(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0069748
(43) 공개일자 2012년06월28일

(51) 국제특허분류(Int. Cl.)
H04L 29/06 (2006.01) *H04L 12/56* (2006.01)
(21) 출원번호 10-2012-7010473
(22) 출원일자(국제) 2010년09월22일
심사청구일자 2012년04월23일
(85) 번역문제출일자 2012년04월23일
(86) 국제출원번호 PCT/US2010/049862
(87) 국제공개번호 WO 2011/038028
국제공개일자 2011년03월31일
(30) 우선권주장
12/887,483 2010년09월21일 미국(US)
(뒷면에 계속)

(71) 출원인
콸콤 인코포레이티드
미국 캘리포니아 샌디에고 모어하우스 드라이브5775 (우 92121-1714)
(72) 발명자
루비, 마이클 지.
미국 95051 캘리포니아 산타클라라 키퍼 로드 3135
왓슨, 마크
미국 94114 캘리포니아 샌 프란시스코 앨버라도 스트리트 629
(뒷면에 계속)
(74) 대리인
남상선

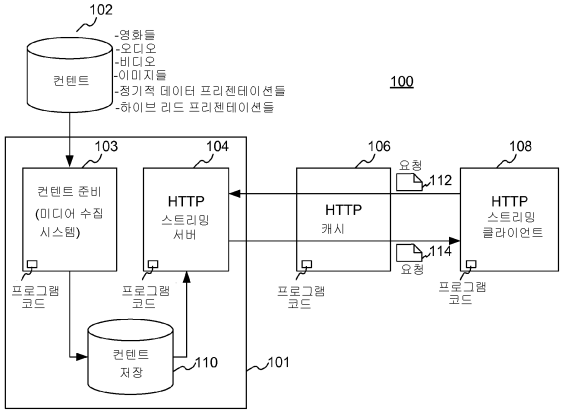
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 **개선된 클라이언트 측 핸들링을 위한 요청 제어들 또는 블록 분할을 이용하는 강화된 블록 요청 스트리밍**

(57) 요 약

블록-요청 스트림 시스템이, 통상적으로 종래 파일 서버(HTTP, FTP 등)에 의해 서빙될 형태로 데이터를 생성하는 수집 시스템을 사용하여, 이러한 시스템들의 대역폭 효율성 및 사용자 경험의 향상들을 제공하는데, 여기서 수집 시스템은 컨텐트를 획득하고 그 컨텐트를 파일 서버에 의해 제공될 파일들 또는 데이터 엘리먼트들로서 준비한다. 클라이언트 디바이스는 수집 프로세스를 이용하도록 적응될 수 있다. 클라이언트 디바이스는 수집 시스템으로부터 클라이언트 디바이스로 이용가능한 정보가 주어지면, 리소스들의 사용을 최적화하기 위해 구성될 수 있다. 이는 통계적 고려들에 기초한 전체 파일 요청들의 사용 및/또는 요청들의 유연한 파이프라이닝, 블록 요청들의 하부의 전달 접속들로의 맵핑, 가변 사이징된 요청들의 사용, 버퍼 크기의 변화의 레이트 및 버퍼 크기를 모니터링하는 것에 기초하여 블록 요청들의 구성, 타이밍 및 시퀀스를 결정하기 위한 구성을 포함할 수 있다.

대 표 도 - 도1



(72) 발명자

비치사노, 로렌조

미국 94708 캘리포니아 버클리 그리즐리 피크 블루버드 1070

파크자드, 파얌

미국 95051 캘리포니아 산타클라라 키퍼 로드 3135

왕, 빈

미국 94555 캘리포니아 프리몬트 바르돌프 써클 33640

(30) 우선권주장

61/244,767 2009년09월22일 미국(US)
61/257,719 2009년11월03일 미국(US)
61/258,088 2009년11월04일 미국(US)
61/285,779 2009년12월11일 미국(US)
61/296,725 2010년01월20일 미국(US)
61/372,399 2010년08월10일 미국(US)

특허청구의 범위

청구항 1

클라이언트 디바이스가 하나 또는 그 초과의 미디어 수집 시스템으로부터 미디어의 블록들을 수신하는 통신 시스템에서의 방법으로서,

상기 클라이언트 디바이스에서, 상기 클라이언트 디바이스의 출력에 의해 소모되기 전에 미디어의 수신된 블록들을 저장하기 위한 사용가능한 버퍼의 크기를 식별하는 단계;

상기 클라이언트 디바이스에서, 상기 클라이언트 디바이스의 출력에서 상기 미디어의 소모의 레이트를 식별하는 단계;

상기 사용가능한 버퍼의 크기의 변화의 레이트를 결정하는 단계; 및

상기 클라이언트 디바이스의 출력에 의해 소모될 상기 미디어의 블록들에 대응하는 다수의 블록 요청들에 대해, 상기 요청의 타이밍 및 요청에 대한 블록을 결정하는 단계를 포함하며,

요청된 특정 블록 및 상기 요청의 타이밍은 상기 사용가능한 버퍼의 상기 크기의 변화의 레이트 및 상기 사용가능한 버퍼의 크기에 의존하는,

미디어의 블록들을 수신하는 통신 시스템에서의 방법.

청구항 2

클라이언트 디바이스가 하나 또는 그 초과의 미디어 수집 시스템으로부터 미디어의 블록들을 수신하는 통신 시스템에서의 방법으로서,

상기 클라이언트 디바이스에서, 수신된 블록들이 상기 클라이언트 디바이스의 출력에 의해 소모되기 전에 미디어의 수신된 블록들을 저장하기 위해 사용되는 버퍼의 양을 식별하는 단계;

상기 클라이언트 디바이스에서, 상기 저장된 수신 블록들의 소모의 레이트를 결정하는 단계;

사용되는 상기 버퍼의 양 및 소모의 레이트에 기초하여, 미디어 수집 시스템에 대해 행하기 위한 요청의 크기를 결정하는 단계; 및

상기 미디어 수집 시스템에 상기 요청을 발행하는 단계를 포함하는,

미디어의 블록들을 수신하는 통신 시스템에서의 방법.

청구항 3

클라이언트 디바이스가 미디어 수집 시스템으로부터 미디어 데이터를 요청하는 통신 시스템에서의 방법으로서,

상기 클라이언트 디바이스에서, 이전의 요청에 대해 상기 미디어 수집 시스템으로부터 수신되는 바이트들의 수를 결정하는 단계; 및

수신된 바이트들의 수의 테스트에 기초하여 상기 미디어 수집 시스템으로부터 추가의 미디어 데이터를 요청하는 단계를 포함하는,

미디어 데이터를 요청하는 통신 시스템에서의 방법.

