 세그먼트들은 연산자가 "+"라면 [reqSegIdx, reqSegIdx+idxOffset] 범위 내에, 연산자가 "-"라면 [regSegIdx-idxOffset, regSegIdx] 범위에 포함되는 인덱스들을 갖는 노드들 내에 있어야 한다. 연산자의 사용은 기술 파일 내에서 대응하는 제1 미디어 데이터의 미디어 데이터 속성 또는 속성들과 관련해서 제2 비요청 미디어 데이터의 미디어 데이터 속성 또는 속성들을 정의하는 것을 가능하게 한다.

[0413] 예를 들어, 하기의 푸시 정책들을 고려한다:

[0414] 1. <PushPolicy RepresentationIdx="-1" SegmentIdx="2"/>

[0415] 2. <PushPolicy PeriodIdx="+1" SegmentIdx="+2"/>

[0416] 3. <PushPolicy PeriodIdx="+0" SegmentIdx="+2"/>

[0417] PushPolicy #1은 서버가 요청된 세그먼트의 표현 노드에 선행하는 표현 노드에 인덱스 2의 세그먼트를 푸시할 것을 특정한다.

[0418] PushPolicy #2을 이용하여, 서버는 요청된 세그먼트에 후행하는 2개의 세그먼트들을, 현재 기간 또는 다음에 푸시할 것이다. 예를 들어, 도 24에서 세그먼트(2401)를 요청할 때, 세그먼트들(2405, 2402)이 푸시될 것이다.

[0419] PushPolicy #3은 PushPolicy #2와 매우 유사하고, 주요 차이는, 요청된 세그먼트가 기간의 끝에서 두 번째(penultimate)일 때이다. 예를 들어, 2401을 요청할 때, 현재 기간의 마지막 세그먼트(2405)(2개의 세그먼트들 대신에)만이 푸시될 것이다. PushPolicy #3을 이용하여, PeriodIdx는 세그먼트 검색을 요청된 세그먼트의 기간 노드에 제한하고, 이에 따라 기간의 마지막 세그먼트만이 푸시된다(요청된 세그먼트가 기간에서 끝에서 두 번째 세그먼트이기 때문이다). 반대로, PushPolicy #2를 이용하여, 세그먼트들은 다음 기간에서 검색될 수 있다.

[0420] 하나의 대안에서 또는 선택적인 값으로서, 피연산자의 값은 또한 '*'(와일드카드 의미)일 수 있고, 이것은 임의의 세그먼트가 푸시되어야 함을 의미한다. 그것이 연산자 '+'(별도의 "-")와 연관될 때, 그것은 요청된 세그먼트에 대한 모든 후속하는(별도의 선행하는) 세그먼트들이 푸시되어야 함을 의미한다.

[0421] 이 대안은 클라이언트가 하나의 기간의 모든 세그먼트들을 검색하기 위해, 예를 들어 다음의 PushPolicy: <PushPolicy PeriodIdx="+0" SegmentIdx="+*">을 이용하여 단일 HTTP 요청만을 전송하도록 허용한다.

[0422] 이들 예에서, 요청된 제1 미디어 데이터와 관련해서 (푸시될) 제2 미디어 데이터를 식별하기 위한 SegmentIdx 속성을 사용하려면, 제2 미디어 데이터가 제1 미디어 데이터에 인접할 것이 요구된다. 일 실시예에서, SegmentIdx 속성은 요청된 세그먼트의 인덱스에 적용할 오프셋(피연산자와 더불어)을 포함할 수 있다. 이것은 참조 세그먼트의 인덱스를 시프트하고, 이 참조 세그먼트로부터 특정된 양의 세그먼트들이 푸시되어야 한다. 예로서, SegmentIdx 속성의 구문은 다음과 같을 수 있다:

[0423] 푸시 속성:[연산자]피연산자[,오프셋]

[0424] 여기서 "오프셋"은 요청된 세그먼트 인덱스에 적용하기 위한, 0과는 상이한 양의 또는 음의 정수이다. 그러한 경우에, 검색 범위는 연산자가 '+'일 때 [reqSegIdx+offset, reqSegIdx+idxOffset+offset]이고, 연산자가 '-'일 때 [reqSegIdx-idxOffset +offset, reqSegIdx +offset]이다.

[0425] 푸시 정책의 구문은 또한, 푸시중인 표현 내의 최대 사이즈의 데이터 또는 시간과 같은 조건들(비제한적임)을 각각 포함할 수 있다. 예를 들어:

[0426] <PushPolicy SegmentIdx='+*[size<500000]'>은 500킬로 바이트 이하의 세그먼트 데이터를 푸시하기 위한 푸시

정책을 정의한다.

[0427] <PushPolicy SegmentIdx='+*[time<0:01:30]'>은 1분 30초 이하의 다음 세그먼트들 데이터를 푸시하기 위한 푸시 정책을 정의한다.

[0428] 상기의 예들은 어느 세그먼트들이 푸시되어야 하는지 결정하는 푸시 정책을 선언하는 방법을 나타내고 있지만, 세그먼트들이 푸시될 바람직한 순서도 특정할 필요가 있을 수 있다. 이 정보는 또한 클라이언트와 서버 사이에 공유되어야 한다.

[0429] 예로서, 도 7 내지 도 12를 참조하여 전술한 바와 같이 푸시되는 세그먼트들의 전송의 순서의 선언이 적용될 수 있다.

[0430] 푸시되는 세그먼트들의 전송의 순서에 대한 하나의 대안 실시예에서, 기술 파일 내의 기술 정보는 미디어 데이터와 연관된 우선순위 속성들, 각각의 미디어 데이터에 대한 하나의 우선순위 속성(예를 들어, "priorityIdx")을 포함하고, 제2 미디어 데이터의 전송 순서는 연관된 우선순위 속성들에 기초한다. 기술 파일의 전송의 덕분에, 클라이언트는 또한 이러한 우선순위 속성들에 의해 취해진 값들을 알고, 따라서 의도된 전송 순서를 결정할 수 있다.

[0431] 도 16의 예에 도시된 바와 같이, 매니페스트 파일에 기술된 각각의 세그먼트(예를 들어, 하나의 SegmentURL Node에 의해 식별됨)는 세그먼트의 푸시 순서를 특정하는 priorityIdx 속성(1604)을 포함한다. 도 16의 예에서, 세그먼트(1603)는 세그먼트(1602) 전에 푸시된다. 이러한 우선순위들은 서버 측에서 미디어 세그먼트들의 준비 동안 계산된다. 상이한 우선순위 값들이 이용될 수 있다: (도 16에서와 같이) 소정의 표현에서의 상대적인 우선순위 값, 또는 기간 우선순위에 대한 4개의 최상위 비트들, AdaptationSet 우선순위 값에 대한 4개의 다음 MSB, 표현 우선순위 값에 대한 다음 8비트, 및 세그먼트 우선순위에 대한 최하위 16비트를 갖는 32비트 수로서의 절대적인 우선순위 값. 절대적인 우선순위 값을 시그널링하는 대안 방법은 콤마로 분리된 우선순위 값들의 리스트, 위에서 인용된 레벨들 각각에 대한 하나, 예를 들어: 기간 우선순위, AdaptationSet 우선순위, 표현 우선순위, 및 그 다음으로 세그먼트 우선순위를 연속적으로 정의하기 위해 priorityIdx='1, 1, 2, 1'을 이용하는 것이다. 32비트 값을 가진 제1 실시예는 (이진수로) 다음과 같이 주어질 것이다:

[0432] priorityIdx='00010001000000100000000000000001'.

[0433] priorityIdx 값들을 이용하는 주요 이점은 상이한 표현(일반적으로 비디오의 교대 뷰와 같은 연관 표현)으로부터의 세그먼트들 간의 우선순위 순서를 정의하는 것을 가능하게 하는 것이다. 이것은 푸시 정책이 상이한 표현 집합들의 세그먼트들을 전송하는 것으로 이루어질 때 유용하다. 전형적 사용 예는 계층화된 비디오(계층은 멀티-뷰 내의 하나의 뷰이거나 또는 스케일러블 비디오 내의 확장성 계층임)의 스트리밍을 위한 것이거, 이 경우 하나의 계층으로부터의 세그먼트들이 하나 이상의 다른 계층을 가진 세그먼트들로 인터리브될 것이다.

[0434] 도 17을 다시 참조하면, MPD 파일에 정의된 바와 같은 푸시 정책에 기초하여, 서버는 단계 1701에서 푸시될 세그먼트들의 수를 결정한다. 이 수는 SegmentIdx 속성 값으로부터 직접 추론된다: 속성 값에 연산자가 사용되지 않는다면, 이 수는 1과 동일하다; 그렇지 않으면(연산자가 "-" 또는 "+"이라면), 이 수는 피연산자 값과 동일하고, 피연산자가 "*"일 때에는 무한대라고 추정된다(그러나 다른 제약 조건들 및 기존 세그먼트들의 수에 의해 제한된다).

[0435] 다음에, 단계들 1702 내지 1705로 구성되는 반복 프로세스는 푸시될 각각의 세그먼트들을 마킹하기 위해 푸시할 세그먼트들의 수에 도달될 때까지(테스트 1702) 스트리밍 서버에 의해 적용된다.

[0436] 각각의 반복에서, 서버는 단계 1703에서 PushPolicy 제약 조건들(적응 세트(Adaptation Set), 표현(Representation), 기간(Period), 및 세그먼트(Segment) 제약 조건들, 및 선택적인 조건들)을 준수하는 MPD 파일에 정의된 세그먼트들의 리스트를 검색한다.

[0437] 세그먼트의 리스트가 비어있거나 모든 그것의 세그먼트들이 이미 마킹되었다면(테스트 1704), 프로세스는 종료하고, 서버는 (상기의 단계 102에서) 클라이언트의 요청에 대한 응답을 전송하기 시작한다.

[0438] 그렇지 않다면, 리스트의 첫 번째 세그먼트는 단계들 103(PUSH_PROMISE) 및 104(약속된 세그먼트들) 동안 푸시될 것으로서 단계 1705에서 마킹된다.

[0439] 푸시 정책을 선언하는 이러한 MPD 기반 예들에서, 하나의 푸시 정책은 PushPolicy 요소를 이용하여 정의된다(도 16의 1600을 참조).

[0440] 기술 파일은 복수의 미디어 데이터 속성 레벨 즉, 위에서 정의된 기간, AdaptationSet, 및 표현 요소들을 이용하여 미디어 데이터를 기술한다.

[0441] 상기에 대한 약간의 변형으로서, 다양한 공유된 푸시 정책들이 기술 파일의 다양한 각각의 레벨들에서 정의될 수 있다. 이것은 미디어 스트림의 콘텐츠에 푸시 전략을 적응시키기 위해, 관련된 레벨(적응 세트, 표현, 기간)에 따라 다양한 푸시 정책들을 정의할 수 있다.

[0442] 이것은 도 23을 통해 예시되고, 여기서 푸시 정책은, 예를 들어 원하는 레벨에서, 여기서는 표현 레벨에서 "SupplementalProperty" 기술자를 이용하여 정의된다.

[0443] <MPD> 레벨별로 푸시 정책을 이용하면 미디어에 걸쳐 일정하고 동일한 푸시 정책을 갖는 것을 가능하게 한다.

[0444] <Period> 레벨별로 푸시 정책을 이용하면 시간에 따라 변화할 수 있는 푸시 정책을 갖는 것을 가능하게 한다.

[0445] <AdaptationSet> 레벨별로 푸시 정책을 이용하면 미디어-적응된 푸시 정책을 갖는 것을 가능하게 한다.

[0446] <Representation> 레벨별로 푸시 정책을 이용하면 미디어 특성(대역폭...)에 적응될 수 있는 푸시 정책을 갖는 것을 가능하게 한다.

[0447] 도 23의 예에서, 표현 레벨에서 특정된 푸시 정책은 푸시 데이터와 함께 너무 많은 대역폭을 이용하는 것을 회피하기 위해, 높은 비트레이트 비디오(2301)보다 낮은 비트레이트 비디오 세그먼트들(2300)에 대해 더 많은 세그먼트들을 푸시하도록 구성된다.

[0448] 푸시 속성들의 구문에 대한 전술한 설명들이 또한 이 약간의 변형에 적용될 수 있다는 것에 유의한다. 특히, 푸시 정책은 매니페스트에서 새로운 요소(도 16에서와 같이)로서, 또는 (도 23에서와 같이) 새로운 schemeIdUri를 가진 기존 기술자를 이용하여, 또는 새로운 기술자(표현 안됨)로서, 또는 MPD 스키마 또는 MPD 스키마 확장 포인트들에 따르는 임의의 수단으로서 시그널링될 수 있다.

[0449] MPD는 또한 대안의 푸시 정책들의 리스트를 포함할 수 있을 것이고, 그 각각은 고유 식별자를 갖는다(리스트에 대한 추가의 설명에 대해서는 하기를 참조한다).

[0450] 다른 대안 실시예들에서, 푸시 정책은 보충 표현들에 대한 세그먼트들이, 예를 들어 하기의 구문을 이용하여 체계적으로 푸시되는 것을 정의할 수 있다:

[0451] <push_policy Segments='+complementary'>

[0452] 또는 DASH 기술자를 이용할 때 value='complementary'.

[0453] 계층화된 비디오의 경우, 이것은 요청된 비디오 세그먼트에 대해, 보충 표현들로서 선언된 (전형적으로 상이한 표현들 사이의 종속성들을 시그널링하는 MPD 내의 dependencyId 속성을 통한) 모든 표현들로부터 동시에 각각의 세그먼트가 또한 푸시될 것임을 의미한다.

[0454] 다른 푸시 정책은 또한 @associationId 속성 또는 role='supplementary'를 가지고 시그널링되는 연관된 표현들로부터의 세그먼트들을 푸시하는 데 있을 수 있다.

[0455] 완전한 서버 주도 스트리밍의 경우에, 푸시 정책은 서버 거동이 '공격적'(또는 '낙천적') 또는 '보수적'이어야 하는지, 즉 더 높은 품질의 세그먼트들을 푸시하려고 하는지, 또는 동일한 품질 레벨(대역폭을 보존하는)을 푸시하려고 하는지에 대한 정보를 제공할 수 있다.

[0456] 다른 실시예들에서, 푸시 정책은 "푸시-정책" 헤더라고 지칭되는 전용 HTTP 헤더에서 전송된다. 그것은 공유되는 푸시 정책이 서버 디바이스로부터 클라이언트 디바이스에 전송되는 HTTP 프레임의 헤더에 삽입되는 것을 기술하는 푸시 정책 정보이다.

[0457] 이러한 실시예들은 시간의 경과에 따라 푸시 정책을 변경하는 것을 가능하게 하는데, 왜냐하면 그들은 더이상 상기와 같은 MPD 파일의 전송에 의존하지 않고, 클라이언트와 서버가 HTTP/2 프로토콜을 이용하여 교환하기 때문이다.

[0458] 도 18은 HTTP "푸시-정책" 헤더(헤더 명칭 "푸시-정책"은 단지 예임)에서 전송된 푸시 정책을 이용하는 서버와 클라이언트 간의 통신의 예이다.

[0459] 푸시-정책 헤더는 푸시 속성들의 리스트를 포함하며, 그 각각은 푸시될 데이터에 대한 제약 조건을 정의한다.

특히, 전술한 PushPolicy의 구문은 HTTP 헤더 구문으로 전사(transcribe)될 수 있다.

[0460] 도 18a에서, 서버는 클라이언트로부터의 MPD 요청에 응답하여(화살표 1800), 푸시 정책을 공유하기 위해, 전송된 MPD를 수반하는 HTTP 헤더에서 푸시-정책을 전송한다(단계 1801).

[0461] 예를 들어, 푸시 정책은 요청된 세그먼트에 후속하는 세그먼트가 푸시될 것임을 특정한다. 그 결과, 클라이언트가 세그먼트 Data1.1을 요청할 때(화살표 1802), 서버는 세그먼트 Data2.1에 대한 PUSH_PROMISE를 전송하고(화살표 1803), 그 후 세그먼트 Data1.1의 데이터를 전송한다(화살표 1804).

[0462] 후속 세그먼트 요청에 대해 어느 데이터가 전송될지를 정의하기 위해 임의의 구문이 이용될 수 있다: DOM 트리 노드 순회(tree node traversal)에 기초하는 MPD-특정의 하나 이상의 초록.

[0463] 동적으로 공유된 푸시 정책들에 전용인 특정 실시예에서, 클라이언트는 특정 푸시 정책을 요청할 수 있으며, 즉, 현재 공유된 푸시 정책이 그 요구에 대해 적합하지 않거나 또는 향상될 수 있다면, 공유된 푸시 정책을 업데이트할 수 있다.

[0464] 그것은 클라이언트 디바이스가 HTTP 프레임의 헤더에 삽입된 푸시 정책 업데이트 정보를 서버 디바이스에 전송한다는 것을 의미한다. 이에 따라, 서버 디바이스는 클라이언트 디바이스로부터의 HTTP 프레임의 헤더에 삽입된 푸시 정책 업데이트 정보를 수신한다. 따라서 서버 디바이스는 그에 따라 클라이언트 디바이스에 의해(예를 들어, 다음 요청을 위해) 요청된 다른 미디어 데이터로부터 비요청 미디어 데이터를 결정하기 전에 공유된 푸시 정책을 업데이트할 수 있다.

[0465] 일 실시예에서, 클라이언트로부터의 푸시 정책 요청은 HTTP 헤더 또는 "푸시-정책-요청"이라고 명명된(여기서 명칭은 단지 예임) 요청에서 전달된다.

[0466] 도 18b는 클라이언트가 새로운 푸시 정책을 요청할 때 클라이언트-서버의 예시적 교환들을 도시한다.

[0467] 교환들의 시작은 도 18a에서와 같다.

[0468] 세그먼트 Data2.1을 수신한 후, 예를 들어, 서버가 세그먼트 요청에 응답하여 더 많은 세그먼트들을 푸시하게 하기 위해 가용 대역폭이 충분히 안정적이기 때문에, 클라이언트는 현재 푸시 정책이 수정되어야 한다고 식별한다,

[0469] 그 결과, 클라이언트는 단계 1805에서 서버에게 각각의 새로운 요청에 대해 더 많은 세그먼트(1개 대신에 3개)를 푸시하도록 요청하는 푸시-정책-요청을 전송한다.

[0470] 서버는 단계 1806에서, 이 푸시 정책 요청에 OK(200)로 응답한다. 이 긍정적인 대답은 서버가 동일한 클라이언트로부터의 임의의 새로운 요청을 위해 푸시-정책-요청에 기술된 새로운 푸시-정책을 이용할 것임을 의미한다.

[0471] 서버가 그 푸시-정책을 변경하기를 원하지 않는다면, 서버는 푸시 정책 요청이 거절되었다고 클라이언트에게 통지하기 위해 에러 코드 응답을 리턴한다.

[0472] 다음에, 클라이언트가 단계 1807에서 다음 세그먼트 Data3.1을 요청할 때, 서버는 단계 1808에서 다음 3개의 세그먼트 Data 4.1, Data 5.1 및 Data 6.1에 대한 PUSH_PROMISE로 응답한다.

[0473] 도 21은 푸시 정책을 공유하기 위해 HTTP 요청을 이용할 때 서버 측에서의 프로세스의 단계들을 흐름도를 이용하여 도시하는 반면에, 도 22는 푸시 정책을 공유하기 위해 HTTP 요청을 사용할 때 클라이언트 측에서의 프로세스의 단계들을 흐름도를 이용하여 도시한다.

[0474] 도 14의 프로세스와 비교하여, 서버는 클라이언트로부터의 푸시 정책 요청을 다루기 위해 그리고 초기 푸시 정책 및 그 업데이트들을 전송하기 위해 새로운 처리 단계들(2100 내지 2105)을 포함한다.

[0475] 서버에 의해 수신되는 요청이 클라이언트로부터의 푸시 정책 요청이면(테스트 2100), 서버는 우선 클라이언트에 의해 제안된 데이터 푸시의 제약 조건들을 추출하기 위해 단계 2101에서 푸시 정책 요청을 파싱한다.

[0476] 이 단계 동안, 서버는 클라이언트에 의해 요청된 푸시 정책을 추종하도록 결정할 수 있다. 그러한 경우에, 서버는 그 내부 푸시 정책을 업데이트하고(단계 2102), 제안된 푸시 정책을 확인하기 위해, 단계 2103에서 OK(200) 응답을 클라이언트에 전송한다.

[0477] 그렇지 않으면, 서버가 푸시 정책을 폐기할 때 (예를 들어 제안된 정책이 리소스들의 관점에서 너무 비싸거나, 또는 적용될 수 없기 때문에), 단계 2102는 서버 측의 내부 푸시 정책을 수정하지 않고 에러 코드가 단계 2103

에서 클라이언트에 전송된다.

[0478] 특정 실시예에 따르면, 서버는 또한 클라이언트의 요청들과 독립적으로 그것의 푸시 정책을 업데이트할 수 있다. 그러한 경우에, 서버는 단계 1402 동안 푸시 정책을 결정하고, 그것의 특성들을 변경하기로(예를 들어, 클라이언트에 의해 수행되는 요청들 및 네트워크 특성을 분석함으로써) 결정할 수 있거나, 또는 결정된 푸시 정책이 현재의 것과 상이함을 알 수 있다. 그러한 상황에서, 서버는 새로운 푸시 정책을 클라이언트와 공유해야 하고, 후자가 이미 그것을 알지 않으면(테스트 2104), 그런 경우에 단계 2105에서 새로운 푸시 정책이 HTTP 헤더에서 전송된다.

[0479] 클라이언트 측에서의 대응하는 프로세스는 도 22를 참조하여 설명된다. 서버 처리에 있어서, 도 14의 프로세스와 비교해 새로운 처리 단계들(2200 내지 2204)이 푸시 정책 메시지들을 처리하고 푸시 정책 요청들을 수행하기 위해 추가된다.

[0480] 단계 1400에서 현재 공유된 푸시 정책(즉, 서버의 푸시 정책)을 결정한 후, 클라이언트는 예를 들어, 미디어 스트림의 세그먼트들을 검색하기 위해 전송할 HTTP 요청들의 수를 감소시키기 위해, 새로운 푸시 정책을 원할 수 있다. 그러므로, 새로운 푸시 정책이 클라이언트에 의해 요청될 때(테스트 2200), 클라이언트는 전술한 바와 같이 "푸시-정책-요청"를 가진 HTTP 요청을 단계 2201에서 전송한다.

[0481] 이 요청에 대한 응답이 단계 2204에서 처리되며, 이 단계에서 클라이언트는 서버가 OK(200) 응답을 리턴함으로써 그렇지 않으면 에러 코드를 리턴함으로써 요청을 확인하는지 체크한다.

[0482] 서버가 OK(200) 응답을 리턴하면, 단계 1400에서 결정된 현재 푸시 정책은 요청된 정책에 의해 대체된다. 그렇지 않다면, 그 정책은 변경되지 않는다.

[0483] 도 14의 프로세스 이외에, 클라이언트가 서버로부터 새로운 푸시 정책을 가진 프레임을 수신할 때(테스트 2202), 푸시 정책은 단계 1400의 다음 발생에서 검색되도록 하기 위해 파싱되어 메모리에 저장된다(단계 2203).

[0484] 다른 데이터(예를 들어, 미디어 데이터)도 포함하는 프레임에 푸시-정책 요청이 있을 때, 다른 데이터는 단계들 109-111-113-115를 통하여 처리된다는 것에 유의한다.

[0485] 상기의 HTTP-기반 예들은 적용될 푸시 정책을 완전히 정의하기 위해 HTTP 요청을 이용하지만, 하나의 특정 실시예는 클라이언트 측 및 서버 측 양쪽에 정의된 동일한 미리 정의된 푸시 정책들의 세트를 갖는 것에 의존할 수 있고, 그 각각은 고유 식별자를 가진다. 이 경우에, HTTP 요청은 세트 중에서 사용될 푸시 정책의 식별자를 특정하기 위해서만 이용된다. 이 특정 실시예는 HTTP 요청의 사이즈를 감소시킨다.

[0486] 일 실시예에서, 푸시 정책 요청은 서버 리소스 중 하나를 요청하기 위해 이용된 HTTP 요청들 중 하나의 추가 HEADER로서 전송된다: 일반적으로, 푸시 정책 요청은 MPD 파일에 대한 GET 요청 내의 "Accept-Push-Policy" HTTP 헤더에서 전송된다.

[0487] 다른 실시예에서, 클라이언트는 클라이언트에 의해 지원되는(또는 요구되는) 푸시 정책들의 리스트를 지시하기 위해 하나의 HTTP 요청에서 여러 "Accept-Push-Policy"를 특정한다. HTTP 요청에 응답하여, 서버는 제안된 리스트에서 푸시 정책 중 하나를 선택한 후 HTTP 응답에서 푸시 정책을 특정할 수 있거나, 또는 아무것도 지원되지 않는다면 새로운 푸시 정책에 의해 응답할 수 있다.

[0488] 또 다른 실시예에서, 푸시 정책 요청은 서버에 의해 알려진 임의의 리소스와는 독립적인 전용 HTTP 요청에서 전송된다. 예를 들어, GET(또는 POST) 요청은 웹 페이지, 예를 들어 http://server/push_policy의 리소스 중 어떤 것에도 대응하지 않는 URL로 형성되고, 또한 적어도 하나의 Accept-Push-Policy 헤더로 형성된다.

[0489] 또 다른 특정 실시예에서, 대안의 푸시 정책들의 세트는 서버와 클라이언트 간에 교환되는 MPD 파일에서 정의될 수 있고, 그 각각은 고유 식별자를 갖는다. 푸시 정책들 중 하나는 서버에 의해 선택되는 디폴트 푸시 정책으로서 마킹될 수 있다. 클라이언트는 디폴트 푸시 정책 대신에 사용될 푸시 정책의 식별자를 포함하는 새로운 푸시 정책 요청을 전송함으로써 어느 푸시 정책이 이용되어야 하는지 특정할 수 있다.

[0490] 일 실시예에서, 고속 시작을 위해 MPD 문서에 대한 요청 직후에 어느 세그먼트가 푸시될 것인지를 지시하기 위해 특정 푸시 정책이 정의된다.

[0491] 하이브리드 접근법에서, 공유된 푸시 정책을 기술하는 푸시 정책 정보는 제1 푸시 정책 부분과 제2 푸시 정책 부분에 의해 정의되고, 제1 푸시 정책 부분은 기술 파일(MPD)에 삽입되고, 제2 푸시 정책 부분은 서버 디바이스로부터 클라이언트 디바이스에 전송되는 HTTP 프레임의 헤더 내에 삽입된다.

[0492] 예를 들어, MPD는 푸시-정책 HTTP 요청 덕분에 이후 서버에 의해 정의되는(또는 심지어 오버로드되는) 템플릿 인수들로 푸시 정책을 정의할 수 있다. 예로서, MPD 파일에 정의된 푸시 정책은 <PushPolicy SegmentIdx="parameter"/>일 수 있고, 변수 "parameter"의 값은 푸시-정책 HTTP 요청에서 정의될 수 있다. 이 예에서, 제2 푸시 정책 부분은 제1 푸시 정책 부분에서 정의된 하나 이상의 연관된 변수에 대한 하나 이상의 값(만)을 포함한다.

[0493] 상술된 푸시-정책-식별자 기반 접근법을 이용할 때, 기술 파일은 복수의 후보 푸시 정책의 기술을 포함할 수 있으며, 제2 푸시 정책 부분은 따라서 상기 복수로부터 후보 푸시 정책의 식별자를 포함할 수 있고, 이에 따라 식별된 후보 푸시 정책은 제1 푸시 정책 부분을 형성한다.

[0494] 클라이언트에 푸시 정책을 선언하기 위한 또 다른 실시예에서, 푸시 정책은 푸시 데이터가 선택되는 표현이 어느 것인지를 나타내기 위해 MPD에 정의된 <Role> 기술자에 의지한다. 전형적으로, 푸시 정책은 푸시 전략이 "대체(alternate)" 또는 "보충(supplementary)" 역할 값을 가진 표현의 세그먼트를 이용할 것이라고 특정할 수 있다.

[0495] 또 다른 실시예에서, 리소스들의 문서, 예를 들어, 스트리밍 매니페스트 또는 HTML 페이지는 GET 요청이 수신된 후 푸시될 리소스들을 결정하기 위해 브라우징된 우선순위 트리로 변환된다. 우선순위 트리 내의 내비게이션은 XPath 요청으로 인해 수행될 수 있다. 이러한 접근법에서, 푸시 정책 정보는 제2 비-요청 미디어 데이터를 식별하기 위해 리소스들의 문서의 트리 표현에 대해 평가될 XPath 표현을 포함한다.

[0496] 예를 들어, 스트리밍 매니페스트에서, "following[name()="SegmentURL"][2]" XPath 표현식은, 푸시될 세그먼트들로서 선택하는데 사용될 수 있는데, 다음 2개의 세그먼트는 GET 요청에서 클라이언트에 의해 요청되는 세그먼트 뒤에 온다. 또한 챕터-스위칭 사용의 경우에, "((following[name()="Period"]//SegmentURL)[2])" XPath 표현식은 각각의 챕터의 2개의 제1 세그먼트를 사전 로딩하기 위한 다음 기간의 2개의 제1 세그먼트를 선택하는 것을 가능하게 한다. 예를 들어, 클라이언트가 도 24의 MPD 파일에서 세그먼트(2401)를 요청할 때, 다음 기간의 세그먼트들(2402 및 2403)은 또한, 푸시된 데이터로서 서버에 의해 전송된다.

[0497] 또한, 우선순위 트리는 예를 들어, 향상된 푸시 정책 규칙들을 위한 XPath 표현식 기록을 단순화하기 위해서 XSLT 명령을 이용하여 먼저 재순서화될 수 있다. XSLT 명령은 푸시 정책을 적용하기 전에 트리를 재구성하는 것을 가능하게 한다. XPath 표현식들은 바람직하게는 클라이언트에 전송되고, 예를 들어 하나의 HTTP 헤더와 XSLT 스타일시트는 웹 페이지에 정의된다. 이것은 특히, 예를 들어, 문서에서 선언된 모든 픽처들, DOM 트리의 동일한 레벨에서의 연속적인 노드들로서의 모든 CSS 리소스들을 그룹화하기 위해, HTML 문서들에 적용한다.

[0498] 예를 들어, 도 25의 트리(2501)는 상이한 타입들의 상이한 리소스를 가진 HTML 페이지를 나타낸다: 단색(2521-2524)의 이미지 리소스들 및 노드들에 대응하는 해시 노드들(2511-2514)은 스크립트된 리소스들이다(CSS 또는 자바스크립트). 트리(2502)는 타입별로 리소스들을 그룹화(2530의 이미지들과 2540의 스크립트된 리소스들)하기 위한 XSLT 변환 결과의 예이다. 따라서 단순한 XPath 표현식은, 주어진 타입의 제1 리소스가 요청될 때 이러한 주어진 타입에 대한 일부 리소스가 푸시될 것을 나타내기 위해 정의될 수 있다.

[0499] 상술된 모든 실시예들에서, 각각의 클라이언트 요청에 대해 서버는 푸시 정책이 여러 세그먼트들에게 푸시되도록 요청하는 경우 여러 PUSH PROMISE로 응답할 가능성이 매우 크다.

[0500] 예를 들어, 도 19의 MPD(1900)는 요청된 세그먼트에 후행하는 3개의 세그먼트가 푸시될 것을 나타내는 푸시 정책을 갖는다(<PushPolicy> 요소를 참조). 따라서, 클라이언트가 0-999와 동일한 바이트 범위로 미디어(1901)에 대한 GET 요청을 가진 초기화 세그먼트를 요청하면, 서버는 단계 103 동안 3개의 PUSH_PROMISE 메시지(1902)을 전송할 것이다.

[0501] 한 실시예에서, 식별된 제2 미디어 데이터가 알림 메시지(즉, PUSH_PROMISE)를 각각 요청하는 복수의 미디어 세그먼트를 포함하면, 대응하는 복수의 알림 메시지는 클라이언트 디바이스로 전송될 단일 알림 메시지로 통합될 수 있다.

[0502] 도 20에 도시된 바와 같이, 이 상황을 달성하기 위해, 서버에서의 처리는 바람직하게 도 14의 일반적 프로세스와 비교하여, 단계 103에서 푸시 약속들을 전송하기 직전에 사전-처리 단계 2000을 포함한다. 사전-처리 단계는 알림 메시지들의 상기 통합을 수행하려고 한다.

[0503] 푸시 약속들이 1902에서와 같이, 바이트 범위 요청들을 포함하면, 푸시 약속들(1902)의 리스트는 연속적인 바이트 범위 어드레스들을 포함하는 감소된 세트의 푸시 약속들(1903)을 생성하기 위해 브라우징된다. 다음에, 푸

시 약속들(1902)의 각각의 세트는, 푸시 약속들 세트에서 바이트 범위들의 연쇄와 동일한 연속적인 바이트 범위를 가진 감소된 세트의 푸시 약속들(1903)에 의해, 또는 비연속적인 바이트 범위들의 리스트를 가진 단일 푸시 약속들(예를 들어, 1905)에 의해 대체된다.

[0504] 예를 들어, 3개의 푸시 약속(1902)은 도 19에 도시된 단일 푸시 약속(1903)에 의해 대체된다.

[0505] 푸시 약속들을 통합하는 이러한 접근법은 더 단순한 방식으로 그리고 더 낮은 대역폭과 처리 비용들로 클라이언트가 푸시 데이터의 전송을 취소하는 것을 가능하게 한다. 이것은 클라이언트가 비통합 푸시 약속들 각각에 대한 여러 스트림들을 닫기보다는 단일 푸시 약속에 대한 단일 스트림을 닫아야만 하기 때문이다.

[0506] 대안으로, 푸시 약속들이 분리된 바이트 범위 구간들을 갖고 있을지라도, 모든 푸시 약속들은 (연속적인 바이트 범위 구간들이 연결되는 경우에) 바이트 범위들의 리스트에 의해 대체될 수 있다.

[0507] 또한, 푸시 약속들이 바이트 범위 구간들을 포함하지 않고 오히려 상이한 SegmentURL 값들을 포함한다면, 푸시 약속들은 다음과 같이 단일 푸시 약속 메시지를 생성하기 위해 또한 연결될 수 있다: 생성된 푸시 약속 메시지의 방법은 (다수의 GET을 위해) MGET로서 정의되고, 경로 필드는 1904에서 표현된 바와 같은 세그먼트 URL들의 리스트이다. 이전 실시예와 유사하게, 클라이언트는 모든 세그먼트들의 푸시를 취소시키기 위해 생성된 푸시 약속에 해당하는 단일 스트림을 닫아야 한다.

[0508] 클라이언트가 각각의 푸시된 세그먼트를 파싱하고 식별할 수 있다는 것을 보장하기 위해, 서버가 그때 전송된 데이터에서 각각의 세그먼트의 끝에서 END_SEGMENT 플래그들을 포함할 수 있다는 것에 유의한다.

[0509] 또한, HTTP/2의 SETTINGS 프레임은, 푸시 약속들의 그룹화가 스트리밍 세션에 대해 허용되는지를 나타낼 수 있게 하는 새로운 SETTINGS_ENABLE_GROUP_PUSH_PROMISE 파라미터를 포함하도록 확장된다.

[0510] 본 발명의 실시예들은 하나 또는 여러 라운드트립(들)이 회피될 수 있기 때문에 DASH 고속 시작을 갖는 것이 가능할 수 있다. 본 발명의 이러한 양태는 이하 도 26 내지 28을 참고하여 설명된다.

[0511] DASH 고속 시작 특징은 도 7 내지 12 및 14 내지 25의 모두 또는 일부를 참고하여 상술한 바와 같은 임의의 통신 접근법과 함께 사용될 수 있다.

[0512] 도 26은 DASH 고속 시작을 획득하기 위해서, 본 발명의 교시들에 따라서 서버에 의해 그리고 클라이언트 디바이스에 의해 각각 구현되는 예시적인 방법들을 보여준다.

[0513] 방금 설명된 표준 프로세스에서와 같이, 제1 단계는 기술 파일, 여기에서는 MPD 파일을 클라이언트가 요청하는 것으로 구성된다(단계 2650). 클라이언트는 그 후 서버의 응답을 대기한다(단계 2651).

[0514] 한편, 서버는 특히 아래 설명된 것처럼, 클라이언트가 더 빨리 시작하는데 도움이 될 초기화 데이터를 식별하기 위해(단계 2653), MPD 파일을 파싱한다(단계 2652). 단계 2653에 대한 예시적 실시예는 도 27을 참고하여 아래 설명된다.

[0515] 초기화 데이터가 서버에 의해 식별되면, 서버는 클라이언트의 요청을 대기하지 않고 초기화 데이터를 푸시하기 위한 그 의도를 나타내기 위해 단계 2654에서 PUSH_PROMISE 프레임을 클라이언트에 전송한다.

[0516] 아마도, 또한 클라이언트가 관련된 리소스, 즉 :scheme, :host 및 :path와 같은 관련된 초기 미디어 데이터를 식별할 수 있게 하는 헤더 필드들을 포함하는 다른 PUSH_PROMISE 프레임을 전송함으로써 초기 미디어 데이터를 푸시할 것이라고(단계 2656) 또한 시그널링한다.

[0517] 초기화 데이터를 위한 PUSH_PROMISE 프레임과 초기 미디어 데이터를 위한 PUSH_PROMISE 프레임의 양쪽의 경우에, 다른 헤더 필드들은 또한 서버가 그것이 푸시하기로 결정한 데이터에서 얼마나 확신하는지를 나타내기 위해 서버에 의해 추가된다: 본 발명의 실시예에서는, confidence_level 파라미터는 PUSH_PROMISE 프레임과 연관된다(즉, 헤더에 포함된다). confidence_level 파라미터의 결정은 도 27을 참고하여 아래 설명된다. 서버는 또한 그것이 푸시하려고 하는 세그먼트를 명백히 나타내기 위해 특정 DASH 헤더를 삽입할 수 있다.

[0518] 클라이언트가 푸시될 초기화 데이터와 제1 미디어 데이터에 대해 요청을 할 위험을 최소화하기 위해, PUSH_PROMISE 프레임들은 응답에서의 임의의 콘텐츠 이전에 전송되어야 하고, 즉 단계 2654와 단계 2656은 서버에서부터 클라이언트 디바이스까지 MPD 파일을 전송하는 단계 2655 이전에 발생하여야 한다.

[0519] 그러므로, PUSH_PROMISE 프레임들이 클라이언트 디바이스에 전송될 때, 서버는 단계 2655에서 MPD 파일을 클라이언트 디바이스에 전송한다.

[0520] 서버가 그 동안에 클라이언트 디바이스로부터 임의의 CANCEL 또는 ERROR 메시지를 수신하지 않았다면, 그것은 초기화 데이터를 푸싱하고(단계 2657) 제1 미디어 데이터를 푸싱(단계 2658)하기 시작한다.

[0521] PUSH_PROMISE 프레임들과, 서버에서 클라이언트 디바이스로의 데이터의 푸싱은, 예를 들어 문서 "Hypertext Transfer Protocol version 2.0, draft-ietf-httpbis-http2-latest", HTTPbis Working Group, Internet-Draft, 2013년 6월 24일(예를 들어 http://http2.github.io/http2-spec/에서 이용 가능한)에서 설명된 바와 같이, 예를 들어 HTTP 2.0의 프레임에서 개발되는 대응하는 특징들에 따라서 수행된다.

[0522] 클라이언트 디바이스에서의 수신시, 초기화 데이터는 디코더(들)를 셋업하기 위해 클라이언트에 의해 이용될 수 있고(단계 2659), 제1 미디어 데이터는 데이터의 충분한 양이 화면 동결 없이 디코딩과 렌더링(예를 들어, 표시)에 이용 가능할 때까지 버퍼링된다(단계 2660).

[0523] 클라이언트가 완전히 MPD 파일을 수신했을 때, 클라이언트는 이를 파싱하고(단계 2662), 데이터가 충분히 버퍼링되었다면(단계 2661) 디코딩 및 표시를 시작한다(단계 2663). 이것은 그 경우가 아니고, 클라이언트 디바이스가 서버에 의해 전송된 PUSH_PROMISE 프레임들로부터 더 많은 세그먼트들이 전송될 것임을 알게 되면(단계 2656 참조), 그것은 단계 2664에서 서버로부터 제1 미디어 데이터의 푸시의 완료를 대기한다. 이러한 유휴 단계 2664 동안, 상기에 이미 설명한 것처럼, 클라이언트 디바이스는 표준 클라이언트 제어 DASH에서 나올 후속 세그먼트들에 대한 다음 요청들을 준비할 수 있다(단계 2665). 이것은 상응하는 PUSH_PROMISE 프레임에서 클라이언트 디바이스가 푸시될(또는 푸시되고 있는) 초기 미디어 데이터에 대한 정보를 수신했고(상기 단계 2656 참조), 따라서 서버에 의해 푸시되도록 의도된 최종 시간 세그먼트 바로 뒤의 시간 세그먼트에 대한 요청들을 준비할 수 있기 때문에 가능하다.

[0524] 클라이언트 디바이스는, MPD를 완전히 수신했을 때, (푸시된 초기 미디어 데이터가 버퍼를 충전하는 경우를 보여주는 도 26에 도시된 것과 대조적으로) 단계 2661 이전에 표준 클라이언트 제어 DASH 프로세스에 따라서 이러한 초기 미디어 데이터가 버퍼를 충전하는지를 체크하고, 그렇지 않다면, 후속하는 미디어 데이터(예를 들면, 초기 미디어 데이터에 의해 표현되는 시간 세그먼트에 후행하는 시간 세그먼트에 해당하는 미디어 데이터)에 대한 요청을 전송하기 위해, 단계 2656에서 수신된 초기 미디어 데이터에 대한 정보를 이용할 수도 있다. 이것은 클라이언트가 푸시할 제1 미디어 데이터의 양에 대한 서버로부터의 나쁜 추정을 보정할 수 있게 한다.

[0525] 이러한 프로세스는 표준 매니페스트 기반 스트리밍에서보다 더 빨리 스트리밍 클라이언트가 미디어의 표시를 시작할 수 있게 한다. 사실상, 네트워크에 대한 HTTP 라운드트립들의 수가 초기화 데이터 및/또는 초기 미디어 데이터를 얻기 위해 감소되기 때문에 시동 지연은 감소된다.

[0526] 이러한 프로세스는 그러나 현재의 DASH 표준을 아직도 준수하는데, 그 이유는 다음과 같다:

[0527] - MPD 파일의 수정이 없다: 그 전송은 가볍고 빠르게 유지된다;

[0528] - 표준 DASH 클라이언트들의 거동(즉, 본 발명의 교시들로부터 혜택이 없는)이 변하지 않을 수 있다: 그와 같은 클라이언트 디바이스들은 인식되지 않은 HTTP 헤더들을 무시할 것이고, 푸시 특징을 수용하지 않을 때는, 간단하게 더 많은 요청/응답을 수행하여야 하고 따라서 표현을 시작하기 위해 더 많은 시간을 소비한다.

[0529] 도 27은 클라이언트 디바이스로부터 매니페스트(또는 기술 파일)에 대한 요청에 후행하는 서버 측에서 구현된 예시적인 방법을 설명한다.

[0530] 이러한 방법은 클라이언트가 미디어 표현의 표시를 빨리 시작할 수 있도록 미리 푸시하기 위한 가장 적절한 초기 데이터를 식별하려고 한다.

[0531] 단계 2700에서, 매니페스트에 대한 요청이 수신된다. 서버는 그 후 단계 2701에서 클라이언트 디바이스가 일부 선호들을 요청에 삽입했는지를 체크한다. 이것은, 예를 들어 미디어 표현을 위한 전송 레이트와 오디오 스트림을 위한 바람직한 언어를 표현하기 위해 전용 HTTP 헤더를 통해 행해질 수 있다:

[0532] GET http://myserver.com/presentation/pres1.mpd\r\n

[0533] Prefered-MediaRange:bw=2000;lang=FR\r\n\r\n

[0534] 요청이 선호들을 포함하면(테스트 2701에서 참), 서버는 클라이언트의 선호들을 분석하고(단계 2703), 그 confidence_level 파라미터를 값 "high"로 설정한다(단계 2704).

[0535] 표시가 요청에서 제공되지 않으면(테스트 2701에서 거짓), 서버는 단계 2702에서 그것이 이미 이러한 클라이언

트를 위해 서비스 사용 정보(로그들)(즉, 사용자 또는 클라이언트 디바이스와 서버 사이의 이전 교환들을 기초로 한 통계치들 또는 사용 데이터) 또는 User-Agent 헤더로부터의 정보를 등록했는지를 체크한다. 사실상, User-Agent 헤더는 RFC2616에서 HTTP 헤더로서 정의되고(예를 들어, http://www.ietf.org/rfc/rfc2616.txt 참조), 예를 들어, 운영 체계, 브라우저 타입, 애플리케이션 명칭 등과 같은 정보를 교환하기 위한 애플리케이션들용 수단을 제공한다. 예를 들어, DASH 서버는 서비스를 사용할 수 있기 전에 클라이언트들을 위한 인증 방식을 가질 수 있고; 변형에서, 그것은 서비스에 대한 액세스를 얻기 전의 사용자 로깅일 수 있다. 그와 같은 수단에 의해, 서버는 미디어 파라미터들을 연결된 사용자 또는 디바이스에 링크시킬 수 있다.

[0536] 이전의 사용 정보(로그들)가 단계 2705에서 로그들을 분석함으로써 관련된 클라이언트 디바이스 또는 사용자에 이용 가능하면(테스트 2702에서 참), 서버는 주어진 클라이언트 또는 사용자에 대해 가장 빈번한 사용들을 추론할 수 있다. 예를 들어, 그것은 사용자 또는 클라이언트 디바이스가 불어로 된 오디오 스트림과 HD(고해상도)의 비디오 스트림을 항상 선택한다고 추론할 수 있다. 더욱이, 서버는 이것이 오픈 TCP 연결에서 제1 요청인지 아닌지를 알 수 있다(클라이언트가 서비스에 연결되어 제2 미디어 표현을 요청하는). 이 경우에, 대역폭 추정은 더 정확하고 믿을 만하고 TCP 혼잡 윈도우는 제1 요청을 위한 것보다 더 클지도 모른다. 이것은 적절한 표현의 관점에서 서버에 의해 만들어진 선택에 영향을 줄 수 있다.