청구항 4

클라이언트 디바이스가 하나 또는 그 초과의 서버들로부터 미디어 파일들을 요청하는 통신 시스템에서의 방법으로서,

상기 클라이언트 디바이스에서, 요청될 상기 미디어 파일의 식별자를 결정하는 단계;

상기 서버들의 IP 어드레스들의 세트를 결정하는 단계;

상기 식별자를 포함하는 계산에 기초하여 상기 IP 어드레스들을 오더링하는 단계; 및

상기 오더링에 기초하여 서버를 선택하는 단계를 포함하는,

미디어 파일들을 요청하는 통신 시스템에서의 방법.

청구항 5

클라이언트 디바이스가 미디어 수집 시스템에 저장된 파일로부터 미디어의 블록들을 수신하는 통신 시스템에서의 방법으로서,

상기 클라이언트 디바이스에서, 상기 미디어 수집 시스템으로부터 요청될 블록들의 범위를 식별하는 단계;

블록들의 범위가 상기 파일의 제1 블록을 포함하는지를 결정하는 단계;

통계적 파라미터를 계산하는 단계; 및

상기 통계적 파라미터에 기초하여 그리고 상기 결정에 기초하여, 블록들의 범위에 대한 상기 요청을 상기 파일에 대한 요청으로 변환하는 단계를 포함하는,

미디어의 블록들을 수신하는 통신 시스템에서의 방법.

청구항 6

제 5 항에 있어서,

상기 통계적 파라미터는 임계치에 대응하는, 미디어의 블록들을 수신하는 통신 시스템에서의 방법.

청구항 7

제 5 항에 있어서,

상기 통계적 파라미터는 블록들의 상기 범위에 기초하는 해쉬 함수의 출력에 대응하는, 미디어의 블록들을 수신하는 통신 시스템에서의 방법.

청구항 8

제 5 항에 있어서,

다수의 수신기들로부터 상기 파일에 대한 요청들의 원하는 비율이 변환되도록 임계치 및 값이 설정되는, 미디어의 블록들을 수신하는 통신 시스템에서의 방법.

명 세 서

기 술 분 야

[0001] 본 출원은, 각각 Michael G. Luby 등에 의해 출원되고 각각 "Enhanced Block-Request Streaming System"으로 명명된 하기 가출원들에 대해 우선권의 이익을 주장하는 정식(nonprovisional) 특허출원이다.

[0002] 2009년 9월 22일자로 출원된 미국 가특허출원 제 61/244,767호,

[0003] 2009년 11월 3일자로 출원된 미국 가특허출원 제 61/257,719호,

[0004] 2009년 11월 4일자로 출원된 미국 가특허출원 제 61/258,088호,

[0005] 2009년 12월 11일자로 출원된 미국 가특허출원 제 61/285,779호, 및

[0006] 2010년 1월 20일자로 출원된 미국 가특허출원 제 61/296,725호.

[0007] 본 출원은 또한 2010년 8월 10일자로 Ying Chen 등에 의해 출원되고 "HTTP Streaming Extensions"으로 명명된 미국 가특허출원 제 61/372,399호에 대해 우선권의 이익을 주장한다.

[0008] 상기 인용된 각각의 가출원은 모든 목적들을 위해 본 명세서에 참조로 통합되었다. 본 개시는 또한, 모든 목적들을 위해 본 명세서에서 완전히 기술된 것처럼, 하기 공통 양도된 출원들/특허들을 참조로 통합한다.

[0009] Luby에게 특허된 미국 특허 제 6,307,487호(이하, "Luby I");

[0010] Shokrollahi 등에게 특허된 미국 특허 제 7,068,729호(이하, "Shokrollahi I");

[0011] 2006년 6월 9일자로 Luby 등에 의해 출원되고, "Forward Error-Correcting (FEC) Coding and Streaming"으로 명명된 미국 특허 출원 제 11/423,391호(이하, "Luby II");

[0012] 2008년 4월 15일자로 Luby 등에 의해 출원되고, "Dynamic Stream Interleaving and Sub-Stream Based Delivery"으로 명명된 미국 특허 출원 제 12/103,605호(이하, "Luby III");

[0013] 2010년 2월 12일자로 Pakzad 등에 의해 출원되고, "Block Partitioning for a Data Stream"으로 명명된 미국 특허 출원 제 12/705,202호(이하, "Pakzad"); 및

[0014] 2010년 8월 18일자로 Luby 등에 의해 출원되고, "Methods and Apparatus Employing FEC Codes with Permanent Inactivation of Symbols for Encoding and Decoding Processes"으로 명명된 미국 특허 출원 제 12/859,161호(이하, "Luby IV").

[0015] 본 발명은 개선된 미디어 스트리밍 시스템들 및 방법들에 관한 것이고, 더 상세하게는, 스트리밍된 미디어의 프리젠테이션을 최적화하고, 스트리밍된 미디어 데이터의 효율적인 동시 전달 또는 적시에 분배된 전달을 허용하기 위해, 네트워크 및 버퍼 조건들에 적응적인 시스템들 및 방법들에 관한 것이다.

배 경 기 술

[0016] 스트리밍 미디어 전달은, 인터넷과 같은 패킷 기반 네트워크들, 셀룰러 및 무선 네트워크들, 전력선 네트워크들 및 다른 유형들의 네트워크들을 통해 전달될 고품질 오디오 및 비디오에 더욱 통상적이 됨에 따라 점차 중요해지고 있다. 전달되는 스트리밍 미디어에 제공될 수 있는 품질은, 본래 컨텐트의 해상도(또는 다른 속성들), 본래 컨텐트의 인코딩 품질, 미디어를 디코딩 및 제공하는 수신 디바이스들의 성능들, 수신기들에서 수신된 신호의 적시성 및 품질 등을 포함하는 다수의 팩터들에 의존할 수 있다. 인지된 양호한 스트리밍 미디어 경험을 생성하기 위해, 수신기들에서 수신된 신호의 전송 및 적시성은 특히 중요할 수 있다. 양호한 전송은, 전송기가 전송한 스트림에 대한 수신기에서 수신된 스트림의 신뢰도를 제공할 수 있는 한편, 적시성은, 컨텐트에 대한 초기 요청 이후 수신기가 얼마나 신속하게 그 컨텐트의 플레이아웃을 시작할 수 있는지를 표현할 수 있다.

[0017] 미디어 전달 시스템은 미디어 소스들, 미디어 목적지들, 및 소스들과 목적지들을 분리시키는 (시간 및/또는 공간에서의) 채널들을 갖는 시스템으로 특징지어질 수 있다. 전형적으로, 소스는 전자적으로 관리가능한 형태의 미디어로의 액세스를 갖는 전송기, 미디어(또는 이들의 근사치)의 수신을 전자적으로 제어하는 능력을 갖는 수신기를 포함하고, 이를 미디어 소비자(예를 들어, 몇몇 방식으로 그 수신기, 저장 디바이스 또는 엘리먼트, 다른 채널 등에 커플링된 디스플레이 디바이스를 갖는 사용자)에게 제공한다.