[0537] DASH 품질 메트릭을 등록함으로써, 서버는 그 로그들에서 사용자/클라이언트가 보통 수행하는 다양한 표현들 중에서 변화들을 가질 수 있다. 이것으로부터, 서버는 변화들의 빈도에 의존하는 "어그레시브(aggressive)" 또는 변함없는 거동 사이의 통상의 거동을 결정한다(변화들에 의해, 그 기준: 대역폭, 해상도, 프레임 레이트 등이 무엇이든지 간에 다른 표현으로의 스위칭을 의미한다). 어그레시브 클라이언트는 그 상황이 변할 때 자동으로 상이한 표현으로 스위칭될 DASH 클라이언트이다. 예로서, 대역폭 또는 버퍼 점유를 모니터링할 때, 어그레시브 클라이언트는 새로운 표현이 현재 표현과 비교하여 특성들을 클라이언트의 상황에 더 가까운 특징들을 갖자마자 상이한 대역폭을 가진 표현을 요청할 것이다. 반대로, 변함없는 클라이언트는 안정적 품질 및 표시 레이트를 유지하기 위해 빈번한 표현 스위치들을 회피하려고 할 것이다. 사용자/클라이언트 디바이스 거동이 적응의 관점에서 보다 어그레시브할 때, 서버는 이후 그것이 스트리밍을 시작하기 위해 초기 표현으로서 선택하는 것이 무엇이든지, 클라이언트가 스트리밍의 후행하는 처음의 수초(seconds) 또는 수분(minutes)에서 적응하려고 하는 것을 안다.

[0538] 선호들이 로그들에서 추론될 때, 서버는 단계 2706에서 그것의 confidence_level 파라미터를 값 "mid"로 설정한다. 사실상, 이러한 정보는 클라이언트 자체에 의해 시그널링하는 명시적 선호들보다 약간 덜 관련될지도 모른다(테스트 2701에서 참).

[0539] 로그 정보가 이용 가능하지 않으면(테스트 2702에서 거짓), 서버는 그것의 confidence_level 파라미터를 단계 2707에서 최하위 값, "low"에 둔다. 이것은 서버가 그것이 푸시한 정보에 대한 최상의 추측을 수행한 것을 나타내는데, 그 이유는 결정할 사전 정보가 없기 때문이다. 이 경우에 추가 프로세스는 아래 설명된다(단계 2711 참조).

[0540] 이러한 confidence_level 파라미터 계산과 동시에, 서버는 단계 2708에서 매니페스트를 파싱할 수 있다. 매니페스트가 아주 자주 변하지 않을 것 같은 경우(라이브 서비스와 반대로, 특히 주문형 서비스의 경우), 매니페스트의 파싱은, 다양한 표현들의 기술을 룩업 테이블에 등록함으로써, 오프라인에서 전부에 대해서 한번 수행될 수 있다. 이러한 룩업 테이블은 또한 미디어 표현의 일부들에 클라이언트들의 로그들을 링크시키기 위해 서버에 의해 사용될 수 있다. 이것은 더 빠른 로그 처리(상술된 단계 2705 참조)가 일부 클라이언트의 선호들을 추론할 수 있게 한다.

[0541] 매니페스트의 파싱(단계 2708)은 스트리밍을 시작하기 위해 초기 표현(즉, 초기 미디어 데이터)으로서 적절한 표현의 선택(단계 2709) 시에 서버에 정보를 제공한다.

[0542] 단계들 2703 및 2705(요청에 있으며 또는 이전 교환들로부터의 사용 데이터를 기초로 하는 선호들을 각각 획득하는) 모두는 클라이언트 디바이스/사용자로부터의 선호들 또는 사용들을 MPD 속성들과 일치하는 구체적인 파라미터들로 번역하는 단계로 이루어진다. 예를 들어, 그것은 대역폭, 비디오의 폭과 높이, 사용시의 코덱 종류, 자막들 또는 오디오 스트림들을 위한 언어일 수 있다. 그 후, 이러한 파라미터들에 대한 획득한 값들로부터, 서버는 클라이언트에게 푸시할 가장 편리한 표현을 단계 2709에서 식별하기 위해 매니페스트에서의 값들과 비교한다.

[0543] 이러한 단계 2709가 전형적으로, 클라이언트 디바이스가 DASH와 같은 동적 및 적응적 스트리밍 프로토콜에서 연

속적으로 수행하는 것임을 유의할 필요가 있다. 여기서, 동일한 단계는 MPD 파싱 수단에 의한 스트리밍 세션의 시작에서 서버에 의해 수행된다.

[0544] 적절한 표현이 2709에서 추론될 수 없는 경우에, 테스트 2710은 거짓이고, 서버는 (이전에 언급된 단계 2707에서) 그것의 confidence_level 파라미터를 "low" 값에 둔다.

[0545] confidence_value 파라미터가 "low" 값을 갖고 있을 때(선호들이 결정될 수 없거나 또는 적절한 표현이 선호들에 기초하여 발견될 수 없기 때문에), 서버는 단계 2711에서 가장 단순한 표현을 선택하도록 결정한다. 비디오에서, 예를 들면, 가장 단순한 표현은 가장 낮은 공간 해상도를 갖고 가장 낮은 대역폭을 위해 설계된 표현일 수 있다.

[0546] 가능한 상보적 특징(도 27에 도시되지 않음)에 따르면, 코덱에 대한 불확실성이 없을 때(즉, 모든 비디오 표현들이 코덱 속성에 대해 동일한 값을 갖고, 즉 동일한 코덱으로서, 예를 들어 HEVC가 모든 비디오 표현들을 인코딩하는데 사용될 때), confidence_level 파라미터는 값 "mid"까지 높아질 수 있다.

[0547] 단계 2711 후, 또는 적절한 표현이 발견되었을 때(테스트 2710에서 참)의 다음 단계는 초기화 데이터를 식별하는 단계로 이루어진다(2712 단계). 사실상, DASH 매니페스트(또는 기술 파일)에서, 초기화 정보는 다른 방식으로 시그널링될 수 있다: 초기화 정보는 초기화 데이터에 다이렉트 URL을 제공하는 SegmentBase, SegmentList 또는 SegmentTemplate 요소의 초기화 요소에 명시적으로 둘 수 있다.

[0548] 이 경우에, 이러한 URL은 (변수들 :scheme, :host 및 :path 및 결국에는 :Range를 특정함으로써) 클라이언트가 푸시될 것으로 약속된 리소스를 식별할 수 있게 하는 PUSH_PROMISE 프레임의 헤더 필드에 둔다(도 26을 참고하여 상술된 단계 2654 참조).

[0549] 초기화 데이터가 명시적으로 기술되지 않을 때, 이는 미디어 세그먼트들이 자체 초기화된다는 것을 의미한다. 그와 같은 경우에, 서버는 세그먼트(예를 들어, mp4 포맷에서 세그먼트들을 위한 세그먼트 인덱스 정보 박스들)의 시작을 파싱하여야 한다. 이러한 분석을 기반으로, 그것은 PUSH_PROMISE 프레임에 헤더로서 둘 적정 바이트 범위를 가진 상응하는 URL을 구축할 수 있다.

[0550] 일단 식별되면, 초기화 데이터를 위한 PUSH_PROMISE 프레임은 즉시 클라이언트에 전송되고(도 26에서 단계 2654에 해당하는 단계 2713), 초기화 데이터의 푸시에 의해 여기에 바로 후행한다(도 26에서 단계 2657에 해당하는 단계 2717a). 초기화 데이터가 수신되면, 클라이언트는 그 후 그것의 미디어 디코더들을 초기화할 수 있다 (단계 2717b).

[0551] 선택적으로, PUSH_PROMISE 프레임을 처리할 때 클라이언트 디바이스에 의한 세그먼트 시그널링과 이후의 식별을 향상시키기 위해(아래 설명된 단계 2806 참조), 서버는 단계 2713에서 다음을 나타낼 수 있다: 푸시된 데이터의 속성: 초기화 또는 미디어 또는 양쪽(자체 초기화 세그먼트들의 경우); 도 5b의 MPD 표현 트리에서의 경로로서의 URL 템플릿의 파라미터들 또는 세그먼트의 표시(예를 들어: P2AS21R211S1; 즉, 식별자에 후행하는 요소 타입의 연결). 이것은 클라이언트 디바이스가 수신된 MPD를 갖도록 요청하는 것에 유의할 필요가 있다. 그 후, 서버는 클라이언트 디바이스에 의한 MPD 수신 후에 처리될 것이라고 생각하는 PUSH_PROMISE 메시지들에만 이러한 특정한 정보를 추가하기로 결정할 수 있다. MPD 수신과 파싱 이전에 PUSH_PROMISE를 수용할지에 대한 클라이언트 디바이스에서의 결정을 돕기 위해, 서버는 그것이 기저 계층 또는 향상 계층으로부터의 세그먼트인지의 여부와 같은, MPD 내의 세그먼트 경로 대신에, 푸시된 세그먼트에 대한 정성(qualitative) 정보를 나타낼 수 있고; 또 다른 예에 따르면, 서버는 그들의 값들을 가진 선택된 표현의 속성들을 헤더에 위치시킬 수 있다.

[0552] 가능한 실시예(도 27에 표시되지 않음)에 따르면, 단계 2708에서의 매니페스트의 파싱은 초기화 데이터가 매니페스트의 최상위 레벨 요소들에 존재한다고 결정할 때(즉, 어떠한 표현들이든, 초기화 데이터는 모든 표현들에 공통이고; 예를 들어, 종속적인 표현의 경우에), 서버는 즉시(즉, 단계 2708과 동시에), 푸시된 데이터와 클라이언트가 선택한 것 사이의 불일치의 위험이 없기 때문에 값 "high"에 설정된 confidence_level 파라미터를 가진 초기화 데이터를 지정하는 PUSH_PROMISE 프레임을 전송할 수 있다. 예를 들어, HTTP 헤더로서 PUSH_PROMISE 프레임을 가진 confidence_level 파라미터를 전송하는 이점은 그것이 푸시 약속을 수용하거나 취소시킬 시에 클라이언트 디바이스를 도울 수 있다는 것이다(하기 도 28의 설명 참조).

[0553] 이 특징의 덕분에, 클라이언트는 (PUSH_PROMISE 프레임이 초기에 전송된 바와 같이) 그것의 디코더들을 셋업하는데 요구된 초기화 데이터를 오히려 더 빨리 수신할 것이다. 이것은 또한 초기화 데이터가 주어진 미디어 타입(예를 들어, AdaptationSet에서 표현들의 수와 상관없이 AdaptationSet당 오직 하나의 InitializationSegment)에 대해 고유할 때 작동한다. 이러한 보다 더 빠른 푸시는 매니페스트의 파싱 직후(상

술된 단계 2708)에, 이에 따라 로그들 또는 선호들을 처리하기 전(상술된 단계들 2701, 2703 및 2705)에 오게 될 것이다.

[0554] 그 후, 이전에 서버에 의해 결정된 confidence_level 파라미터가 "mid" 값보다 크거나 같으면(테스트 2714), 서버는 그것이 클라이언트에 적합한 것으로서 고려된 제1 미디어 데이터를 푸시하는 것을 능동적으로 취한다.

[0555] 이것은 2개의 단계에서 반복적으로 행해진다: 먼저 PUSH_PROMISE 프레임이 전송되고(도 26에서 단계 2656에 해당하는 단계 2715), 그 후 제1 미디어 데이터의 푸시는 단계 2719에서 시작한다. 이것은 단계 2709에서 푸시되도록 선택된 각각의 제1 미디어 데이터 세그먼트에 대해 반복된다.

[0556] 가능한 실시예에 따르면, 연속적인 미디어 세그먼트들이 푸시되도록 약속될 때(즉, 복수의 PUSH_PROMISE가 각각의 미디어 세그먼트들에 전송될 때), 현재 미디어 세그먼트와 연관된 PUSH_PROMISE는 이전 PUSH_PROMISE의 차일드 또는 팔로워(follower)로서 마킹된다(단계 2716). 이것은 서버가 비상태 유지(stateless)되는 경우 PUSH_PROMISE 프레임에 새로운 HTTP 헤더로서 둘 수 있거나 또는 서버가 상태 유지(stateful)인 경우 테이블에 유지될 수 있다. 이러한 관계성을 유지하는 것은 푸시 약속들에 대한 계층 취소를 수행하는데 유용할 수 있다(도 28을 참고하여 아래 설명된 바와 같이).

[0557] 데이터의 다양한 전송들의 가능한 스케줄은 다음과 같다: 실제로 제1 미디어 데이터를 푸시하기 전에, 서버는 상기에 언급된 단계 2717a에서 초기화 데이터를 푸시하기 시작하고; 제1 미디어 데이터와 초기화 데이터와 관련된 PUSH_PROMISE 프레임을 전송하는 것과 동시에, 서버는 또한 단계 2718에서 MPD 파일(매니페스트)를 전송하고, 푸시된 데이터가 완전히 전송될 때까지 스트림을 오픈 상태로 유지한다.

[0558] 또 다른 실시예에서, 테스트 2714는 신뢰의 레벨과 관계없이 제1 미디어 데이터를 푸시하기 위해 회피될 수 있다. 그러나 confidence_level 파라미터가 "low"로 설정되는 경우에, 서버는 실제로 제1(또는 초기) 미디어 데이터를 푸시하기 전에 클라이언트로부터 잠재적 CANCEL를 대기할 수 있다.

[0559] 제1 미디어 데이터를 푸시할 때, 서버는 (흐름 제어)를 이용하기 위해 푸시하기 위한 데이터의 전체 수량과 속도를 결정한다.

[0560] 제1 양태와 관련하여, 서버는, 예를 들어 매니페스트의 시작에 언급된 minBufferTime 속성과 같은 매니페스트로부터의 정보를 활용할 수 있다. 이러한 속성을 이용하고, 단계 2709 또는 2711에서 선택된 표현을 고려하고, 매니페스트에도 제공되는 세그먼트 지속시간 속성이 주어지면, 서버는 minBufferTime 제약 조건(즉, 세그먼트들의 수량, 따라서 푸시될 초기 미디어 데이터를 형성하는 데이터의 수량)을 충족시키도록 푸시하기 위해 세그먼트들의 수를 쉽게 결정한다. 이롭게, 매니페스트의 파싱(단계 2708)이 오프라인으로 수행될 때, 이러한 제1 미디어 세그먼트들의 수는 서버의 메모리의 테이블에 기록될 수 있다.

[0561] 제2 양태와 관련하여, 세그먼트의 지속시간과 선택된 표현의 대역폭이 주어지면, 필요한 비트레이트의 추정치는 서버에 의해 획득될 수 있다. 이것은, 주로 비디오 세그먼트들, 이용하기 위한 전송 레이트를 위해 제공한다. 예를 들어, 5초의 지속시간의 세그먼트들을 갖고 있는 1.6 M비트/s와 동일한 대역폭을 가진 압축된 비디오 표현을 위해, 각각의 세그먼트는 전송할 데이터의 1 메가바이트를 나타낼 것이다. 디폴트에 의해, HTTP v2.0의 흐름 제어는 기껏해야 65535바이트들과 동일한 스트림 윈도우 사이즈를 제공한다. 그러므로, 본 예에서, 이것은 클라이언트가 65536의 푸시된 바이트들, 그래서 본 예에서는 세그먼트당 15배보다 큰, 각각의 패킷을 위한 확인응답을 서버로 다시 전송해야 할 것이라는 것을 의미한다. 본 발명이 개선 HTTP 2.0 하에서 푸시 특징을 이용할 때 네트워크 라운드트립들과 트래픽을 감소시키는 것을 목표로 하기 때문에, 본 발명은 (네트워크 트래픽을 감소시키는 것에 의해) DASH 고속 시작을 가능하게 하기 위해 디폴트 거동(실제로 디폴트 혼잡 윈도우 사이즈)을 수정할 필요가 있음을 명확히 알고 있다.

[0562] 클라이언트 디바이스가 매니페스트에 대한 그 요청에 포함된 선호들을 전송하는 경우에, 그것은 또한, SETTINGS 프레임이 요청 바로 뒤에 전송되는 것을 나타낼 수 있고; 이러한 SETTINGS 프레임은, 예를 들어 그것의 버퍼링 용량들에 일치하는 초기 윈도 사이즈(SETTINGS_INITIAL_WINDOW_SIZE)를 특정한다. 가능한 변동에 따르면, 이러한 SETTINGS 프레임은 연결 셋업 시간에 전송될 수 있다. 또 다른 가능성은 클라이언트 디바이스가, 적절한 사이즈를 가진 WINDOW_UPDATE를 전송하기 위해 제1 푸시된 데이터를 확인응답할 때를 위한 것이다.

[0563] 본 발명의 교시들에 따라서, 예를 들어 도 27에 설명된 바와 같이 방법을 실행하는 서버와 데이터를 교환할 때, 도 28은 클라이언트 디바이스에 의해 구현된 가능한 방법을 설명한다.

[0564] 이 방법의 가능한 애플리케이션에 따르면, 클라이언트 디바이스는 비디오 온 디맨드 서비스로부터 이익을 얻기

위해 서버에 연결된다. 클라이언트와 서버 사이의 연결 설정이 일반적이다. 본 예에서, 클라이언트 디바이스와 서버의 양쪽은, 예를 들어 이미 언급된 문서 "Hypertext Transfer Protocol version 2.0, draft-ietf-httpbis-http2-latest"에 설명된 HTTP/2.0 프로토콜을 사용하여 메시지들을 교환할 수 있다.

[0565] 한 시점(예를 들어, 클라이언트 디바이스에서의 사용자가 주어진 비디오를 선택할 때)에서, 클라이언트 디바이스는 미디어 표현(여기에는 사용자가 보고 싶은 비디오)을 기술하는 매니페스트의 어드레스(예를 들면, URL)에 대한 서버로부터의 정보를 얻는다.

[0566] 클라이언트 디바이스는 그 후 매니페스트를 다운로드하기 위한 요청을 준비한다(단계 2800). 바람직한 실시예에서, 클라이언트는 HTTP 헤더들을 통하여 비디오 해상도, 코덱들, 그것이 지원하는 대역폭에 대한 일부 선호들을 추가한다(단계 2801). 클라이언트 디바이스는 그후 그것의 요청을 서버에 전송한다(단계 2802).

[0567] 본 발명의 실시예에서, 클라이언트 디바이스는 그 후 단계 2803에서 그것의 버퍼링 용량들에 일치하는 초기 윈도 사이즈(SETTINGS_INITIAL_WINDOW_SIZE)를 나타내기 위해 HTTP/2.0 SETTINGS 프레임을 전송한다(상기에 언급된 문서 "Hypertext Transfer Protocol version 2.0, draft-ietf-httpbis-http2-latest", 섹션 3.8.5 참조).

[0568] 단계 2804에서, 클라이언트 디바이스는 다양한 서버 응답들의 처리를 시작한다: 매니페스트를 형성하는 데이터를 수신하고 이를 파싱하고(단계 2805), 서버에 의해 전송된 PUSH_PROMISE 프레임(들)도 파싱한다(단계 2806).

[0569] PUSH_PROMISE 프레임(들)에서 지정된 푸시(들)를 수용하거나 취소시키기로 결정하기 전에, 클라이언트는 서버에 의해 PUSH_PROMISE 프레임에 포함된 confidence_level 파라미터를 서버가 푸시하고(단계 2806) 체크(단계 2807)하려고 하는 리소스의 URL을 구축한다.

[0570] 동시에 그리고 매니페스트(또는 기술 파일)가 완전히 수신될 때, 클라이언트 디바이스는 그것이 얻고 싶은 원하는 미디어 세그먼트들의 리스트(즉, 그것의 필요들에 가장 적합한 각각의 세그먼트의 버전들의 리스트)를 구축하고(단계 2808) 현재의 segment_index 변수를 0으로 초기화한다(단계 2809). PUSH_PROMISE의 처리에서의 제1 단계는 confidence_level 파라미터의 체크로 구성된다(단계 2810a). 그 후, (미리 정의된) 클라이언트 설정 또는 사용자 선호들에 따라서 클라이언트는 신뢰의 소정 레벨 하에서의 PUSH_PROMISE, 예를 들어, PUSH_PROMISE 프레임들이 "low" 값을 가진 confidence_level 파라미터를 포함하는 PUSH_PROMISEs를 거절하기로 결정할 수 있다.

[0571] 클라이언트가 PUSH_PROMISE 프레임에서 언급된 URL을, (바로 전에 언급된 단계 2808에서 매니페스트로부터 도출된 바와 같이) 원하는 세그먼트의 URL과 매칭할 수 있다면(단계 2810b), 그것은 그들의 전송 상태로 전송 중인 세그먼트들의 리스트를 위한 테이블을 초기화한다(단계 2811). 클라이언트가 원하는 미디어 세그먼트들의 리스트에서, 단계 2810b에서 서버에 의해 푸시되도록 의도된 세그먼트를 식별할 수 없으면, 그것은 적절한 CANCEL 명령을 서버로 전송함으로써 푸시를 취소시킨다(단계 2812).

[0572] 단계 2810b에서 세그먼트 식별을 용이하게 하기 위해, 클라이언트는 MPD 트리 표현에서의 경로로서, 예를 들어 푸시된 세그먼트의 인덱스와 같은 추가 헤더 정보, 또는 기술 파일(즉, MPD 파일 또는 매니페스트)이 SegmentTemplate에 의존할 때의 URL 템플릿 파라미터들을 활용할 수 있다(도 5b 참조).

[0573] 이것은, PUSH_PROMISE를 구축할 때(상기 도 27의 설명 참조) 서버에 의해 삽입된 계층 관계를 이용하여 클라이언트가 다음과 같은 것들 이외에 현재 PUSH_PROMISE의 취소를 야기하는 순환적 CANCEL을 전송할 수 있기 때문에 여기서는(단계 2812) 특정 CANCEL 메시지이다.

[0574] 가능한 실시예에 따르면, 클라이언트 디바이스가 푸시 약속을 해석할 수 없을 때, 그것은 디폴트에 의해 미디어 리소스의 다음 시간적 세그먼트들에 해당하는 미디어 데이터의 모든 푸시들을 중지한다.

[0575] CANCEL 명령들의 이러한 새로운 사용은, 일단 그것이 미디어 세그먼트 식별의 관점에서 서버와 비동기화되면 클라이언트가 CANCEL 메시지들을 반복하는 것을 회피하게 될 것이다. 그런 경우, 클라이언트는 풀 모드로 후퇴할 것이다.

[0576] 서버로부터 푸시에 의해 수신될 세그먼트가 원하는 세그먼트에 해당될 때(테스트 2810b에서 참), 클라이언트는 그 후 PUSH_PROMISE 프레임들의 처리를 계속한다(테스트 2813 및 단계 2806 상의 루프).

[0577] 모든 PUSH_PROMISE 프레임들이 처리되었을 때, 클라이언트 디바이스는 수용된 PUSH_PROMISE에 해당하는 데이터의 수신 및 버퍼링을 예상하고 시작한다(단계 2814).

[0578] 충분한 미디어 세그먼트들이 클라이언트의 수신 버퍼에 수신되면(테스트 2815), 그들은 클라이언트에 의해 처리

된다(2816). 현재 segment_index 변수는 그 후 리스트에서 제1 세그먼트의 순서 번호로 업데이트된다(단계 2817). 모든 클라이언트가 클라이언트의 버퍼에 액세스할 수 없다는 것을 주목해야한다. 예를 들어, 웹 애플리케이션들은 특히, 보통 웹 브라우저 캐시에 액세스하지 않는다. 이런 경우에, 서버는 푸시된 세그먼트들의 리스트를 웹 애프리케이션 프로그램 클라이언트에 직접적으로 전송할 수 있다. 이러한 정보는 예를 들어, 웹 소켓 연결을 이용하여 서버로부터 클라이언트로 교환될 수 있다.

[0579] 모든 푸시된 미디어 세그먼트들이 처리되었을 때, 클라이언트는 그 후 표준 풀-기반 DASH로 돌아갈 수 있고(단계 2818), 변수 segment_index + 1에 의해 지정된 다음 세그먼트에 해당하는 데이터의 요청을 시작한다. 동시에, 푸시된 세그먼트 데이터는 선택된 비디오의 디코딩 및 표시를 시작하는데 사용된다.

[0580] 도 13은 실시예들에 따른 디바이스의 개략도이다. 디바이스는 서버, 클라이언트 또는 프록시일 수 있다. 디바이스는 실시예들에 따른 방법을 구현하도록 구성된 제어 유닛(1301)을 위한 작업 메모리로서 사용될 수 있는 RAM 메모리(1302)를 포함한다. 예를 들어, 제어 유닛은 ROM 메모리(1303)로부터 로딩된 컴퓨터 프로그램의 명령어들을 실행하도록 구성될 수 있다. 프로그램은 또한 하드 드라이브(1306)로부터 로딩될 수 있다. 예를 들어, 컴퓨터 프로그램은 도 8-12, 14, 15, 17, 20-22 및 26-28의 흐름도들 그리고 상기 설명을 기초로 설계된다.

[0581] 디바이스는 또한 단일 네트워크 인터페이스일 수 있는 네트워크 인터페이스(1304)를 포함하거나, 일련의 네트워크 인터페이스들(예를 들어, 여러 무선 인터페이스들, 또는 여러 타입의 유선 또는 무선 인터페이스들)을 포함한다. 디바이스는 정보를 사용자에 표시하기 위해 그리고 사용자로부터의 입력들을 수신하기 위한 사용자 인터페이스(1305)를 포함할 수 있다.

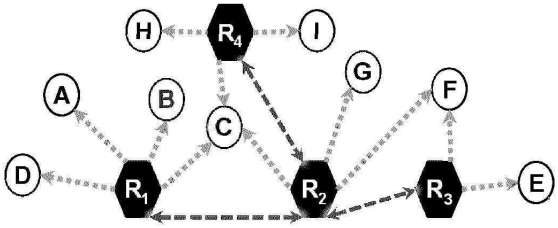
[0582] 디바이스는 또한 외부 디바이스들로부터/로 데이터를 수신 및/또는 전송하기 위한 입출력 모듈(1307)을 포함할 수 있다.

[0583] 본 발명이 도면들 및 상기 설명에서 상세하게 예시되고 설명되었지만, 이러한 예시 및 설명은 제한이 아닌 예시 또는 사례로 간주되어야 하고; 본 발명은 개시된 실시예들에 한정되지는 않는다. 본 분야의 숙련자들에 의해, 도면들, 개시 내용 및 첨부된 청구항들에 대한 연구로부터 청구된 발명을 실시하는데 있어서, 개시된 실시예들에 대한 또 다른 변형들이 이해되어지고 수행될 수 있다.

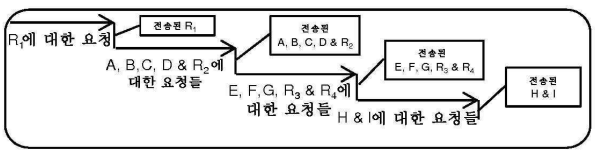
[0584] 청구항들에서, 용어 "포함하는(comprising)"은 다른 요소들 또는 단계들을 배제하지 아니하며, 부정 관사 "하나(a)" 또는 "하나(an)"는 복수를 배제하지 아니한다. 단일 프로세서 또는 다른 유닛은 청구범위에 인용된 여러 항목의 기능을 충족시킬 수 있다. 상이한 특징들이 서로 상이한 종속항들에 인용되어 있다는 사실만으로는 이들 특징의 조합이 유리하게 이용될 수 없다는 것을 나타내지 않는다. 청구항들 내의 임의의 참조 부호도 본 발명의 범위를 제한하는 것으로서 해석되어서는 안 된다.

도면

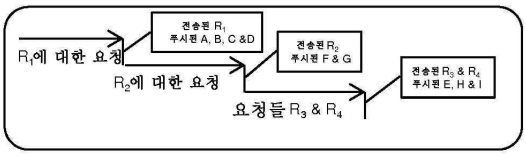
도면1a



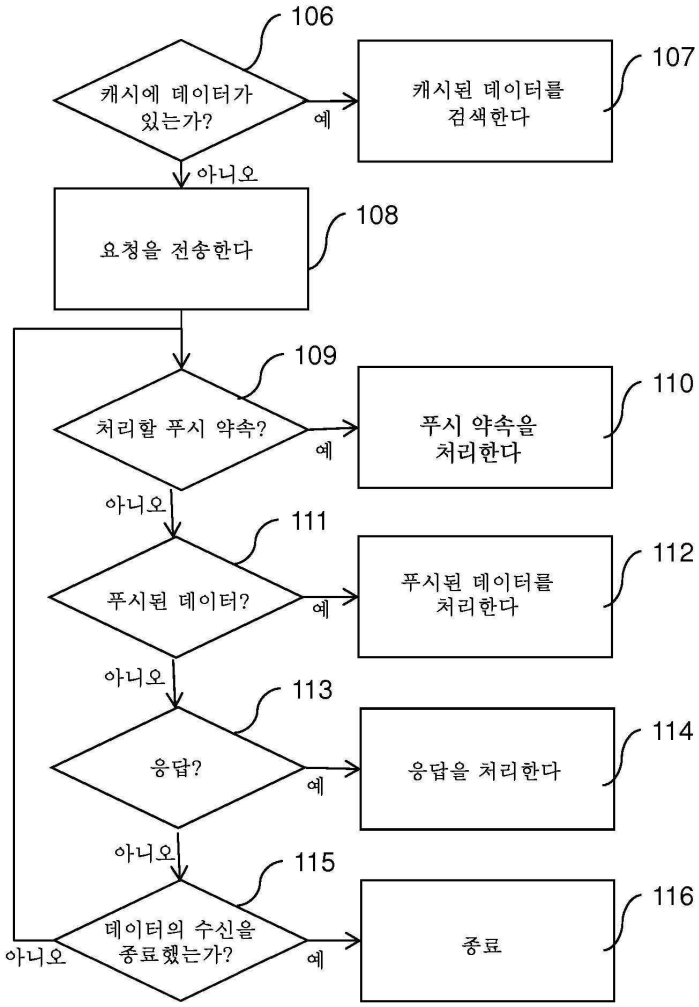
도면1b



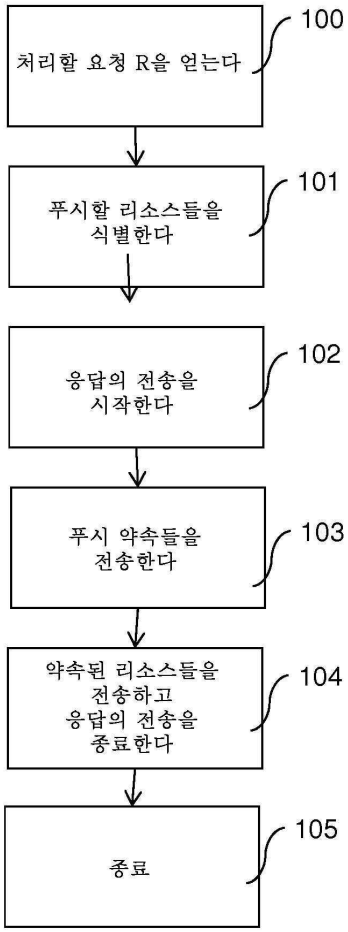
도면1c


전송원 R1
푸시원 A, B, C &D
R1에 대한 요청
전송원 R2
푸시원 F & G
R2에 대한 요청
전송원 R3 & R4
푸시원 E, H & I
요청들 R3 & R4

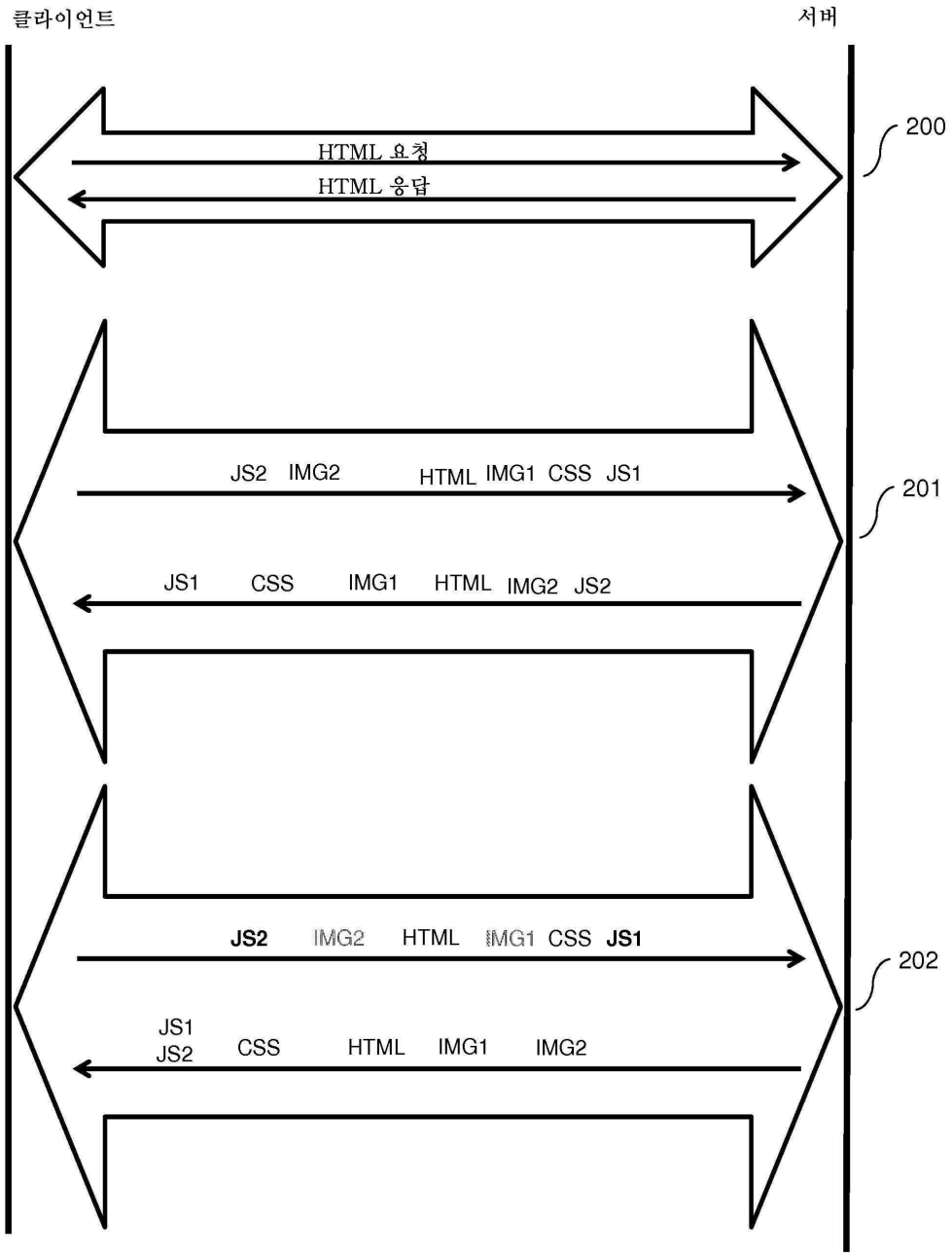
도면1d


106
캐시에 데이터가 있는가?
예
107
캐시된 데이터를 검색한다
아니오
108
요청을 전송한다
109
처리할 푸시 약속?
예
110
푸시 약속을 처리한다
아니오
111
푸시된 데이터?
예
112
푸시된 데이터를 처리한다
아니오
113
응답?
예
114
응답을 처리한다
아니오
115
데이터의 수신을 종료했는가?
아니오
예
116
종료

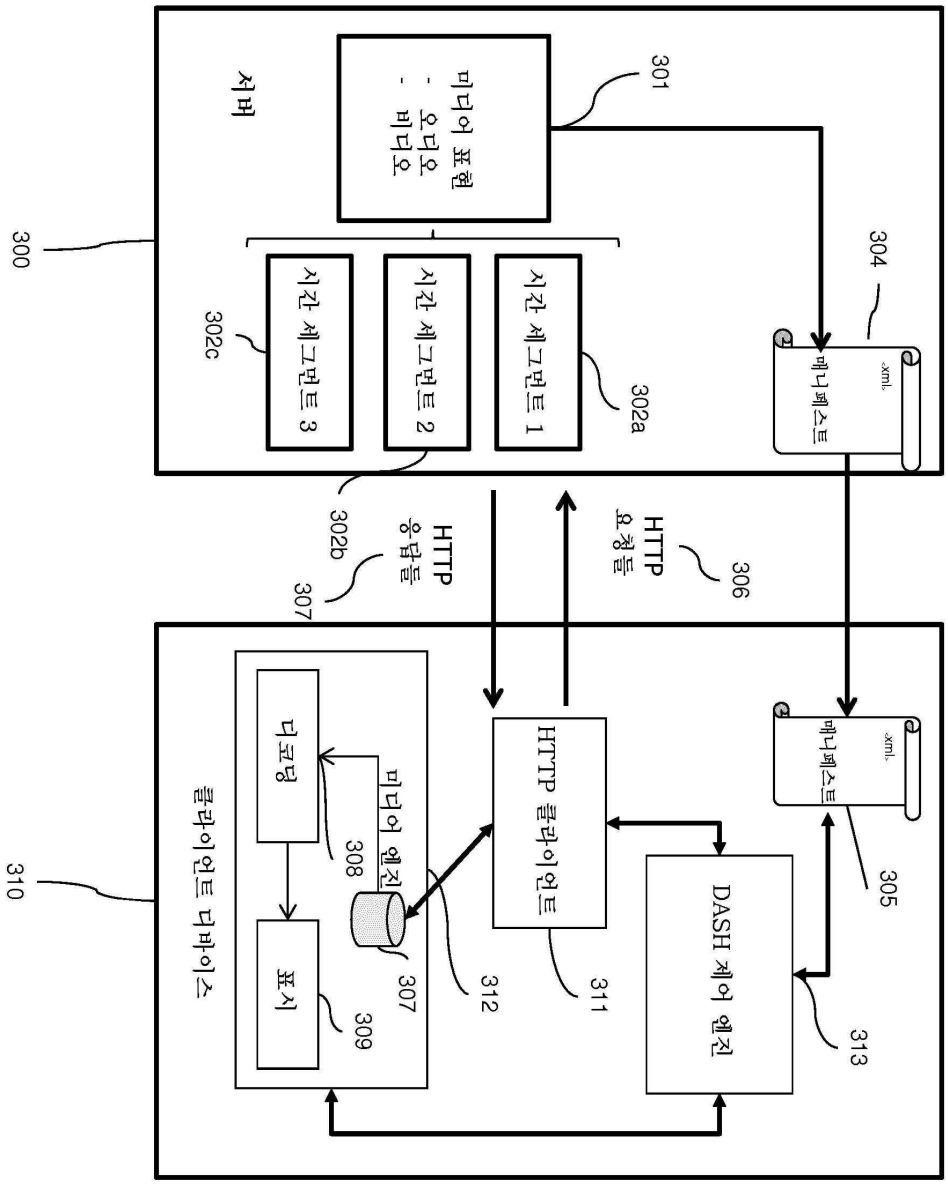
도면1e


처리할 요청 R을 얻는다
100
푸시할 리소스들을 식별한다
101
응답의 전송을 시작한다
102
푸시 약속들을 전송한다
103
약속된 리소스들을 전송하고 응답의 전송을 종료한다
104
종료
105

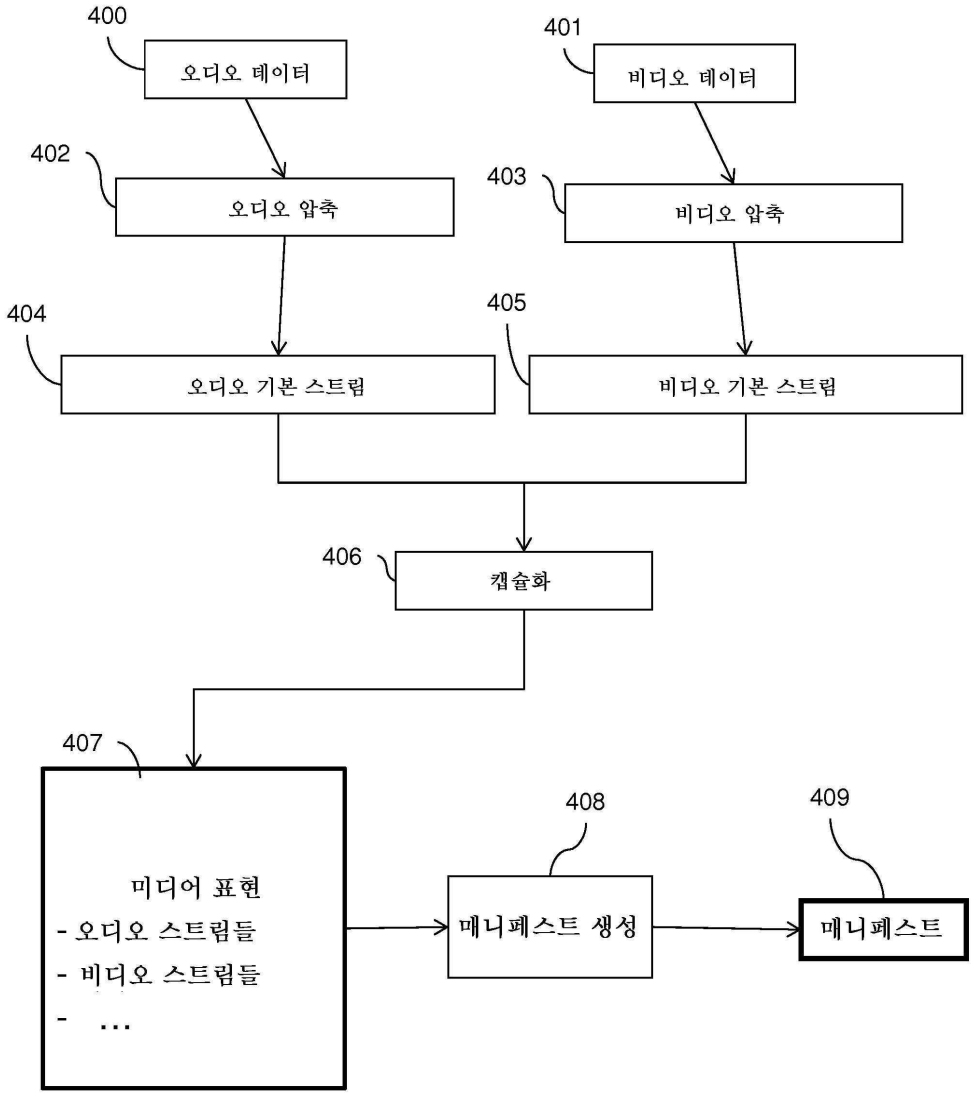
도면2


클라이언트
서버
200
HTML 요청
HTML 응답
201
JS2 IMG2
HTML IMG1 CSS JS1
JS1 CSS IMG1 HTML IMG2 JS2
202
JS2 IMG2 HTML IMG1 CSS JS1
JS1
JS2
CSS HTML IMG1 IMG2

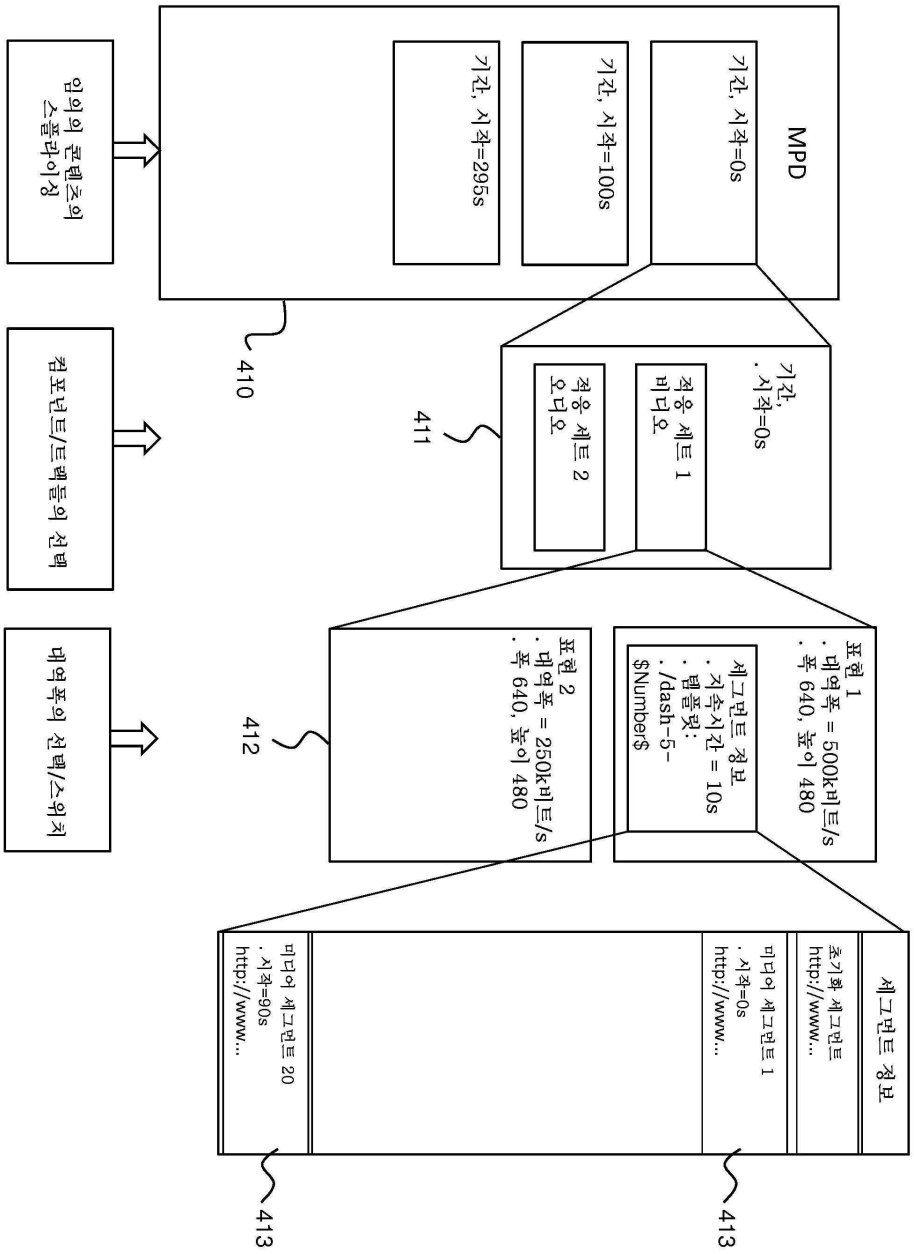
도면3


서버
미디어 표현
- 오디오
- 비디오
301
300
304
시간 세그먼트 3
시간 세그먼트 2
시간 세그먼트 1
302c
302a
매니페스트
.xml
HTTP
응답들
302b
307
HTTP
요청들
306
클라이언트 디바이스
310
디코딩
미디어 엔진
308
표시
309
307
312
HTTP 클라이언트
311
DASH 제어 엔진
313
매니페스트
.xml
305