[0018] 다수의 변형들이 가능하지만, 통상적인 예에서, 미디어 전달 시스템은, 전자적 형태의 미디어 컨텐트로의 액세스를 갖는 하나 또는 그 초과의 서버들을 갖고, 하나 또는 그 초과의 클라이언트 시스템들 또는 디바이스들은 서버들에 미디어를 요청하고, 서버들은 서버의 일부로서의 전송기를 이용하여 미디어를 전달하고, 클라이언트의 수신기로 전송하여, 수신된 미디어가 몇몇 방식으로 클라이언트에서 소비될 수 있다. 간단한 예에서, 소정의 요청 및 응답에 대해 하나의 서버 및 하나의 클라이언트가 존재할 수 있지만, 이것은 반드시 필요한 것은 아니다.

[0019] 종래에, 미디어 전달 시스템들은 "다운로드" 모델 또는 "스트리밍" 모델로 특징지어질 수 있다. "다운로드" 모델은 미디어 데이터의 전달과 사용자 또는 수신 디바이스로의 미디어의 플레이아웃과의 사이에서의 타이밍 독립성에 의해 특징지어질 수 있다.

[0020] 일예로, 미디어는 미디어가 요구되거나 이용될 시점보다 충분히 미리 다운로드되고, 미디어가 이용될 때 수신기에서 요구되는 만큼 충분히 이미 이용가능하다. 다운로드 콘텍스트에서의 전달은 종종 HTTP, FTP 또는 FLUTE(File Delivery over Unidirectional Trnasport)와 같은 파일 전송 프로토콜을 이용하여 수행되고, 전달 레이트는 TCP/IP와 같은 내재하는 플로우 및/또는 혼잡 제어 프로토콜에 의해 결정될 수 있다. 플로우 또는 혼잡 제어 프로토콜의 동작은 사용자 또는 목적지 디바이스로의 미디어의 플레이아웃에 독립적일 수 있고, 이것은 다운로드와 동시에 또는 몇몇 다른 시간에 발생할 수 있다.

[0021] "스트리밍" 모드는 미디어 데이터의 전달 타이밍과 사용자 또는 수신기 디바이스로의 미디어의 플레이아웃과의 사이에서의 엄격한 커플링에 의해 특징지어질 수 있다. 이러한 콘텍스트에서의 전달은 종종, 제어를 위한 실시간 스트리밍 프로토콜(RTSP) 및 미디어 데이터를 위한 실시간 전송 프로토콜(RTP)와 같은 스트리밍 프로

토콜을 이용하여 수행된다. 전달 레이트는, 데이터의 플레이아웃 레이트에 종종 매칭하는 스트리밍 서버에 의해 결정될 수 있다.

[0022] "다운로드" 모델의 몇몇 단점들은, 전달 및 플레이아웃의 타이밍 독립성에 기인하여, 플레이아웃이 요구될 때 미디어 데이터가 이용가능하지 못할 수 있어서(예를 들어, 이용가능한 대역폭이 미디어 데이터 레이트 미만인 것에 기인함) 플레이아웃을 일시적으로 중지하게 하여("스톨링(stalling)") 나쁜 사용자 경험을 초래할 수 있다는 점, 또는 미디어 데이터가 플레이아웃에 앞서 너무 일찍 다운로드되도록 요구될 수 있어서(예를 들어, 이용가능한 대역폭이 미디어 데이터 레이트보다 큰 것에 기인함), 수신 디바이스 상의 희박할 수 있는 저장 자원들을 소비하고, 결국 컨텐트가 플레이아웃되지 않거나 달리 이용되지 않으면 낭비될 수 있는, 전달을 위한 소중한 네트워크 자원들을 소비할 수 있다는 점일 수 있다.

[0023] "다운로드" 모델의 이점은, 이러한 다운로드들을 수행하기 위해 필요한, 예를 들어, HTTP와 같은 기술이 매우 발전되었고 널리 배치되었으며, 광범위한 애플리케이션들에 걸쳐 적용가능하다는 점일 수 있다. 이러한 파일 다운로드들의 큰 확장성을 위한 다운로드 서버들 및 솔루션들(예를 들어, HTTP 웹 서버들 및 컨텐트 전달 네트워크들)은 쉽게 이용가능할 수 있어서, 이러한 기술에 기반한 서비스들의 전개를 간단하고 저렴하게 할 수 있다.

[0024] "스트리밍" 모델의 몇몇 단점들은, 일반적으로 미디어 데이터의 전달 레이트가 서버로부터 클라이언트로의 접속에 대해 이용가능한 대역폭에 적응적이지 않다는 점, 및 대역폭 및 지연 보장을 제공하는 더 복잡한 네트워크 아키텍쳐 또는 특수한 스트리밍 서버들이 요구된다는 점일 수 있다. 이용가능한 대역폭에 따라 전달 데이터 레이트의 변형을 지원하는 스트리밍 시스템들(예를 들어, Adobe 플래시 적응형 스트리밍)이 존재하지만, 일반적으로, 모든 이용가능한 대역폭을 활용함에 있어서 TCP와 같은 다운로드 전송 플로우 제어 프로토콜들만큼 효율적이지 않다.

[0025] 최근에, "스트리밍" 및 "다운로드" 모델들의 조합에 기반한 새로운 미디어 전달 시스템들이 개발되고 전개되고 있다. 이러한 모델의 일예가 본 명세서에서 "블록-요청 스트리밍" 모델로 지칭되며, 여기서, 미디어 클라이언트는 HTTP와 같은 다운로드 프로토콜을 이용하여 서빙 인프라구조로부터 미디어 데이터의 블록들을 요청한다. 이러한 시스템들에서의 관심사는 스트림의 플레이아웃을 시작하는 능력, 예를 들어, 수신된 오디오 및 비디오 스트림들을 개인용 컴퓨터를 이용하여 디코딩 및 렌더링하고, 비디오를 컴퓨터 스크린 상에 디스플레이하고, 내장된 스피커들을 통해 오디오를 플레이하는 능력, 또는 다른 예로, 수신된 오디오 및 비디오 스트림들을 셋탑 박스를 이용하여 디코딩 및 렌더링하고, 비디오를 텔레비젼 디스플레이 디바이스 상에 디스플레이하고, 스테레오 시스템을 통해 오디오를 플레이하는 능력일 수 있다.