도면4a


400
오디오 데이터
401
비디오 데이터
402
오디오 압축
403
비디오 압축
404
오디오 기본 스트림
405
비디오 기본 스트림
406
캡슐화
407
미디어 표현
- 오디오 스트림들
- 비디오 스트림들
- ...
408
매니페스트 생성
409
매니페스트

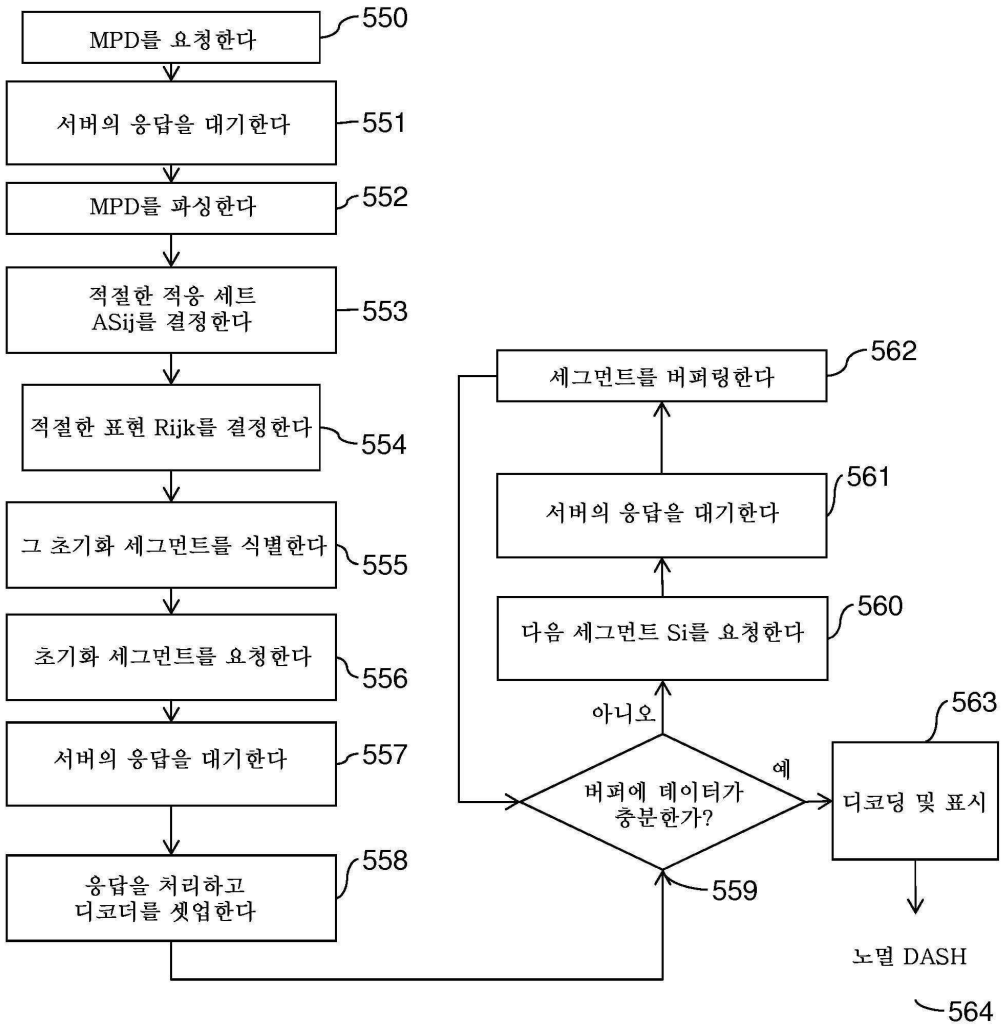
도면4b


MPD
기간, 시작=0s
기간, 시작=100s
기간, 시작=295s
410
기간,
. 시작=0s
적응 세트 1
비디오
적응 세트 2
오디오
411
표현 1
. 대역폭 = 500k비트/s
. 폭 640, 높이 480
세그먼트 정보
. 지속시간 = 10s
. 템플릿:
. /dash-5-
$Number$
표현 2
. 대역폭 = 250k비트/s
. 폭 640, 높이 480
412
세그먼트 정보
초기화 세그먼트
http://www...
미디어 세그먼트 1
. 시작=0s
http://www...
미디어 세그먼트 20
. 시작=90s
http://www...
413
413
임의의 콘텐츠의
스플라이싱
컴포넌트/트랙들의 선택
대역폭의 선택/스위치

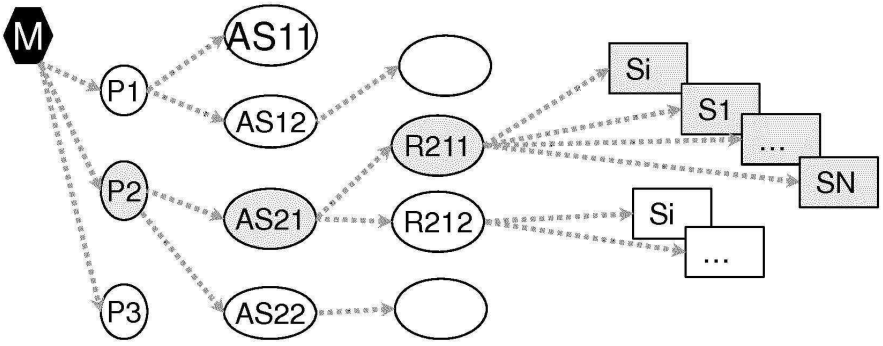
도면5

```
<?xml version="1.0"?>
        <MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="urn:mpeg:DASH:schema:MPD:2011"
         xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd"
         type="static"
         mediaPresentationDuration="PT3256S"
         minBufferTime="PT1.2S"
         profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">
         <BaseURL>http://cdn1.example.com/</BaseURL>
         <BaseURL>http://cdn2.example.com/</BaseURL>
        <Period>
<!-- English Audio -->
    500 <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="en"
        subsegmentAlignment="true" subsegmentStartsWithSAP="1">
         <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-
        656164657221"/>
            <Representation id="1" bandwidth="64000">
    501         <BaseURL>7657412348.mp4</BaseURL>
            </Representation>
            <Representation id="2" bandwidth="32000">
    502         <BaseURL>3463646346.mp4</BaseURL>
            </Representation>
<!-- Video -->
    503 <AdaptationSet mimeType="video/mp4" codecs="avc1.4d0228"
        subsegmentAlignment="true" subsegmentStartsWithSAP="2">
         <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-
        656164657221"/>
            <Representation id="6" bandwidth="256000" width="320" height="240">
                <BaseURL>8563456473.mp4</BaseURL>
            </Representation>
            <Representation id="7" bandwidth="512000" width="320" height="240">
                <BaseURL>56363634.mp4</BaseURL>
            </Representation>
            <Representation id="8" bandwidth="1024000" width="640" height="480">
                <BaseURL>562465736.mp4</BaseURL>
            </Representation>
            <Representation id="9" bandwidth="1384000" width="640" height="480">
                <BaseURL>41325645.mp4</BaseURL>
            </Representation>
            <Representation id="A" bandwidth="1536000" width="1280" height="720">
                <BaseURL>89045625.mp4</BaseURL>
            </Representation>
            <Representation id="B" bandwidth="2048000" width="1280" height="720">
                <BaseURL>23536745734.mp4</BaseURL>
             </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```

도면5a


MPD를 요청한다
550
서버의 응답을 대기한다
551
MPD를 파싱한다
552
적절한 적응 세트 ASij를 결정한다
553
적절한 표현 Rijk를 결정한다
554
그 초기화 세그먼트를 식별한다
555
초기화 세그먼트를 요청한다
556
서버의 응답을 대기한다
557
응답을 처리하고 디코더를 셋업한다
558
세그먼트를 버퍼링한다
562
서버의 응답을 대기한다
561
다음 세그먼트 Si를 요청한다
560
아니오
예
버퍼에 데이터가 충분한가?
559
디코딩 및 표시
563
노멀 DASH
564

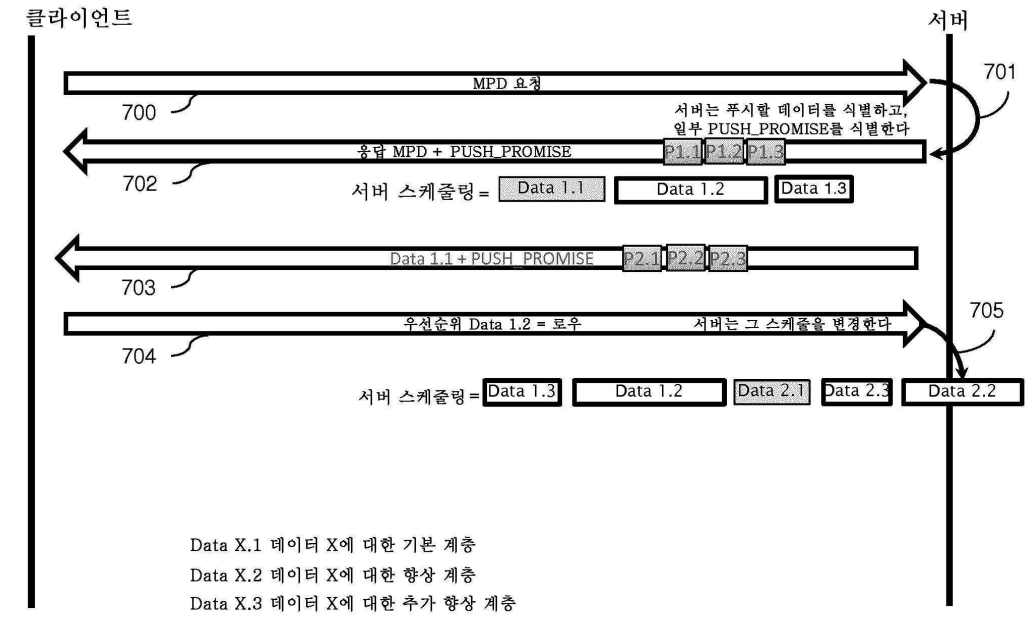
도면5b


M
P1
P2
P3
AS11
AS12
AS21
AS22
R211
R212
Si
S1
...
SN
Si
...

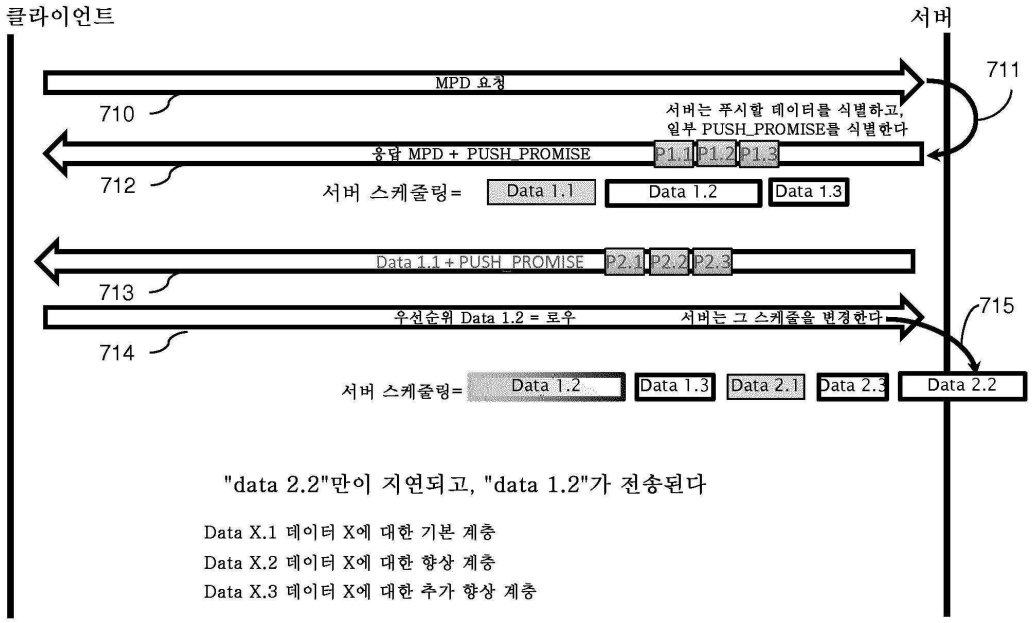
도면6

1. <MPD ...>
2. <Period>
3. <BaseURL>http://myserver.com/media</BaseURL>
4. <SegmentList>
5. <Initialization sourceURL=« URL_SI »/>
6. </SegmentList>
7. <AdaptationSet id='1' contentType='video' framerate='30'>
8. <!-- Base layer description -->
9. <Representation id='R1' mimeType='video/mp4' width='2000' height='1000' bandwidth='512000'>
10. <SegmentList> <SegmentURL media=« URL_BL »/> </SegmentList>
11. </Representation>
12. </AdaptationSet>
13. <!-- Enhancement layer description, composite track -->
14. <AdaptationSet id='2' contentType='video' framerate='30'>
15. <!- Tile a, b, c and d are described as components of composite track -->
16. <ContentComponent id='Ta' /><Role schemeIdUri='tiling' value='1:1'/></ContentComponent>
17. <ContentComponent id='Tb'/><Role schemeIdUri='tiling' value='1:2'/></ContentComponent>
18. <ContentComponent id='Tc'/> <Role schemeIdUri='tiling' value='2:1'/></ContentComponent>
19. <ContentComponent id='Td'/> <Role schemeIdUri='tiling' value='2:2'/></ContentComponent>
20. <ContentComponent id='CT'/> <Role schemeIdUri='...role' value='main'/></ContentComponent>
21. <Representation id='R2' mimeType='video/mp4' dependencyId='R1' width='4000' height='2000'
bandwidth='2048000'>
22. <SubRepresentation level='1' contentComponent='Ta' width='2000' height='1000'/>
23. <SubRepresentation level='2' contentComponent='Tb' width='2000' height='1000'/>
24. <SubRepresentation level='3' contentComponent='Tc' width='2000' height='1000'/>
25. <SubRepresentation level='4' contentComponent='Td' width='2000' height='1000'/>
26. <SubRepresentation level='5' contentComponent='CT' required/>
27. <SegmentList>
28. <SegmentURL media=« URL_X index_range=« 0-43 » »/>
29. ....
30. </SegmentList>
31. </Representation>
32. </AdaptationSet>
33. </Period>
34. </MPD>
600
601
602
603
610
620

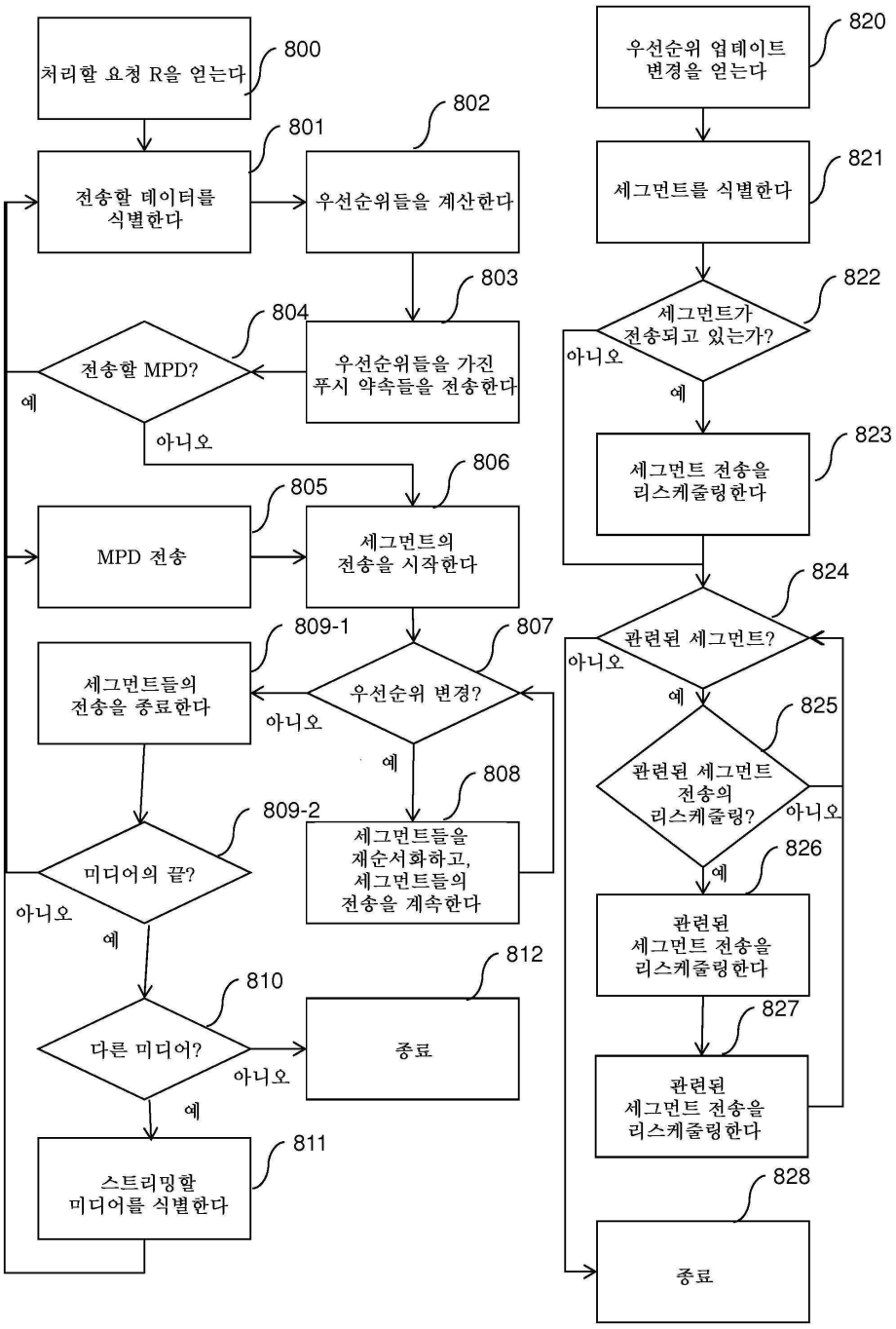
도면7a


클라이언트
서버
MPD 요청
700
701
서버는 푸시할 데이터를 식별하고,
일부 PUSH_PROMISE를 식별한다
응답 MPD + PUSH_PROMISE
P1.1 P1.2 P1.3
702
서버 스케줄링 =
Data 1.1
Data 1.2
Data 1.3
Data 1.1 + PUSH_PROMISE
P2.1 P2.2 P2.3
703
705
우선순위 Data 1.2 = 로우
서버는 그 스케줄을 변경한다
704
서버 스케줄링 =
Data 1.3
Data 1.2
Data 2.1
Data 2.3
Data 2.2
Data X.1 데이터 X에 대한 기본 계층
Data X.2 데이터 X에 대한 향상 계층
Data X.3 데이터 X에 대한 추가 향상 계층

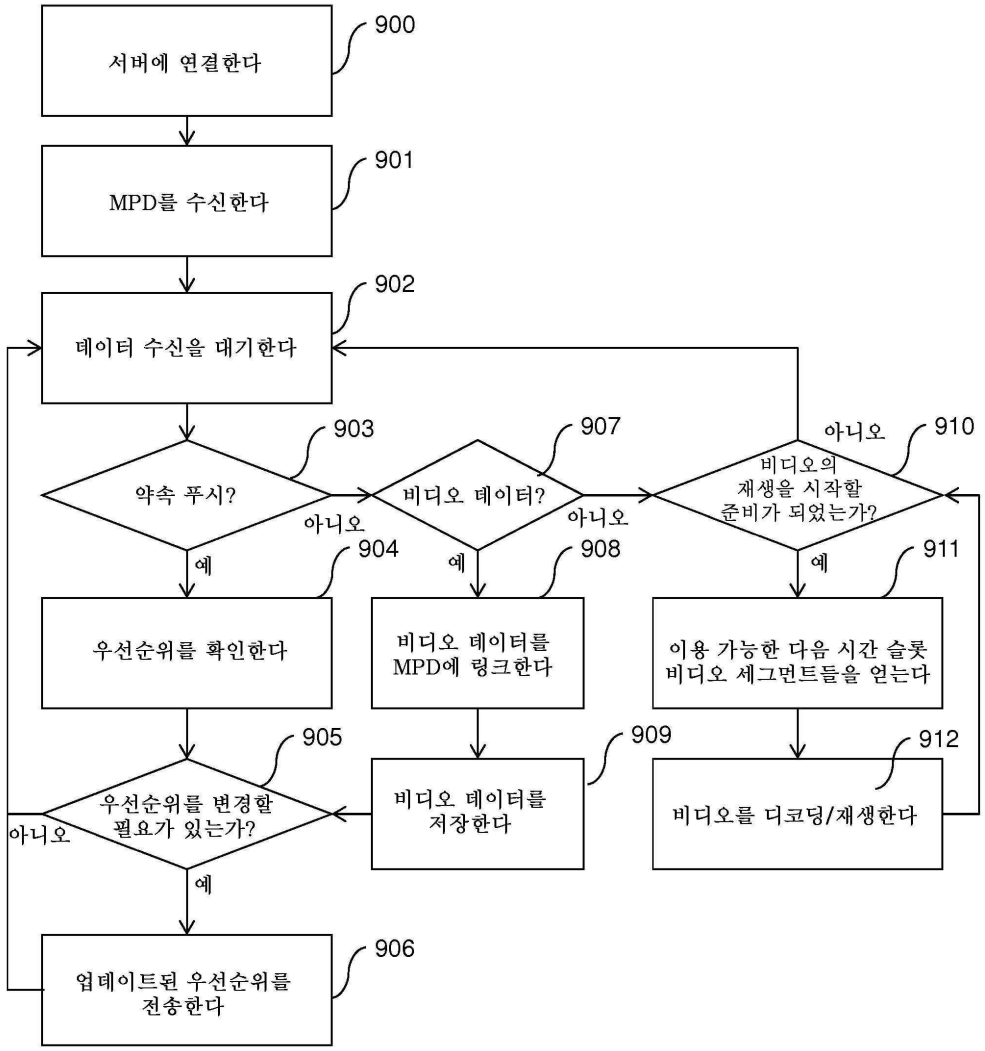
도면7b


클라이언트
서버
MPD 요청
710
711
서버는 푸시할 데이터를 식별하고,
일부 PUSH_PROMISE를 식별한다
응답 MPD + PUSH_PROMISE
P1.1 P1.2 P1.3
712
서버 스케줄링 =
Data 1.1
Data 1.2
Data 1.3
Data 1.1 + PUSH_PROMISE
P2.1 P2.2 P2.3
713
715
우선순위 Data 1.2 = 로우
서버는 그 스케줄을 변경한다
714
서버 스케줄링 =
Data 1.2
Data 1.3
Data 2.1
Data 2.3
Data 2.2
"data 2.2"만이 지연되고, "data 1.2"가 전송된다
Data X.1 데이터 X에 대한 기본 계층
Data X.2 데이터 X에 대한 향상 계층
Data X.3 데이터 X에 대한 추가 향상 계층

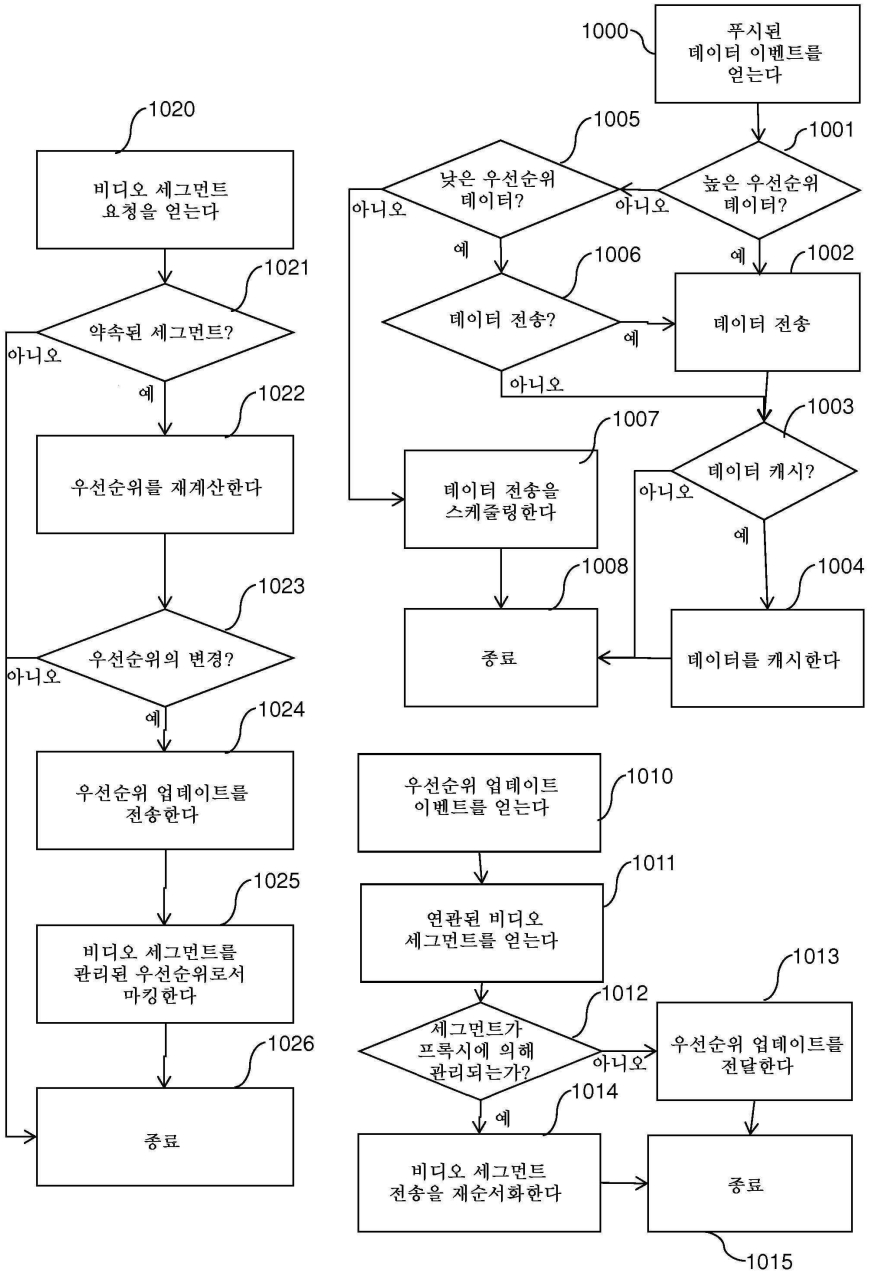
도면8


처리할 요청 R을 얻는다
800
전송할 데이터를 식별한다
801
우선순위들을 계산한다
802
우선순위들을 가진 푸시 약속들을 전송한다
803
전송할 MPD?
804
예
아니오
MPD 전송
805
세그먼트의 전송을 시작한다
806
우선순위 변경?
807
예
아니오
세그먼트들의 전송을 종료한다
809-1
세그먼트들을 재순서화하고, 세그먼트들의 전송을 계속한다
808
미디어의 끝?
809-2
아니오
예
다른 미디어?
810
아니오
예
종료
812
스트리밍할 미디어를 식별한다
811
우선순위 업데이트 변경을 얻는다
820
세그먼트를 식별한다
821
세그먼트가 전송되고 있는가?
822
아니오
예
세그먼트 전송을 리스케줄링한다
823
관련된 세그먼트?
824
아니오
예
관련된 세그먼트 전송의 리스케줄링?
825
아니오
예
관련된 세그먼트 전송을 리스케줄링한다
826
관련된 세그먼트 전송을 리스케줄링한다
827
종료
828

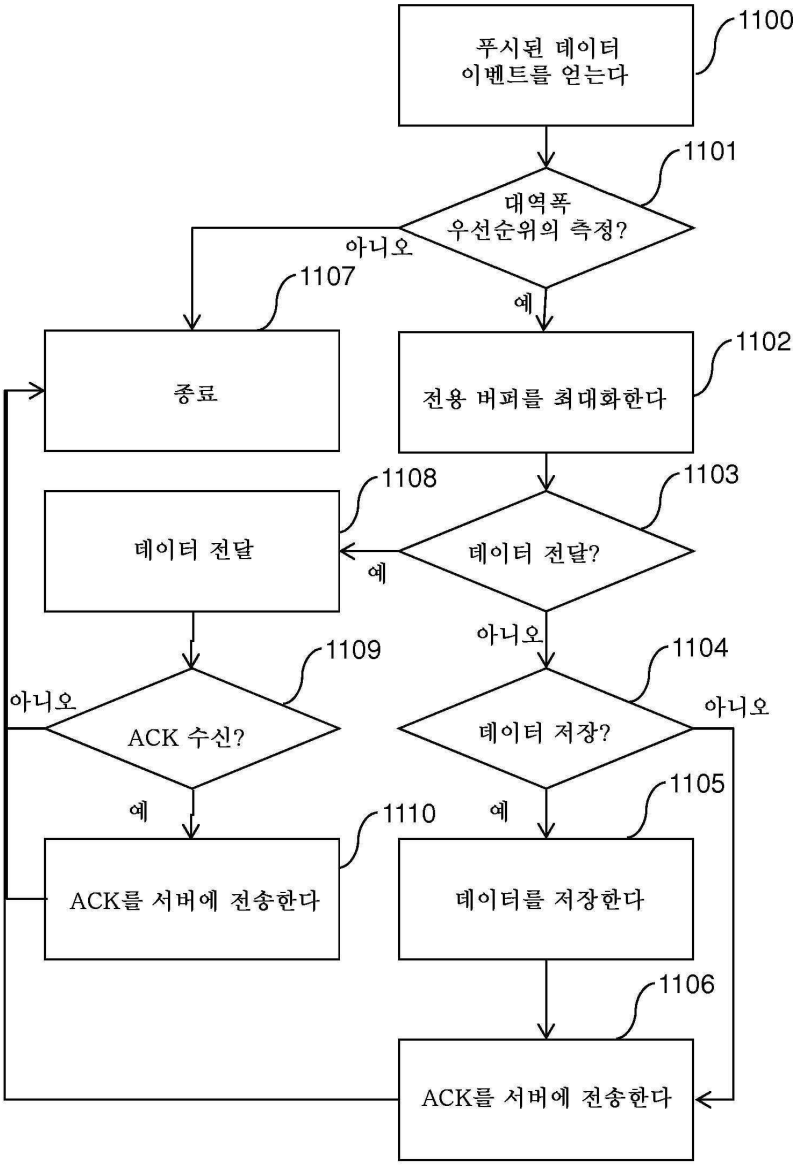
도면9


서버에 연결한다
900
MPD를 수신한다
901
데이터 수신을 대기한다
902
약속 푸시?
903
예
아니오
우선순위를 확인한다
904
우선순위를 변경할 필요가 있는가?
905
아니오
예
업데이트된 우선순위를 전송한다
906
비디오 데이터?
907
예
아니오
비디오 데이터를 MPD에 링크한다
908
비디오 데이터를 저장한다
909
비디오의 재생을 시작할 준비가 되었는가?