[0026] 소스 스트리밍 레이트에 뒤쳐지지 않고, 디코딩 레이턴시를 최소화하고, 이용가능한 CPU 자원들의 이용을 감소시키기 위해, 소스 블록들을 충분히 빨리 디코딩할 수 있는 것과 같은 다른 관심사들이 쟁점이 된다. 또 다른 관심사는, 수신기들에 전달된 스트림들의 품질에 악영향을 주지 않으면서 시스템의 컴포넌트들이 실패하도록 허용하는 견고하고 스케일가능한 스트리밍 전달 솔루션을 제공하는 것이다. 다른 문제점들은 분배되고 있는 프리젠테이션에 대한 정보를 급격하게 변경하는데 기초하여 발생할 수 있다. 따라서, 개선된 프로세스들 및 장치들을 갖는 것이 바람직하다.

발명의 내용

[0027] 블록-요청 스트리밍 시스템은, 전형적으로, 종래의 파일 서버에 의해 서빙될 형태(HTTP, FTP 등)로 데이터를 생성하는 수집(ingestion) 시스템을 이용하여, 블록-요청 스트리밍 시스템들의 대역폭 효율 및 사용자 경험에서의 개선들을 제공하며, 수집 시스템은 컨텐트를 접수하고(intake), 이를, 캐시를 포함하거나 포함하지 않을 수 있는 파일 서버에 의해 서빙될 파일들 또는 데이터 엘리먼트들로 준비한다. 클라이언트 디바이스는 수집 프로세스를 이용하도록 적응될 수 있을 뿐만 아니라 수집 프로세스와는 독립적인 더 양호한 프리젠테이션을 행하기 위한 개선들을 포함한다. 클라이언트 디바이스는 수집 시스템으로부터 클라이언트 디바이스로 이용가능한 정보가 주어지면, 리소스들의 사용을 최적화하기 위해 구성될 수 있다. 이는 버퍼 크기의 변화의 레이트 및 버퍼 크기를 모니터링하는 것에 기초하여 진보된 버퍼 관리 기술들에 기초하여 블록 요청들의 구성, 타이밍 및 시퀀스를 결정하기 위한 구성을 포함할 수 있다.

[0028] 일부 실시예들에서, 버퍼의 상태와 같은 조건들에 기초하여 가변 사이징된 요청들을 행하기 위한 방법들에 대한 신규한 개선들이 제공된다. 일부 실시예들에서, 요청들의 유연한 파이프라이닝을 포함하는 하부의 파일 다운로드 프로토콜 전달 접속들로의 블록 요청들의 맵핑을 위한 방법들에 대한 신규한 개선들이 제공된다.

일부 실시예들에서, 서버 선택의 신규한 방법들뿐만 아니라 통계적 고려들에 기초하여 전체 파일 요청들의 사용이 제공된다.

[0029] 첨부된 도면들과 함께 하기의 상세한 설명은 본 발명의 성질 및 이점들의 더 양호한 이해를 제공할 것이다.

도면의 간단한 설명

[0030] 도 1은 본 발명의 실시예들에 따른 블록-요청 스트리밍 시스템의 엘리먼트들을 도시한다.

도 2는, 컨텐트 수집 시스템에 의해 프로세싱되는 데이터를 수신하기 위해 블록 서빙 인프라구조("BSI")에 커플링되는 클라이언트 시스템의 엘리먼트들을 더 상세히 나타내는, 도 1의 블록-요청 스트리밍 시스템을 도시한다.

도 3은 수집 시스템의 하드웨어/소프트웨어 구현을 도시한다.

도 4는 클라이언트 시스템의 하드웨어/소프트웨어 구현을 도시한다.

도 5는, 세그먼트들 및 미디어 프리젠테이션 디스크립터("MPD") 파일들, 및 MPD 파일 내의 세그먼트들, 타이밍 및 다른 구조의 분해를 포함하는, 도 1에 나타낸 컨텐트 저장부의 가능한 구조들을 도시한다.

도 6은 도 1 및 도 5에 도시된 컨텐트 저장부에 저장될 수 있는 전형적인 소스 세그먼트의 상세들을 도시한다.

도 7a 및 도 7b는 파일들 내의 간단하고 계층적인 인덱싱을 도시한다.

도 8(a)는 미디어 스트림의 복수의 버전들에 대한 정렬된 탐색 포인트들을 갖는 가변적 블록 사이즈 설정을 도시한다.

도 8(b)는 미디어 스트림의 복수의 버전들에 대한 비정렬된 탐색 포인트들을 갖는 가변적 블록 사이즈 설정을 도시한다.

도 9(a)는 메타데이터 테이블을 도시한다.

도 9(b)는 서버로부터 클라이언트로의 블록들 및 메타데이터 테이블의 전송을 도시한다.

도 10은 RAP 경계들에 독립적인 블록들을 도시한다.

도 11은 세그먼트들에 걸친 연속적 및 비연속적 타이밍을 도시한다.

도 12는 스케일가능한 블록들의 일 양상을 도시한다.

도 13은 블록-요청 스트리밍 시스템 내의 특정한 변수들의 시간에 따른 진화에 대한 도식적 리프리젠테이션을 도시한다.

도 14는 블록-요청 스트리밍 시스템 내의 특정한 변수들의 시간에 따른 진화에 대한 다른 도식적 리프리젠테이션을 도시한다.

도 15는 상태(state)들의 셀 그리드를 임계값들의 함수로서 도시한다.

도 16은 요청 당 단일한 블록들 및 다수의 블록들을 요청할 수 있는 수신기에서 수행될 수 있는 프로세스의 흐름도이다.

도 17은 플렉서블 파이프라인 프로세스의 흐름도이다.

도 18은 요청들의 후보 세트, 이들의 우선순위들, 및 특정 시간에 이들이 어떤 접속들에 대해 발행될 수 있는지에 대한 일예를 도시한다.

도 19는 요청들의 후보 세트, 이들의 우선순위들, 및 이들이 어떤 접속들에 대해 발행될 수 있는지에 대한 일예를 도시하며, 이것은 시간에 따라 진화한다.

도 20은 파일 식별자에 기초한 일정한 캐싱 서버 프록시 선택의 흐름도이다.

도 21은 적절한 표현 언어에 대한 신택스 정의를 도시한다.

도 22는 적절한 해시 함수의 일예를 도시한다.

도 23은 파일 식별자 구성 규칙들의 예들을 도시한다.

도 24(a) 내지 도 24(e)는 TCP 접속들의 대역폭 변동들을 도시한다.

도 25는 소스 및 보수 데이터에 대한 다수의 HTTP 요청들을 도시한다.

도 26은 FEC를 갖는 예시적인 채널 재핑 시간 및 FEC를 갖지 않는 예시적인 채널 재핑 시간을 도시한다.

도 27은 도 1에 나타낸 수집 시스템의 일부로서, 소스 세그먼트들 및 제어 세그먼트들로부터 보수 세그먼트들을 생성하는 보수 세그먼트 생성기의 상세들을 도시한다.

도 28은 소스 블록들과 보수 블록들 사이의 관계들을 도시한다.

도 29는 클라이언트에서 상이한 시간들에 라이브 서비스들에 대한 절차를 도시한다.

도면들에서, 유사한 항목들은 유사한 번호들로 참조되고, 유사하거나 동일한 항목들의 다수의 예시들을 표시하기 위해 괄호에 서브-인덱스들이 제공된다. 달리 표시되지 않으면, 마지막 서브-인덱스(예를 들어, "N" 또는 "M")는 임의의 특정한 값으로 한정하려는 의도가 아니며, 하나의 항목의 예시들의 수는, 동일한 번호가 도시되고 서브-인덱스가 재사용되는 경우에도 다른 항목들의 예시들의 수와는 상이할 수 있다.

발명을 실시하기 위한 구체적인 내용

[0031] 본 명세서에서 설명되는 바와 같이, 스트리밍 시스템의 목적은 미디어를 그 미디어의 저장 위치(또는 미디어가 생성되고 있는 위치)로부터 미디어가 소비되고 있는 위치, 즉 사용자에게 제공되거나, 그렇지 않으면 인간 또는 전자적 소비자에 의해 "소모"되는 위치로 이동시키는 것이다. 이상적으로, 스트리밍 시스템은 수신 단에서 방해받지 않는 재생(또는 더 일반적으로는, 방해받지 않는 "소비")을 제공할 수 있고, 사용자가 스트림 또는 스트림들을 요청한 직후에 스트림 또는 스트림들의 집합체의 플레이를 시작할 수 있다. 사용자가 하나의 스트림으로부터 다른 스트림으로 스위칭하는 경우 또는 사용자가 예를 들어, "서브타이틀" 스트림과 같은 스트림의 프리젠테이션을 따르는 경우와 같이 일단 사용자가 그 스트림이 더 이상 필요하지 않은 것으로 표시하면, 효율성 때문에, 각각의 스트림은 중단되는 것이 또한 바람직할 수 있다. 비디오와 같은 미디어 컴포넌트가 제공되는 것이 계속되지만, 상이한 스트림이 그 미디어 컴포넌트를 제공하도록 선택되면, 새로운 스트림을 이용하여 제한된 대역폭을 점유하고 오래된 스트림을 중지하는 것이 종종 선호된다.

[0032] 본 명세서에서 설명되는 실시예들에 따른 블록-요청 스트리밍 시스템은 다수의 이점들을 제공한다. 몇몇 애플리케이션들은 본 명세서에서 설명되는 모든 특징들보다는 적은 특징들만으로도 적절히 만족스러운 경험을 제공할 수도 있기 때문에, 실행가능한 시스템은 본 명세서에서 설명되는 모든 특징들을 포함할 필요는 없음을 이해해야 한다.

[0033] HTTP 스트리밍

[0034] HTTP 스트리밍은 특정한 유형의 스트리밍이다. HTTP 스트리밍에 있어서, 소스들은 표준 웹 서버들 및 컨텐트 전달 네트워크(CDN)들일 수 있고, 표준 HTTP를 이용할 수 있다. 이 기술은 스트림 세그먼트화 및 다수의 스트림들의 이용을 수반할 수 있고, 이들 모두는 표준화된 HTTP 요청들의 콘텍스트 내에 있다. 비디오와 같은 미디어는 상이한 버전들 또는 리프리젠테이션들을 형성하도록 다수의 비트레이트들로 인코딩될 수 있다. 용어들 "버전" 및 "리프리젠테이션"은 본 명세서에서 동의어로 사용된다. 각각의 버전 또는 리프리젠테이션은 세그먼트들을 형성하도록, 아마도 각각 몇초 정도인 더 작은 피스(piece)들로 분해될 수 있다. 다음으로, 각각의 세그먼트는 웹 서버 또는 CDN 상에 별개의 파일로서 저장될 수 있다.

[0035] 다음으로, 클라이언트 측에서, 클라이언트에 의해 끊김없이 함께 스플라이싱되는(spliced) 개별 세그먼트들에 대한 요청들이 HTTP를 이용하여 행해질 수 있다. 클라이언트는 이용가능한 대역폭에 기초하여 상이한 데이터레이트들로 스위칭할 수 있다. 클라이언트는 또한, 각각 상이한 미디어 컴포넌트를 제공하는 다수의 리프리젠테이션들을 요청할 수 있고, 미디어를 이 리프리젠테이션들에서 함께 동기식으로 제공할 수 있다. 스위칭을 위한 트리거들은 예를 들어, 버퍼 점유 및 네트워크 측정들을 포함할 수 있다. 정상 상태에서 동작하는 경우, 클라이언트는 타겟 버퍼 점유를 유지하기 위해 서버로의 요청들을 일정 간격으로 할 수 있다.

[0036] HTTP 스트리밍의 이점들은 비트-레이트 적응성, 빠른 시작 및 탐색, 및 최소의 불필요한 전달을 포함할 수 있다. 이 이점들은, 플레이아웃에 앞서 오직 짧은 시간에만 행해지도록 전달을 제어하는 것, 이용가능한 대역

폭을 (가변 비트 레이트 미디어를 통해) 최대로 이용하는 것, 및 스트림 세그먼트화 및 지능적 클라이언트 절차들을 최적화하는 것으로부터 기인한다.

[0037] 미디어 프리젠테이션 디스크립션이 HTTP 스트리밍 클라이언트에 제공될 수 있어서, 클라이언트는 (예를 들어, 본 명세서에서는 3gp 세그먼트들로 지칭되는, 3GPP에 의해 특정되는 포맷들인) 파일들의 집합체를 이용하여 사용자에게 스트리밍 서비스를 제공할 수 있다. 미디어 프리젠테이션 디스크립션 및 이 미디어 프리젠테이션 디스크립션의 가능한 업데이트들은, 각각 미디어 컴포넌트들을 포함하는 세그먼트들의 구성된 집합체인 미디어 프리젠테이션을 설명하여, 클라이언트는 그 포함된 미디어를 동기화된 방식으로 제공할 수 있고, 탐색, 스위칭 비트레이트들 및 상이한 리프리젠테이션들에서의 미디어 컴포넌트들의 공동 제공과 같은 진보된 특징들을 제공할 수 있다. 클라이언트는 서비스의 제공을 위해 미디어 프리젠테이션 디스크립션 정보를 상이한 방식들로 이용할 수 있다. 더 상세하게는, 미디어 프리젠테이션 디스크립션으로부터, HTTP 스트리밍 클라이언트는, 데이터가 스트리밍 서비스 내의 사용자 및 클라이언트 성능에 유용하도록, 집합체 내의 어떤 세그먼트들이 액세스될 수 있는지 결정할 수 있다.