910
아니오
예
이용 가능한 다음 시간 슬롯 비디오 세그먼트들을 얻는다
911
비디오를 디코딩/재생한다
912

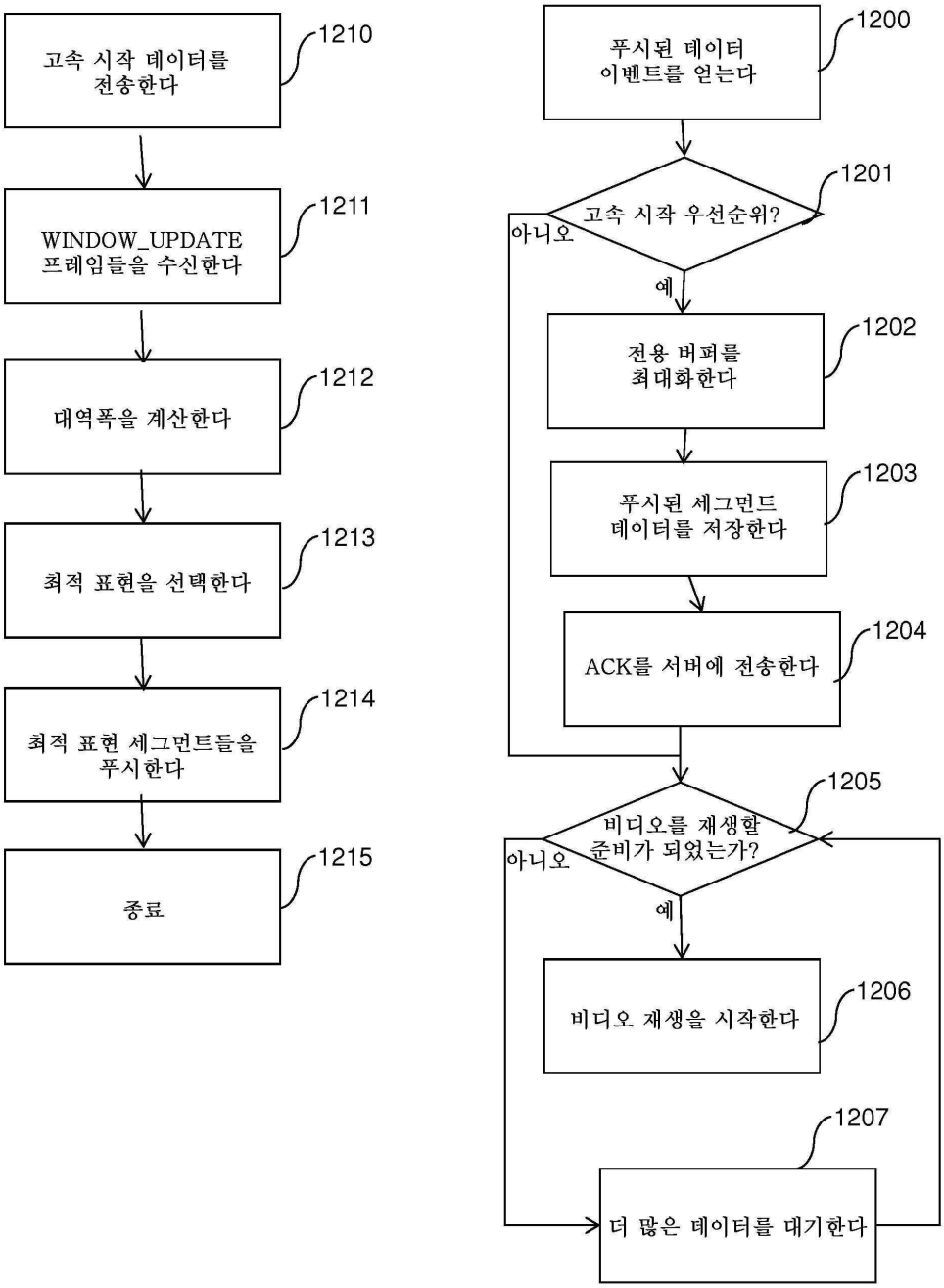
도면10


1000
푸시된 데이터 이벤트를 얻는다
1001
높은 우선순위 데이터?
아니오
1005
낮은 우선순위 데이터?
아니오
예
1002
데이터 전송
예
1006
데이터 전송?
아니오
예
1003
데이터 캐시?
아니오
예
1004
데이터를 캐시한다
1007
데이터 전송을 스케줄링한다
1008
종료
1010
우선순위 업데이트 이벤트를 얻는다
1011
연관된 비디오 세그먼트를 얻는다
1012
세그먼트가 프록시에 의해 관리되는가?
아니오
예
1013
우선순위 업데이트를 전달한다
1014
비디오 세그먼트 전송을 재순서화한다
1015
종료
1020
비디오 세그먼트 요청을 얻는다
1021
약속된 세그먼트?
아니오
예
1022
우선순위를 재계산한다
1023
우선순위의 변경?
아니오
예
1024
우선순위 업데이트를 전송한다
1025
비디오 세그먼트를 관리된 우선순위로서 마킹한다
1026
종료

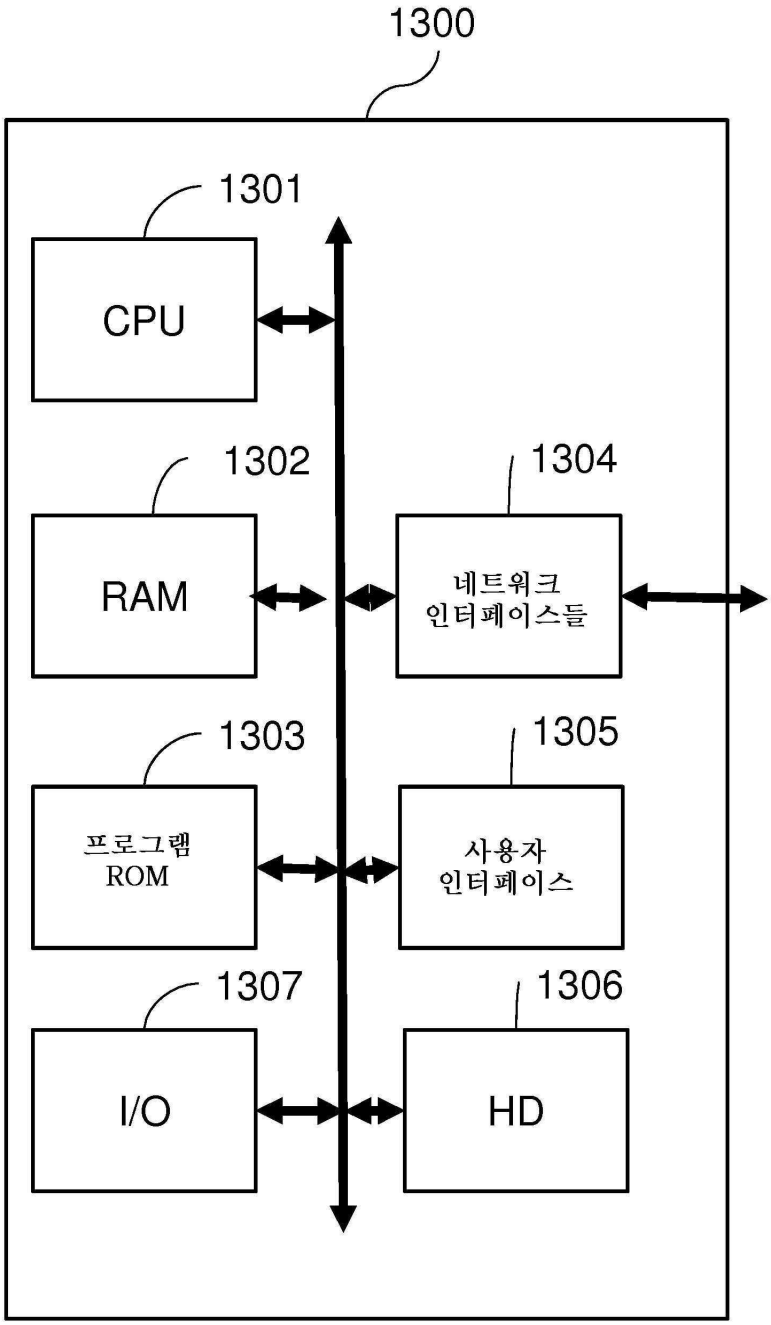
도면11


푸시된 데이터 이벤트를 얻는다
1100
대역폭 우선순위의 측정?
1101
아니오
예
1107
종료
전용 버퍼를 최대화한다
1102
데이터 전달
1108
데이터 전달?
1103
예
아니오
ACK 수신?
1109
아니오
데이터 저장?
1104
아니오
예
예
1110
ACK를 서버에 전송한다
데이터를 저장한다
1105
1106
ACK를 서버에 전송한다

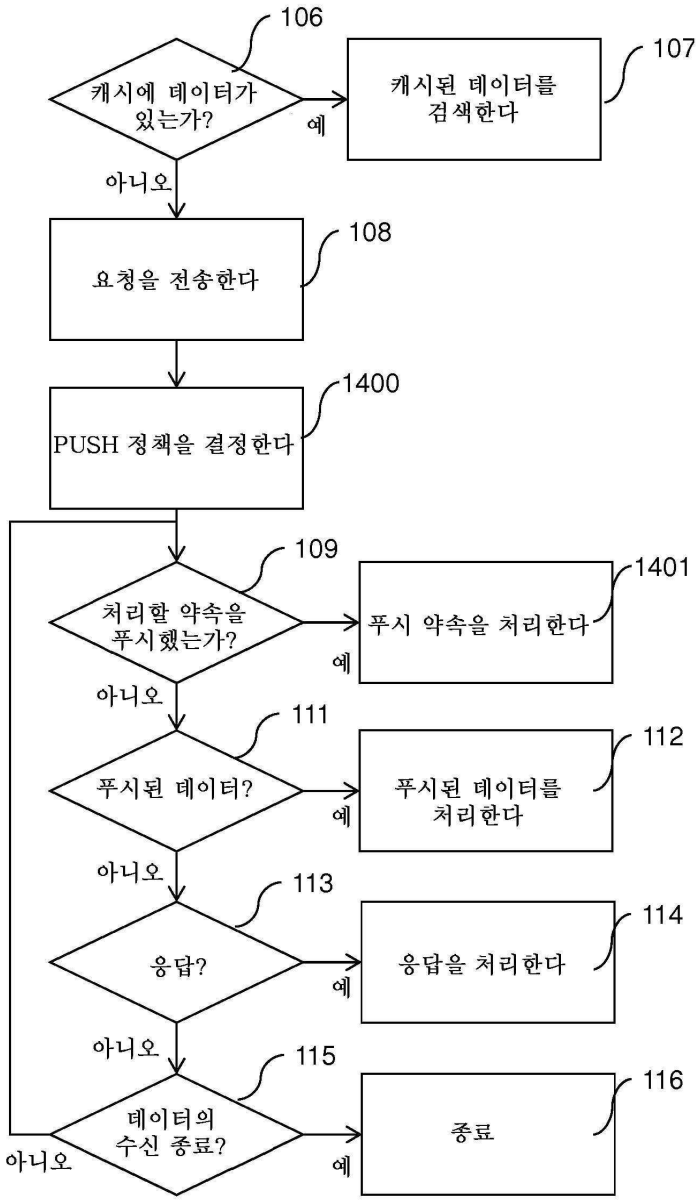
도면12


고속 시작 데이터를 전송한다
1210
WINDOW_UPDATE 프레임들을 수신한다
1211
대역폭을 계산한다
1212
최적 표현을 선택한다
1213
최적 표현 세그먼트들을 푸시한다
1214
종료
1215
푸시된 데이터 이벤트를 얻는다
1200
고속 시작 우선순위?
1201
아니오
예
전용 버퍼를 최대화한다
1202
푸시된 세그먼트 데이터를 저장한다
1203
ACK를 서버에 전송한다
1204
비디오를 재생할 준비가 되었는가?
1205
아니오
예
비디오 재생을 시작한다
1206
더 많은 데이터를 대기한다
1207

도면13

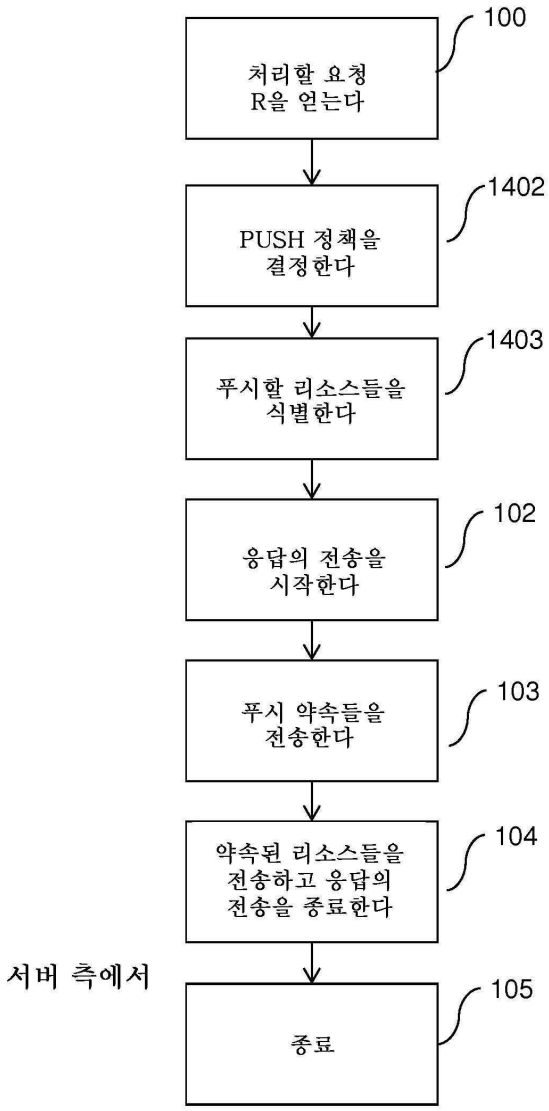

1300
1301
CPU
1302
RAM
1304
네트워크
인터페이스들
1303
프로그램
ROM
1305
사용자
인터페이스
1307
I/O
1306
HD

도면14a

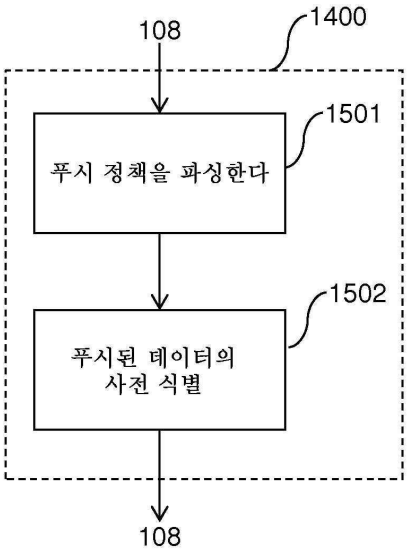


클라이언트 측에서

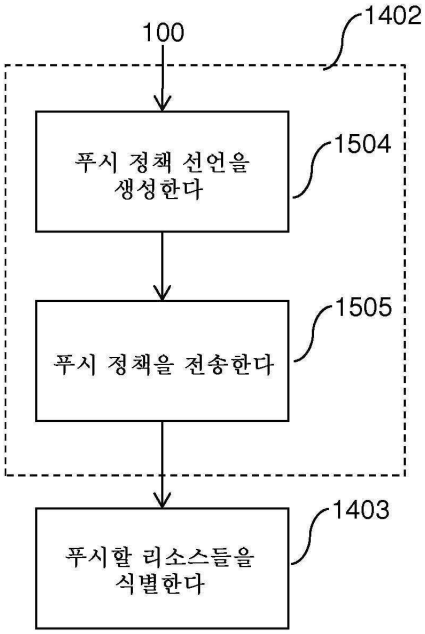
도면14b


서버 측에서
처리할 요청 R을 얻는다
100
PUSH 정책을 결정한다
1402
푸시할 리소스들을 식별한다
1403
응답의 전송을 시작한다
102
푸시 약속들을 전송한다
103
약속된 리소스들을 전송하고 응답의 전송을 종료한다
104
종료
105

도면15a


108
1400
1501
푸시 정책을 파싱한다
1502
푸시된 데이터의
사전 식별
108

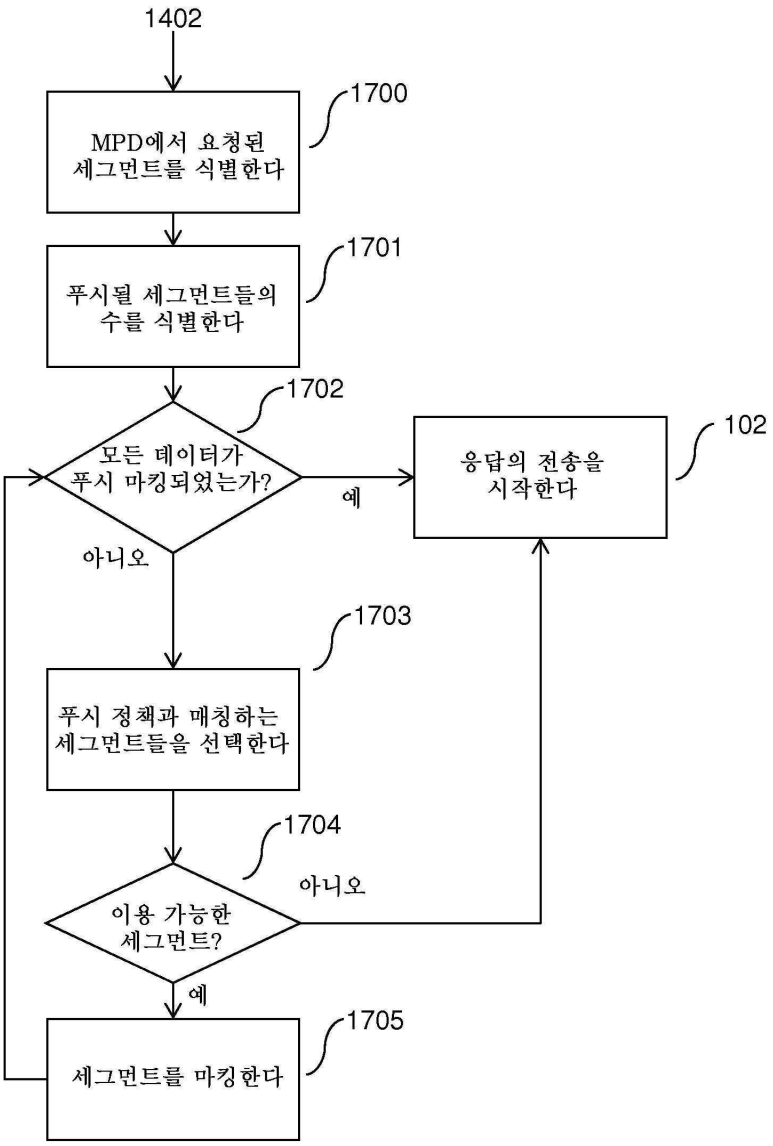
도면15b


100
1402
푸시 정책 선언을
생성한다
1504
1505
푸시 정책을 전송한다
1403
푸시할 리소스들을
식별한다

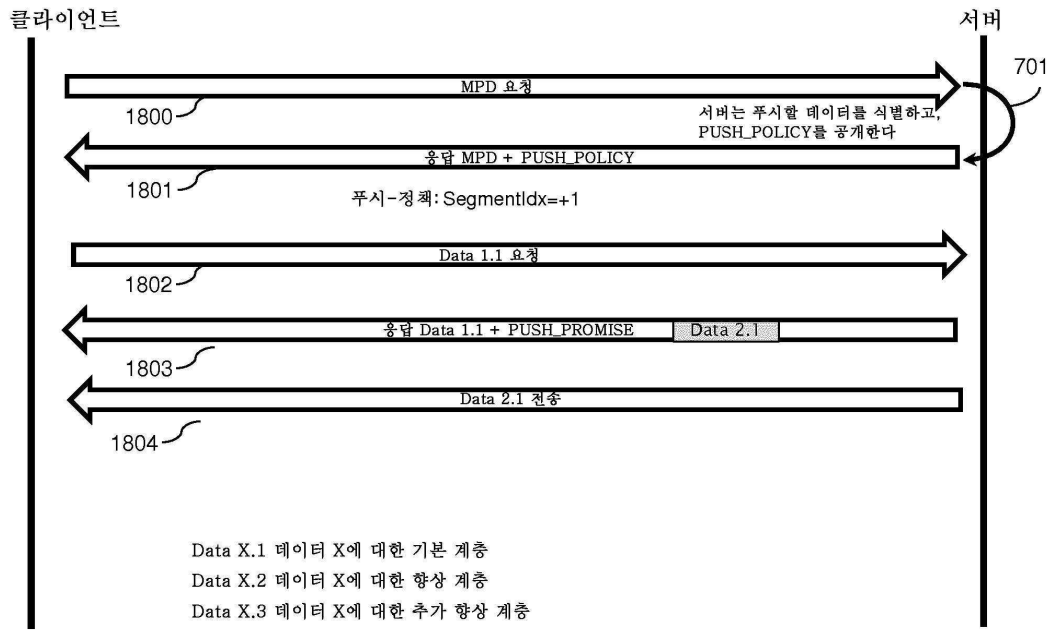
도면16


1600
1601
1602
1603
1604
<MPD …>
<PushPolicy SegmentIdx="+2"/>
<Period>
<BaseURL>http://myserver.com/media</BaseURL>
<SegmentList>
<Initialization sourceURL=« URL_SI »/>
</SegmentList>
<AdaptationSet id="AS1" mimeType='video/mp4' codecs='hev1' >
<Representation id='R0' width='1920' height='1080' frameRate='30'… …bandwidth='256000'>
<Role schemeIdUri="urn:mpeg:DASH:role:2011" value="main"/>
<SegmentList duration='10'>
<SegmentURL media='seg-full-AS1R0-1.mp4' priorityIdx='1'/>
<SegmentURL media='seg-full-AS1R0-2.mp4' priorityIdx='3'/>
<SegmentURL media='seg-full-AS1R0-3.mp4' priorityIdx='2'/>
…
</SegmentList>
</Representation>
<Representation id='R1' width='1920' height='1080' frameRate='30'… …bandwidth='256000'>
<Role schemeIdUri="urn:mpeg:DASH:role:2011" value="associate"/>
<SegmentList duration='10'>
<SegmentURL media='seg-full-AS1R1-1.mp4'/>
<SegmentURL media='seg-full-AS1R1-2.mp4'/>
<SegmentURL media='seg-full-AS1R1-3.mp4'/>
…
</SegmentList>
</Representation>
</AdaptationSet>
<AdaptationSet id='AS2' mimeType='video/mp4' codecs='hev1' >
<Representation id='R1' width='2000' height='1000' bandwidth='512000'>
<SegmentList>
…
</SegmentList>
</Representation>
</AdaptationSet>
</Period>
<Period>
…
</Period>
</MPD>

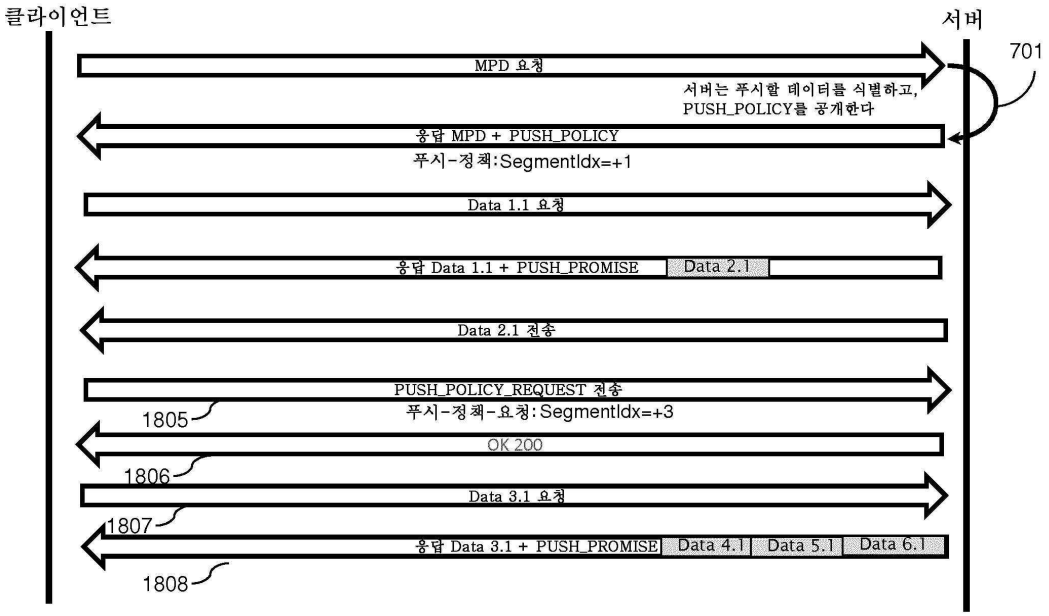
도면17


1402
MPD에서 요청된 세그먼트를 식별한다
1700
푸시될 세그먼트들의 수를 식별한다
1701
모든 데이터가 푸시 마킹되었는가?
1702
예
응답의 전송을 시작한다
102
아니오
푸시 정책과 매칭하는 세그먼트들을 선택한다
1703
이용 가능한 세그먼트?
1704
아니오
예
세그먼트를 마킹한다
1705

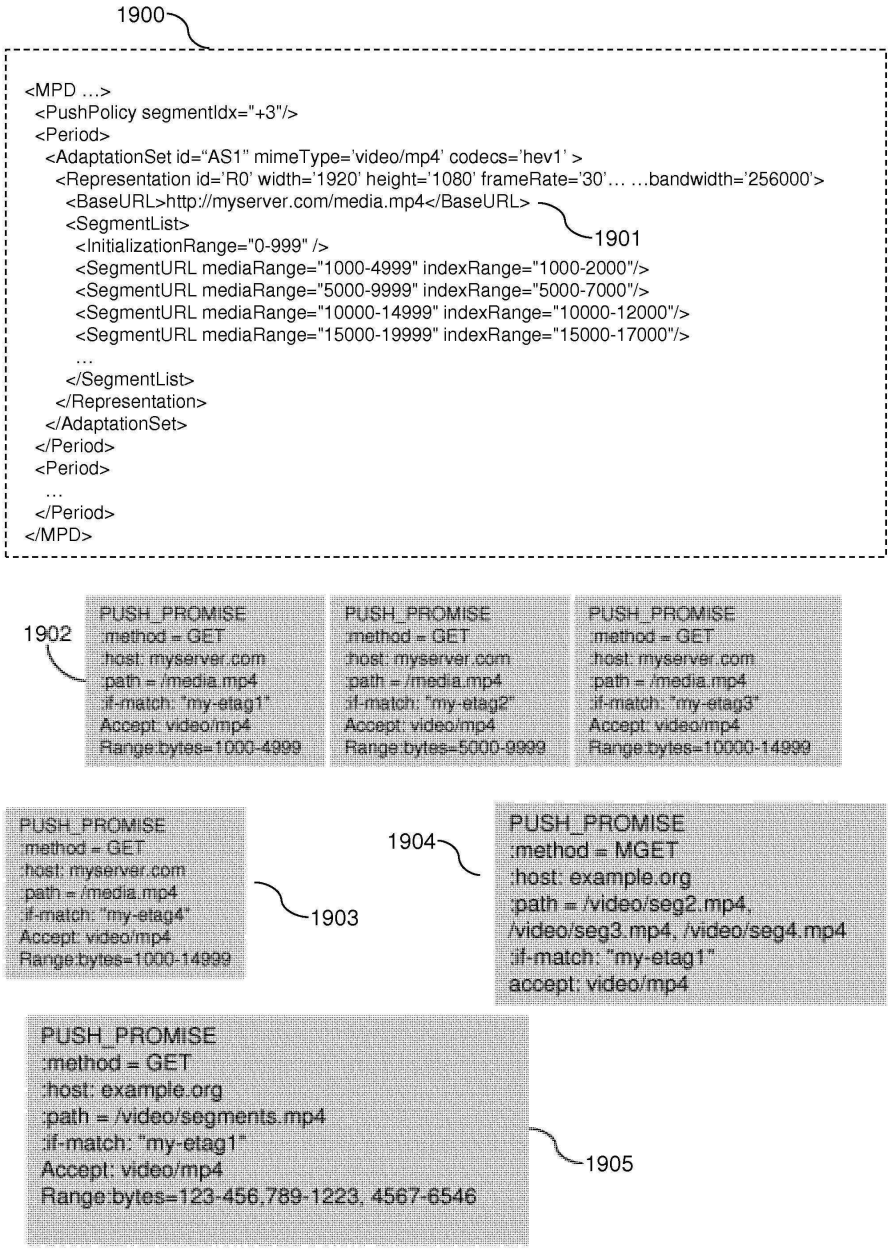
도면18a


클라이언트
서버
701
MPD 요청
1800
서버는 푸시할 데이터를 식별하고,
PUSH_POLICY를 공개한다
응답 MPD + PUSH_POLICY
1801
푸시-정책: SegmentIdx=+1
Data 1.1 요청
1802
응답 Data 1.1 + PUSH_PROMISE
Data 2.1
1803
Data 2.1 전송
1804
Data X.1 데이터 X에 대한 기본 계층
Data X.2 데이터 X에 대한 향상 계층
Data X.3 데이터 X에 대한 추가 향상 계층

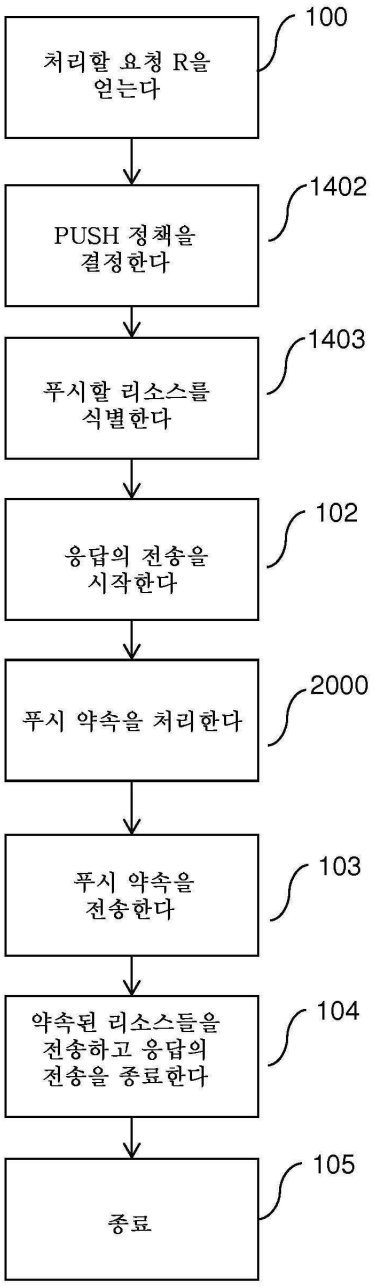
도면18b


클라이언트
서버
701
MPD 요청
서버는 푸시할 데이터를 식별하고,
PUSH_POLICY를 공개한다
응답 MPD + PUSH_POLICY
푸시-정책:SegmentIdx=+1
Data 1.1 요청
응답 Data 1.1 + PUSH_PROMISE
Data 2.1
Data 2.1 전송
PUSH_POLICY_REQUEST 전송
푸시-정책-요청: SegmentIdx=+3
1805
OK 200
1806
Data 3.1 요청
1807
응답 Data 3.1 + PUSH_PROMISE
Data 4.1
Data 5.1
Data 6.1
1808

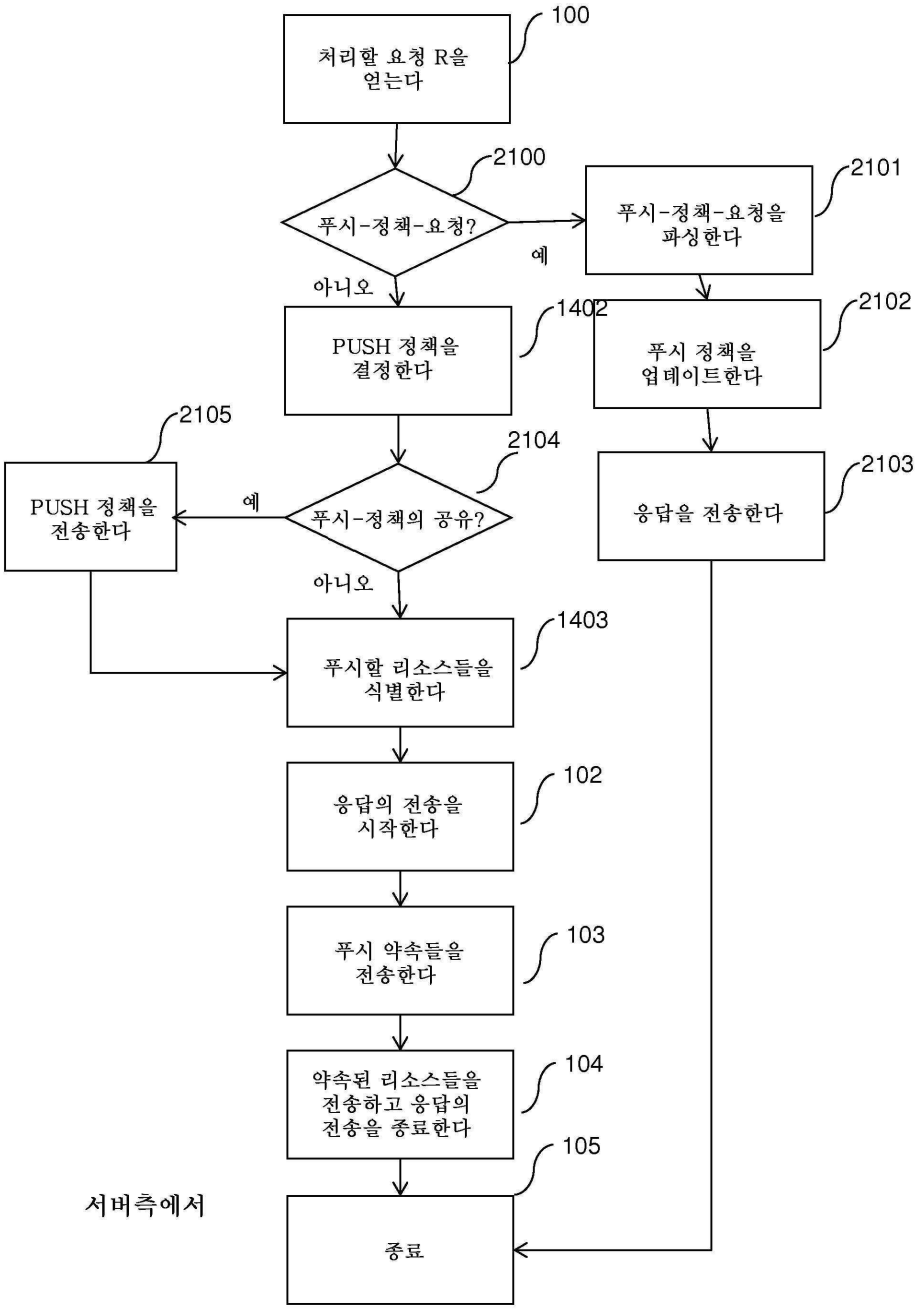
도면19


1900
<MPD ...>
<PushPolicy segmentIdx="+3"/>
<Period>
<AdaptationSet id="AS1" mimeType='video/mp4' codecs='hev1' >
<Representation id='R0' width='1920' height='1080' frameRate='30'... ...bandwidth='256000'>
<BaseURL>http://myserver.com/media.mp4</BaseURL>
<SegmentList>
1901
<InitializationRange="0-999" />
<SegmentURL mediaRange="1000-4999" indexRange="1000-2000"/>
<SegmentURL mediaRange="5000-9999" indexRange="5000-7000"/>
<SegmentURL mediaRange="10000-14999" indexRange="10000-12000"/>
<SegmentURL mediaRange="15000-19999" indexRange="15000-17000"/>
...
</SegmentList>
</Representation>
</AdaptationSet>
</Period>
<Period>
...
</Period>
</MPD>
1902
PUSH_PROMISE
:method = GET
:host: myserver.com
:path = /media.mp4
:if-match: "my-etag1"
Accept: video/mp4
Range:bytes=1000-4999
PUSH_PROMISE
:method = GET
:host: myserver.com
:path = /media.mp4
:if-match: "my-etag2"
Accept: video/mp4
Range:bytes=5000-9999
PUSH_PROMISE
:method = GET
:host: myserver.com
:path = /media.mp4
:if-match: "my-etag3"
Accept: video/mp4
Range:bytes=10000-14999
PUSH_PROMISE
:method = GET
:host: myserver.com
:path = /media.mp4
:if-match: "my-etag4"
Accept: video/mp4
Range:bytes=1000-14999
1903
1904
PUSH_PROMISE
:method = MGET
:host: example.org
:path = /video/seg2.mp4,
/video/seg3.mp4, /video/seg4.mp4
:if-match: "my-etag1"
accept: video/mp4
PUSH_PROMISE
:method = GET
:host: example.org
:path = /video/segments.mp4
:if-match: "my-etag1"
Accept: video/mp4
Range:bytes=123-456,789-1223, 4567-6546
1905

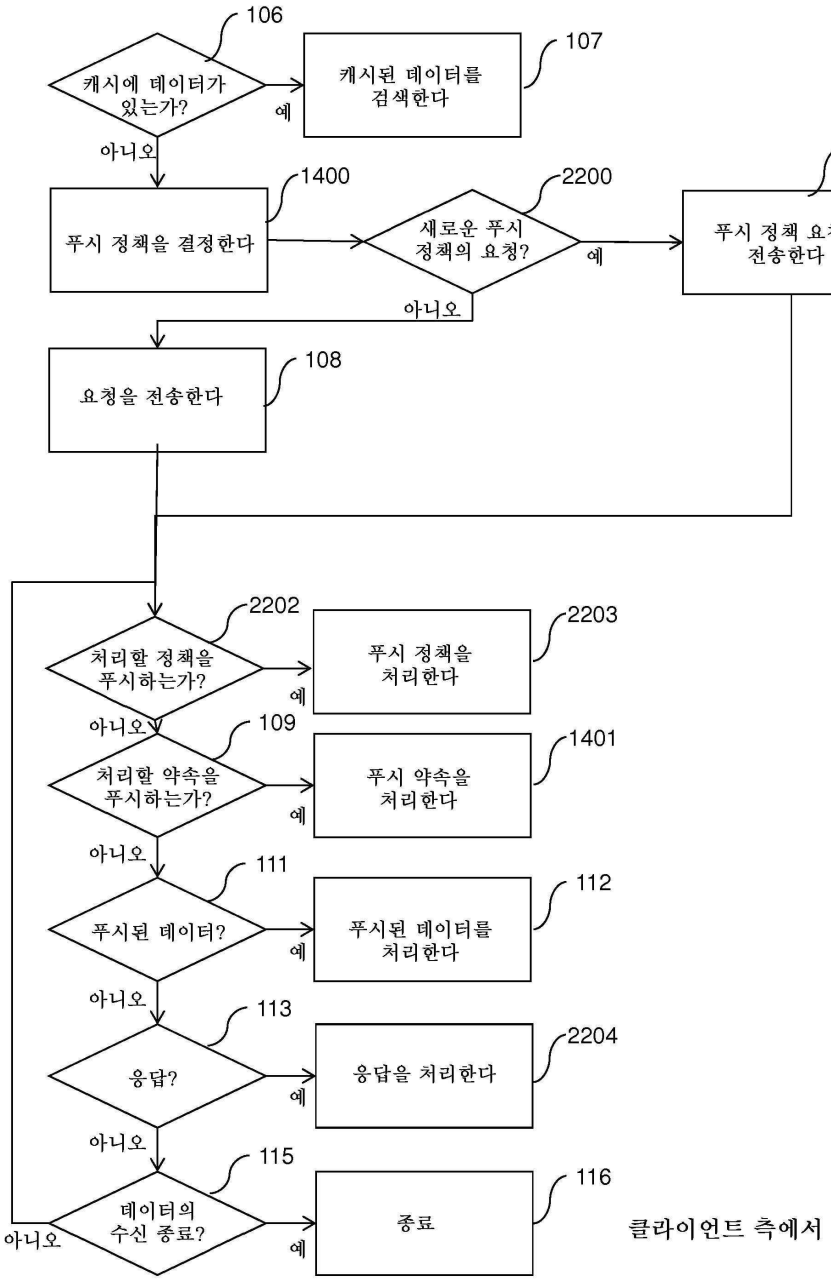
도면20


처리할 요청 R을 얻는다
100
PUSH 정책을 결정한다
1402
푸시할 리소스를 식별한다
1403
응답의 전송을 시작한다
102
푸시 약속을 처리한다
2000
푸시 약속을 전송한다
103
약속된 리소스들을 전송하고 응답의 전송을 종료한다
104
종료
105

도면21


처리할 요청 R을 얻는다
100
푸시-정책-요청?
2100
예
푸시-정책-요청을 파싱한다
2101
아니오
PUSH 정책을 결정한다
1402
푸시 정책을 업데이트한다
2102
PUSH 정책을 전송한다
2105
예
푸시-정책의 공유?
2104
응답을 전송한다
2103
아니오
푸시할 리소스들을 식별한다
1403
응답의 전송을 시작한다
102
푸시 약속들을 전송한다
103
약속된 리소스들을 전송하고 응답의 전송을 종료한다
104
종료
105
서버측에서

도면22


106
캐시에 데이터가 있는가?
예
107
캐시된 데이터를 검색한다
아니오
1400
푸시 정책을 결정한다
2200
새로운 푸시 정책의 요청?
예
2201
푸시 정책 요청을 전송한다
아니오
108
요청을 전송한다
2202
처리할 정책을 푸시하는가?
예
2203
푸시 정책을 처리한다
아니오
109
처리할 약속을 푸시하는가?
예
1401
푸시 약속을 처리한다
아니오
111
푸시된 데이터?
예
112
푸시된 데이터를 처리한다
아니오
113
응답?
예
2204
응답을 처리한다
아니오
115
데이터의 수신 종료?
아니오
예
116
종료
클라이언트 측에서

도면23