[0038] 몇몇 실시예들에서, 미디어 프리젠테이션 디스크립션은 정적일 수도 있지만, 세그먼트들은 동적으로 생성될 수도 있다. 미디어 프리젠테이션 디스크립션은 서비스에 대한 액세스 및 다운로드 시간을 최소화하도록 가능한 한 압축적일 수 있다. 예를 들어, 클라이언트와 서버 사이의 정기적인 또는 빈번한 타이밍 동기화와 같은 다른 전용 서버 접속이 최소화될 수 있다.

[0039] 미디어 프리젠테이션은, 상이한 액세스 네트워크 유형들로의 액세스, 상이한 현재 네트워크 조건들, 디스플레이 사이즈들, 액세스 비트레이트들 및 코덱 지원과 같은 상이한 성능들을 갖는 단말들에 의한 액세스를 허용하도록 구성될 수 있다. 다음으로, 클라이언트는 적절한 정보를 추출하여 사용자에게 스트리밍 서비스를 제공할 수 있다.

[0040] 미디어 프리젠테이션 디스크립션은 또한 요건들에 따른 압축성 및 배치 유동성을 허용할 수 있다.

[0041] 가장 단순한 예에서, 각각의 대안적 리프리젠테이션이 단일한 3GP 파일, 즉, 3GPP TS 26.244에서 정의된 바에 따르는 파일, 또는 ISO/IEC 14496-12 또는 이로부터 유도된 규격들에서 정의된 바와 같은 ISO 기반 미디어 파일 포맷(예를 들어, 3GPP 기술 규격 26.244에서 설명되는 3GP 파일 포맷)에 따르는 임의의 다른 파일에 저장될 수 있다. 본 명세서의 나머지 부분에서, 3GP 파일을 언급하는 경우, ISO/IEC 14496-12 및 이로부터 유도된 규격들은 모든 설명된 특징들을 ISO/IEC 14496-12 또는 임의의 유도된 규격들에서 정의되는 더 일반적인 ISO 기반 미디어 파일 포맷에 매핑할 수 있음을 이해해야 한다. 다음으로, 클라이언트는 영화 프래그먼트 시간들 및 바이트 오프셋들과 함께 미디어 메타데이터(전형적으로 영화 헤더 박스에 저장되고 또한 "moov" 박스로도 지칭됨)를 학습하기 위해 파일의 초기 부분을 요청할 수 있다. 다음으로, 클라이언트는 요구에 따라 영화 프래그먼트들을 획득하기 위해 HTTP 부분 획득 요청들을 발행할 수 있다.

[0042] 몇몇 실시예들에서, 각각의 리프리젠테이션을 다수의 세그먼트들로 스플릿하는 것이 바람직할 수 있다. 세그먼트 포맷이 3GP 파일 포맷에 기초하는 경우, 세그먼트들은, "시간별 스플릿"으로 지칭되는, 영화 프래그먼트들의 비중첩 시간 슬라이스들을 포함한다. 이 세그먼트들 각각은 다수의 영화 프래그먼트들을 포함할 수 있고, 각각은 자신의 권리로서 유효한 3GP 파일일 수 있다. 다른 실시예에서, 리프리젠테이션은, 메타데이터를 포함하는 초기 세그먼트(전형적으로 영화 헤더 "moov" 박스) 및 미디어 세그먼트들의 세트로 스플릿되고, 이들 각각은 미디어 데이터 및 초기 세그먼트의 연접(concatenation)을 포함하고, 임의의 미디어 세그먼트는 유효한 3GP 파일 뿐만 아니라 초기 세그먼트의 연접을 형성하고, 하나의 리프리젠테이션의 모든 미디어 세그먼트들은 유효한 3GP 파일을 형성한다. 전체 프리젠테이션은 각각의 세그먼트를 차례로 플레이아웃하고, 각각의 리프리젠테이션의 시작 시간에 따라 파일 내의 로컬 타임스탬프들을 글로벌 프리젠테이션 시간에 매핑함으로써 형성될 수 있다.

[0043] 본 상세한 설명 전체에 걸쳐, "세그먼트"에 대한 참조들은, 저장 매체로부터 완전히 또는 부분적으로 구성 또는 판독되거나, 그렇지 않으면, 예를 들어, HTTP 요청을 포함하는 파일 다운로드 프로토콜 요청의 결과로서 획득되는 임의의 데이터 객체를 포함하는 것으로 이해되어야 함을 유의해야 한다. 예를 들어, HTTP의 경우, 데이터 객체들은 HTTP 서버에 접속되거나 그 일부를 형성하는 디스크 또는 다른 저장 매체 상에 상주하는 실제 파일들에 저장될 수 있고, 또는 데이터 객체들은 HTTP 요청에 대한 응답으로 실행되는 CGI 스크립트 또는 다른 동적으로 실행되는 프로그램에 의해 구성될 수 있다. 용어 "파일" 및 "세그먼트"는 달리 특정되지 않으면 본 명세서에서 동의어로 사용된다. HTTP의 경우, 세그먼트는 HTTP 요청 응답의 엔티티 바디로서 고려될 수 있다.

[0044] 용어들 "프리젠테이션" 및 "컨텐트 항목"은 본 명세서에서 동의어로 사용된다. 다수의 예들에서, 프리젠테이션은 오디오, 비디오 또는 정의된 "플레이아웃" 시간을 갖는 다른 미디어 프리젠테이션이지만, 다른 변형들이 가능하다.

[0045] 용어들 "블록" 및 "프래그먼트"는 달리 특정되지 않으면 본 명세서에서 동의어로 사용되고, 일반적으로, 인덱싱되는 데이터의 최소 통합을 지칭한다. 이용가능한 인덱싱에 기초하여, 클라이언트는 상이한 HTTP 요청들에서 프래그먼트의 상이한 부분들을 요청할 수 있고, 또는 하나의 HTTP 요청에서 하나 또는 그 초과의 연속적 프래그먼트들 또는 프래그먼트들의 일부들을 요청할 수 있다. ISO 기반 미디어 파일 포맷에 기초한 세그먼트들 또는 3GP 파일 포맷에 기초한 세그먼트들이 이용되는 경우, 프래그먼트는 전형적으로, 영화 프래그먼트 헤더('moof') 박스와 미디어 데이터('mdat') 박스의 조합으로 정의되는 영화 프래그먼트를 지칭한다.

[0046] 본 명세서를 판독한 후 당업자는 본 명세서에서 설명되는 본 발명의 실시예들을 연속적 비트-스트림 네트워크들과 같은 다른 유형들의 전송 네트워크들에 적용할 수 있다는 인식으로, 여기서는, 본 명세서의 설명들을 단순화하기 위해 네트워크 전달 데이터가 패킷-기반인 것으로 가정한다.

[0047] 본 명세서를 판독한 후 당업자는 본 발명의 실시예들을 데이터의 비트-플립 손상과 같은 다른 유형들의 데이터 전송 문제들에 적용할 수 있다는 인식으로, 여기서는, 본 명세서의 설명들을 단순화하기 위해, FEC 코드들이 데이터의 길고 가변적인 전달 시간들에 대한 보호를 제공하는 것으로 가정한다. 예를 들어, FEC가 없으면, 요청된 프래그먼트의 마지막 부분이 프래그먼트의 이전 부분들보다 훨씬 나중에 도달하거나 그 도달 시간에 큰 가변성을 갖는 경우, 컨텐트 재핑 시간이 크고 가변적일 수 있지만, FEC 및 병렬 요청들을 이용하면, 프래그먼트에 대해 요청된 데이터가 복원될 수 있기 전에 그 데이터의 오직 과반(majority)만 도달되도록 요구되어, 컨텐트 재핑 시간 및 컨텐트 재핑 시간에서의 가변성을 감소시킨다. 본 설명에서, 인코딩될 데이터(즉, 소스 데이터)는, (단일 비트까지의) 임의의 길이일 수 있는 동일한 길이의 "심볼들"로 분해된 것으로 가정될 수 있지만, 심볼들은 데이터의 상이한 부분들에 대해 상이한 길이들일 수 있어서, 예를 들어, 상이한 데이터 블록들에 대해 상이한 심볼 사이즈들이 이용될 수 있다.

[0048] 본 설명에서는, 본 명세서의 설명들을 단순화하기 위해, FEC가 데이터의 "블록" 또는 "프래그먼트"에 한번 적용되는 것으로 가정하는데, 즉, "블록"이 FEC 인코딩 및 디코딩 목적들을 위한 "소스 블록"이다. 클라이언트 디바이스는 세그먼트의 소스 블록 구조를 결정하는 것을 돕기 위해 본 명세서에서 설명되는 세그먼트 인덱싱을 이용할 수 있다. 당업자는 본 발명의 실시예들을 다른 유형들의 소스 블록 구조들에 적용할 수 있어서, 예를 들어, 소스 블록은 프래그먼트의 일부일 수 있거나 또는 하나 또는 그 초과의 프래그먼트들 또는 프래그먼트들의 부분들을 포함할 수 있다.

[0049] 블록-요청 스트리밍에 이용하기 위해 고려되는 FEC 코드들은 전형적으로 시스터메틱 FEC 코드들이어서, 즉, 소스 블록의 소스 심볼들은 소스 블록의 인코딩의 일부로서 포함될 수 있고, 따라서, 소스 심볼들이 전송된다. 당업자는, 본 명세서에서 설명되는 실시예들이, 시스터메틱이 아닌 FEC 코드들에도 동등하게 잘 적용됨을 인식할 것이다. 시스터메틱 FEC 인코더는 소스 심볼들의 소스 블록으로부터 몇몇 개수의 보수 심볼들을 생성하고, 소스 및 보수 심볼들의 적어도 일부의 조합은, 소스 블록을 표현하는 채널을 통해 전송되는 인코딩된 심볼들이다. 몇몇 FEC 코드들은 "정보 부가적 코드들" 또는 "파운튼(fountain) 코드들"과 같은, 필요한 만큼 많은 보수 심볼들을 효율적으로 생성하는데 유용할 수 있고, 이 코드들의 예들은 "연쇄 반응 코드들" 및 "멀티-스테이지 연쇄 반응 코드들"을 포함한다. Reed-Solomon 코드들과 같은 다른 FEC 코드들은 실제로 각각의 소스 블록에 대해 제한된 수의 보수 심볼들만을 생성할 수 있다.

[0050] 이 예들의 대부분에서, 클라이언트는 일 미디어 서버 또는 복수의 미디어 서버들에 커플링되고, 클라이언트는 일 미디어 서버 또는 복수의 미디어 서버들로부터 일 채널 또는 복수의 채널들을 통해 스트리밍 미디어를 요청하는 것으로 가정한다. 그러나, 더 많은 관련된 배열들이 또한 가능하다.

[0051] 이점들의 예들

[0052] 블록-요청 스트리밍에 있어서, 미디어 클라이언트는 이 블록 요청들의 타이밍과 사용자에게 플레이아웃되는 미디어의 타이밍 사이의 커플링을 유지한다. 이 모델은, 전술한 "다운로드" 모델의 이점들을 보유하면서, 데이터 전달로부터 미디어 플레이아웃의 통상적 디커플링으로부터 기인하는 단점들의 일부를 회피할 수 있다. 블록-요청 스트리밍 모델은, TCP와 같은 전송 프로토콜들에서 이용가능한 레이트 및 혼잡 제어 메커니즘들을 이용하여, 미디어 데이터에 대한 최대 이용가능한 대역폭이 이용되는 것을 보장한다. 부가적으로, 미디어 프

리젠테이션의 블록들로의 분할은 인코딩된 미디어 데이터의 각각의 블록이 다수의 이용가능한 인코딩들의 세트로부터 선택되도록 허용한다.

[0053] 이 선택은, 이용가능한 대역폭이 시간에 따라 변경되고 있는 경우에도 그 이용가능한 대역폭에 미디어 데이터 레이트를 매칭시키는 기준, 클라이언트 성능들 또는 구성에 미디어 해상도 또는 디코딩 복잡도를 매칭시키는 기준, 또는 언어들과 같은 사용자 선호도들에 매칭시키는 기준을 포함하는 임의의 수의 기준에 기초할 수 있다. 이 선택은 또한, 액세스 가능성 컴포넌트들, 폐쇄 자막화(closed captioning), 부제들(sub-titles), 부호 언어 비디오 등과 같은 보조 컴포넌트들의 다운로드 및 프리젠테이션을 포함할 수 있다. 블록-요청 스트리밍 모델을 이용하는 기존 시스템들의 예들은 Move Networks™, Microsoft Smooth Streaming 및 Apple iPhone™ Streaming Protocol을 포함한다.