```
<MPD …>
  <Period>
  <BaseURL>http://myserver.com/media/</BaseURL>
  <SegmentList>
   <Initialization sourceURL=« URL_SI »/>
  </SegmentList>
  <AdaptationSet id="AS1" mimeType='video/mp4' codecs='hev1' >
   <Representation id='R0' width='1920' height='1080' frameRate='30'… …bandwidth='256000'>
     <SupplementalProperty schemeIdUri='urn:mpeg:dash:push_policy' value = 'next:3'/>
    <Role schemeIdUri="urn:mpeg:DASH:role:2011" value="main"/>
    <SegmentList duration='10'>
     <SegmentURL media='seg-full-AS1R0-1.mp4'/>
     <SegmentURL media='seg-full-AS1R0-2.mp4'/>
     <SegmentURL media='seg-full-AS1R0-3.mp4'/>
     …
    </SegmentList>
   </Representation>

   <Representation id='R1' width='1920' height='1080' frameRate='30'… …bandwidth='2256000'>
   <SupplementalProperty schemeIdUri='urn:mpeg:dash:push_policy' value = 'next:1'/>
    <Role schemeIdUri="urn:mpeg:DASH:role:2011" value="associate"/>
    <SegmentList duration='10'>
     <SegmentURL media='seg-full-AS1R1-1.mp4'/>
     <SegmentURL media='seg-full-AS1R1-2.mp4'/>
     <SegmentURL media='seg-full-AS1R1-3.mp4'/>
     …
    </SegmentList>
   </Representation>
  </AdaptationSet>

  <AdaptationSet id='AS2' mimeType='video/mp4' codecs='hev1' >
   <Representation id='R1' width='2000' height='1000' bandwidth='512000'>
    <SegmentList>
     …
    </SegmentList>
   </Representation>
  </AdaptationSet>
 </Period>
 <Period>
  …
 </Period>
</MPD>
```

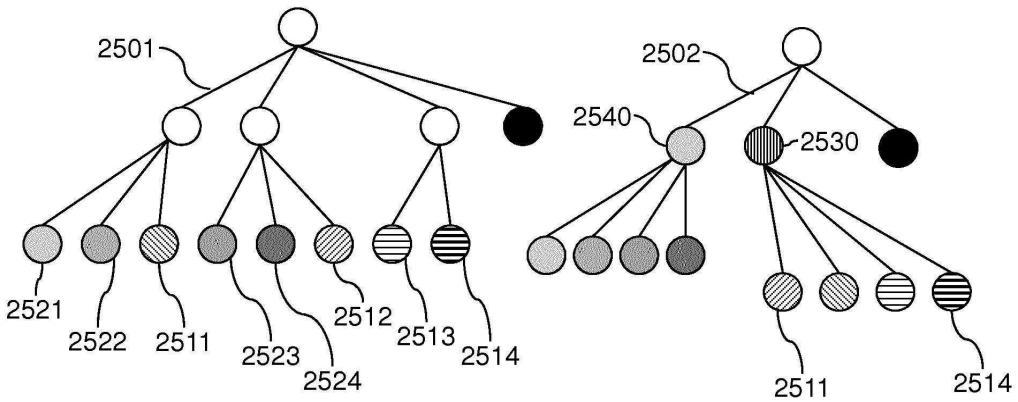
2300

2301

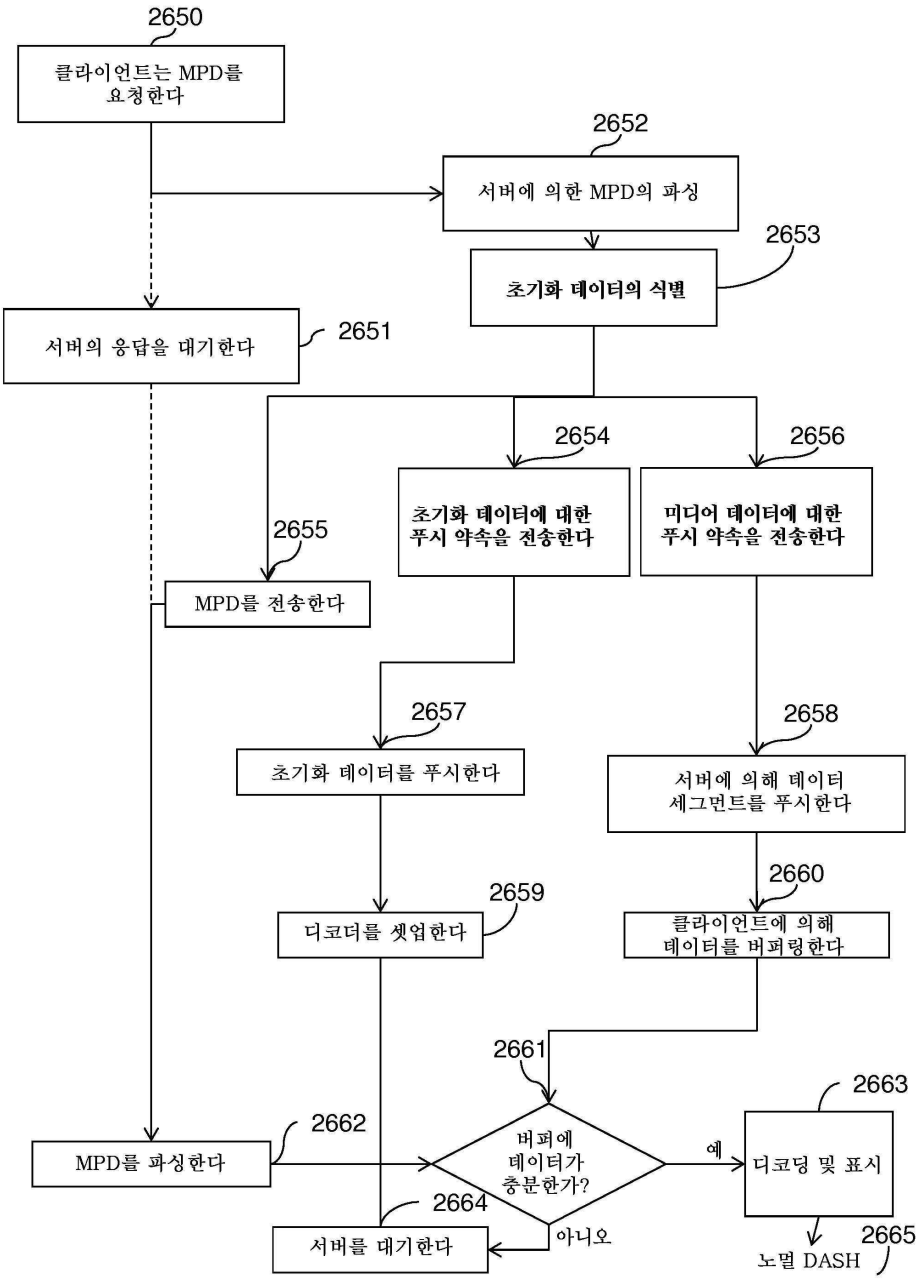
도면24

```
<MPD …>
 <Period id='1'>
   <BaseURL>http://myserver.com/media/</BaseURL>
   <SegmentList>
    <Initialization sourceURL=« URL_SI »/>
   </SegmentList>
   <AdaptationSet id="AS1" mimeType='video/mp4' codecs='hev1' >
    <Representation id='R0' width='1920' height='1080' frameRate='30'… …bandwidth='256000'>
     <Role schemeIdUri="urn:mpeg:DASH:role:2011" value="main"/>
     <SegmentList duration='10'>                                          2401, 2400
      <SegmentURL media='seg-full-AS1R0-1.mp4' />
      <SegmentURL media='seg-full-AS1R0-2.mp4' />                         2405
      <SegmentURL media='seg-full-AS1R0-3.mp4' />
     </SegmentList>
  </Period id ='2'>
    <BaseURL>http://myserver.com/media/</BaseURL>
    <SegmentList>
     <Initialization sourceURL=« URL_SI »/>
    </SegmentList>
    <AdaptationSet id='AS1' mimeType='video/mp4' codecs='hev1' >
     <Representation id='R0' width='1920' height='1080' frameRate='30'… …bandwidth='256000'>
      <Role schemeIdUri='urn:mpeg:DASH:role:2011' value='main'/>          2402
      <SegmentList duration='10'>                                         2403
       <SegmentURL media='seg-full-AS1R0-1.mp4' />                        2404
       <SegmentURL media='seg-full-AS1R0-2.mp4' />
       <SegmentURL media='seg-full-AS1R0-3.mp4' />
     </SegmentList>
   <Period>
     …
    </Period>
   </MPD>
```

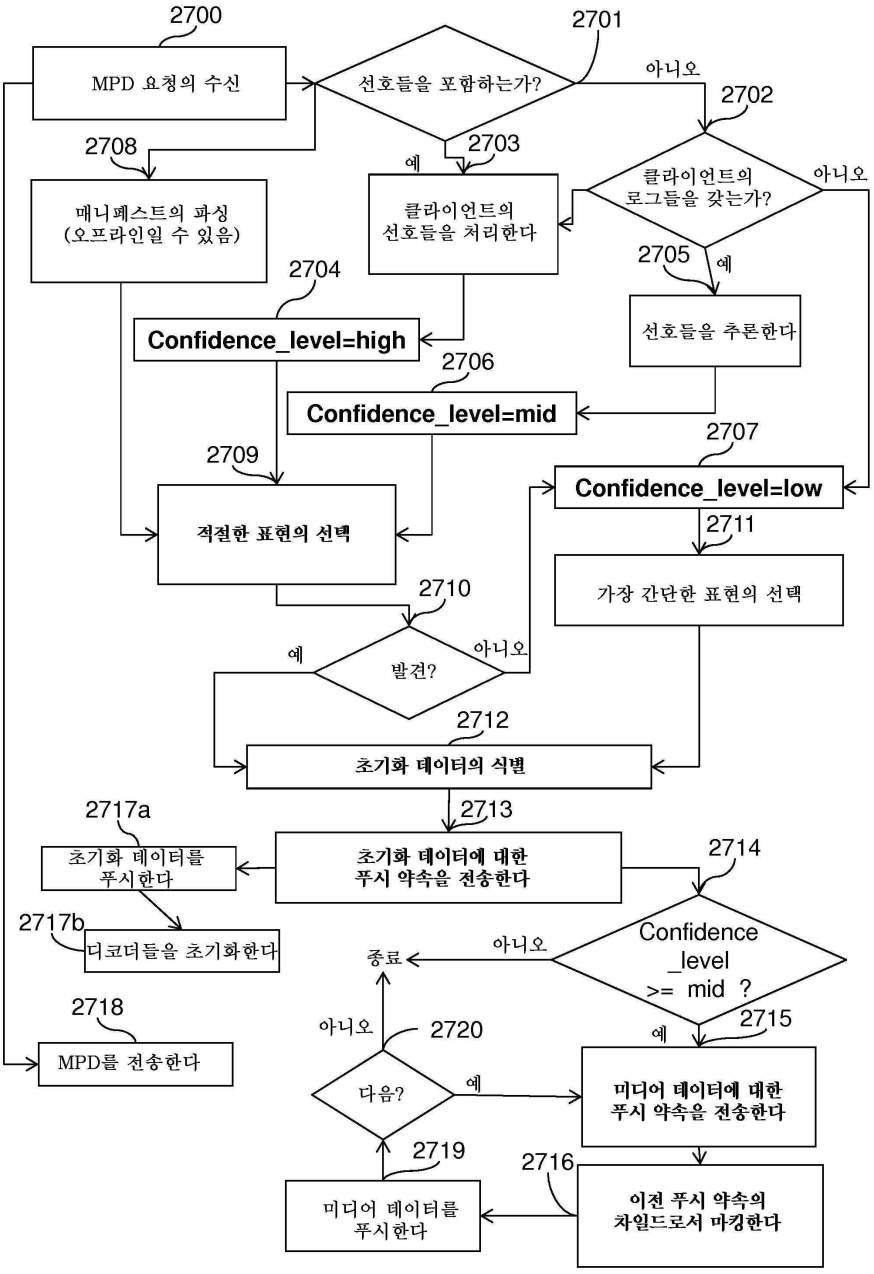
도면25



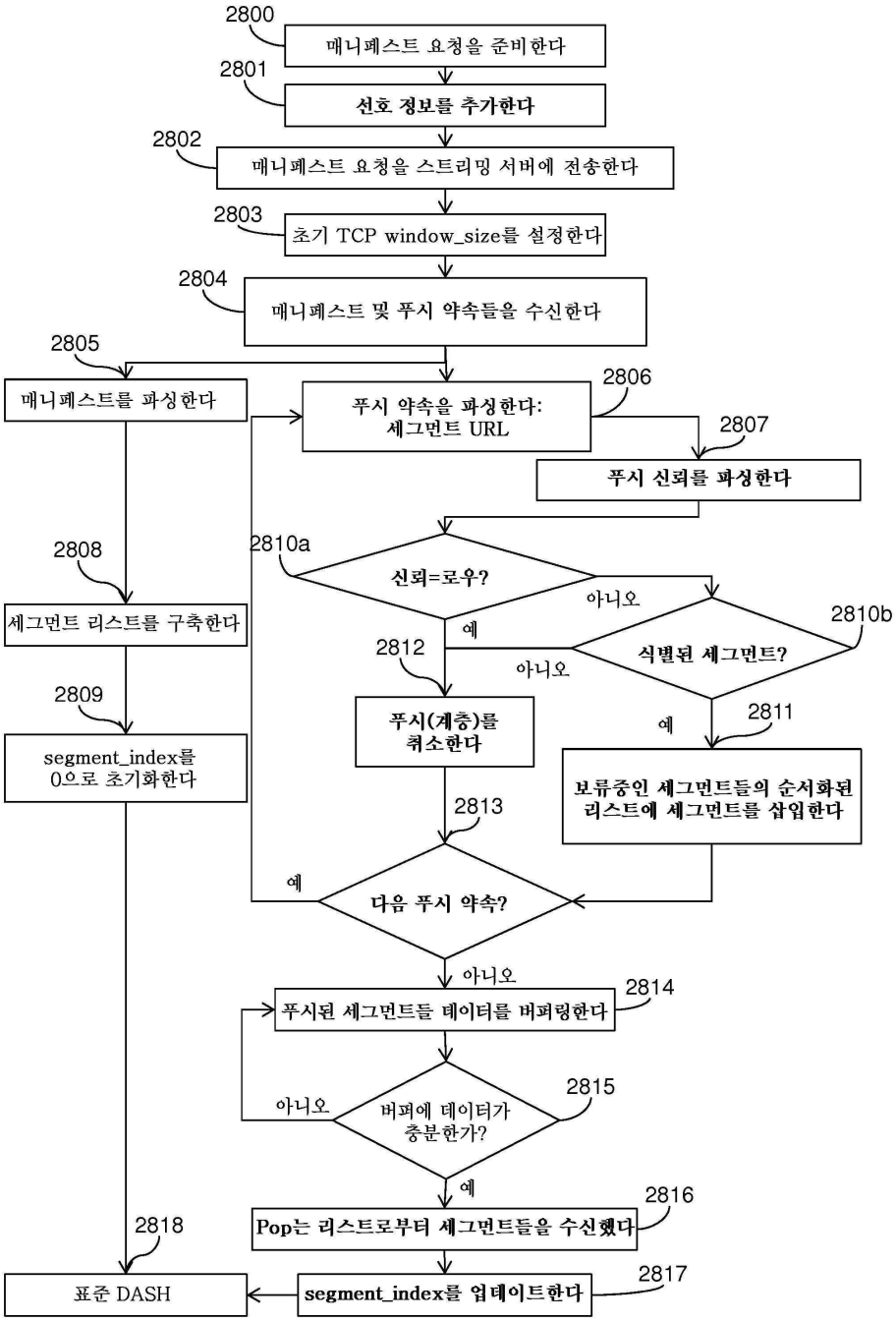
도면26


2650
클라이언트는 MPD를 요청한다
2652
서버에 의한 MPD의 파싱
2653
초기화 데이터의 식별
서버의 응답을 대기한다
2651
2654
초기화 데이터에 대한 푸시 약속을 전송한다
2656
미디어 데이터에 대한 푸시 약속을 전송한다
2655
MPD를 전송한다
2657
초기화 데이터를 푸시한다
2658
서버에 의해 데이터 세그먼트를 푸시한다
2659
디코더를 셋업한다
2660
클라이언트에 의해 데이터를 버퍼링한다
2661
2662
MPD를 파싱한다
버퍼에 데이터가 충분한가?
예
2663
디코딩 및 표시
2664
서버를 대기한다
아니오
2665
노멀 DASH

도면27


2700
MPD 요청의 수신
2701
선호들을 포함하는가?
아니오
2702
예
2703
2708
매니페스트의 파싱
(오프라인일 수 있음)
클라이언트의
선호들을 처리한다
클라이언트의
로그들을 갖는가?
아니오
2705
예
선호들을 추론한다
2704
Confidence_level=high
2706
Confidence_level=mid
2707
Confidence_level=low
2709
적절한 표현의 선택
2711
가장 간단한 표현의 선택
2710
예
발견?
아니오
2712
초기화 데이터의 식별
2713
2717a
초기화 데이터를
푸시한다
초기화 데이터에 대한
푸시 약속을 전송한다
2714
2717b
디코더들을 초기화한다
Confidence
_level
>= mid ?
아니오
종료
2715
예
2718
MPD를 전송한다
아니오
2720
다음?
예
미디어 데이터에 대한
푸시 약속을 전송한다
2719
2716
미디어 데이터를
푸시한다
이전 푸시 약속의
차일드로서 마킹한다

도면28


2800
매니페스트 요청을 준비한다
2801
선호 정보를 추가한다
2802
매니페스트 요청을 스트리밍 서버에 전송한다
2803
초기 TCP window_size를 설정한다
2804
매니페스트 및 푸시 약속들을 수신한다
2805
매니페스트를 파싱한다
2806
푸시 약속을 파싱한다:
세그먼트 URL
2807
푸시 신뢰를 파싱한다
2808
세그먼트 리스트를 구축한다
2809
segment_index를
0으로 초기화한다
2810a
신뢰=로우?
아니오
예
2810b
식별된 세그먼트?
아니오
예
2811
보류중인 세그먼트들의 순서화된
리스트에 세그먼트를 삽입한다
2812
푸시(계층)를
취소한다
2813
다음 푸시 약속?
예
아니오
2814
푸시된 세그먼트들 데이터를 버퍼링한다
2815
버퍼에 데이터가
충분한가?
아니오
예
2816
Pop는 리스트로부터 세그먼트들을 수신했다
2817
segment_index를 업데이트한다
2818
표준 DASH