[0054] 통상적으로, 미디어 데이터의 각각의 블록은 개별 파일로서 서버 상에 저장될 수 있고, 일 단위로서 파일을 요청하기 위해, 서버 상에서 실행되는 HTTP 서버 소프트웨어와 함께 HTTP와 같은 프로토콜이 이용된다. 전형적으로, 클라이언트에는 메타데이터 파일들이 제공되고, 메타데이터 파일들은, 예를 들어, XML(Extensible Markup Language) 포맷 또는 플레이리스트 텍스트 포맷 또는 바이너리 포맷일 수 있고, 본 명세서에서는 전형적으로 "리프리젠테이션들"로 지칭되는 이용가능한 인코딩들(예를 들어, 요구되는 대역폭, 해상도들, 인코딩 파라미터들, 미디어 유형, 언어) 및 인코딩들이 블록들로 분할된 방식과 같은 미디어 프리젠테이션의 특징들을 설명한다. 예를 들어, 메타데이터는 각각의 블록에 대한 URL(Uniform Resource Locator)을 포함할 수 있다. URL들 자체는, 문서화된 자원에 액세스하는데 이용될 프로토콜이 HTTP임을 표시하기 위해, 스트링 "http://"가 선행되는 것과 같은 방식을 제공할 수 있다. 다른 예는, 이용될 프로토콜이 FTP임을 표시하기 위한 "ftp://"이다.

[0055] 예를 들어, 다른 시스템들에서, 미디어 블록들은 요청되는 미디어 프리젠테이션 부분을 적시에 표시하는 클라이언트로부터의 요청에 응답하여 서버에 의해 "온더플라이(on-the-fly)"로 구성될 수 있다. 예를 들어, 방식 "http://"를 갖는 HTTP의 경우, 이 URL의 요청의 실행은 요청 응답을 제공하며, 요청 응답은 이 요청 응답의 엔티티 바디에 몇몇 특정한 데이터를 포함시킨다. 이 요청 응답을 어떻게 생성할지에 대한 네트워크에서의 구현은 이러한 요청들을 서비스하는 서버의 구현에 따라 매우 상이할 수 있다.

[0056] 전형적으로, 각각의 블록은 독립적으로 디코딩가능할 수 있다. 예를 들어, 비디오 미디어의 경우, 각각의 블록은 "탐색 포인트"로 시작할 수 있다. 몇몇 코딩 방식들에서, 탐색 포인트는 "랜덤 액세스 포인트들" 또는 "RAPs"로 지칭되지만, 모든 RAP들이 탐색 포인트로서 지정되지는 않을 수 있다. 유사하게, 다른 코딩 방식들에서, 탐색 포인트는 "독립 데이터 리프레시" 프레임 또는 "IDR"에서 시작하지만, H.264 비디오 인코딩의 경우, 모든 IDR들이 탐색 포인트로서 지정되지는 않을 수 있다. 탐색 포인트는, 디코딩되고 있는 프레임 또는 샘플이 독립형 방식으로 인코딩되지 않았지만, 예를 들어 현재 프레임과 이전 프레임 사이의 차이로 인코딩된 경우에서와 같이, 디코더가 이전 프레임들 또는 데이터 또는 샘플들에 대한 임의의 데이터를 요구하지 않고 디코딩을 시작할 수 있는 비디오 (또는 다른) 미디어 내의 위치이다.

[0057] 이러한 시스템들에서의 관심사는, 예를 들어, 수신된 오디오 및 비디오 스트림들을 개인용 컴퓨터를 이용하여 디코딩 및 렌더링하고, 비디오를 컴퓨터 스크린 상에 디스플레이하고 내장 스피커들을 통해 오디오를 플레이하는 것, 또는 다른 예로, 수신된 오디오 및 비디오 스트림들을 셋탑 박스를 이용하여 디코딩 및 렌더링하고, 비디오를 텔레비젼 디스플레이 디바이스 상에 디스플레이하고 스테레오 시스템을 통해 오디오를 플레이하는 것과 같은 스트림의 플레이아웃을 시작하는 능력일 수 있다. 주요 관심사는, 사용자가 스트림으로 전달된 새로운 컨텐트를 시청하려 결정하고, 예를 들어, 사용자가 브라우저 윈도우 내의 링크를 클릭하거나 원격 제어 디바이스의 플레이 버튼을 클릭하는 것과 같이, 그 결정을 표현하는 동작을 취할 때와, 이하, "컨텐트 재핑 시간"으로 지칭되는, 컨텐트가 사용자의 스크린 상에 디스플레이되기 시작하는 때 사이의 지연을 최소화하는 것일 수 있다. 이들 관심사들 각각은 본 명세서에서 설명되는 향상된 시스템의 엘리먼트들에 의해 처리될 수 있다.

[0058] 컨텐트 재핑의 일예는, 사용자가 제 1 스트림을 통해 전달된 제 1 컨텐트를 시청하고 있고, 다음으로, 사용자가 제 2 스트림을 통해 전달된 제 2 컨텐트를 시청하려 결정하고, 그 제 2 컨텐트를 시청하는 것을 시작하기 위한 동작을 개시할 때이다. 제 2 스트림은 제 1 스트림으로서의 서버들과 동일한 세트 또는 상이한 세트의 서버들로부터 전송될 수 있다. 컨텐트 재핑의 다른 예는, 사용자가 웹사이트를 방문하고 있고, 브라우저 윈도우 내의 링크를 클릭함으로써 제 1 스트림을 통해 전달된 제 1 컨텐트를 시청하기 시작하려 결정할 때이다. 유사한 방식으로, 사용자는 컨텐트를 시작부부터 플레이하지 않고 스트림 내의 어느 시점부터 플레이하기 시작하려 결정할 수 있다. 사용자는 자신의 클라이언트 디바이스에 일 시간 위치를 탐색할 것을 표시하고, 사

용자는, 그 선택된 시간이 즉시 렌더링될 것을 기대할 수도 있다. 컨텐트 재핑 시간을 최소화하는 것은, 광범위한 이용가능한 컨텐트들을 탐색하고 샘플링할 때 사용자에게 고품질의 빠른 컨텐트 서핑 경험을 허용하기 위해, 비디오 시청에 있어서 중요하다.

[0059] 최근에, 전송 동안 스트리밍 미디어의 보호를 위해 순방향 에러 정정(FEC) 코드들의 이용을 고려하는 것이 통상적인 관행이 되고 있다. 3GPP, 3GPP2 및 DVB와 같은 그룹들에 의해 표준화된 네트워크들과 같은 무선 네트워크들 및 인터넷을 예시들로 포함하는 패킷 네트워크를 통해 전송될 때, 소스 스트림은 이용가능하게 생성 또는 형성되는 패킷들에 배치되고, 따라서, 이 패킷들은, 이들이 이용가능하게 생성 또는 형성된 순서로 소스 또는 컨텐트 스트림을 수신기들에 전달하는데 이용될 수 있다.

[0060] FEC 코드들을 이러한 유형들의 시나리오들에 전형적으로 적용할 때, 인코더는 보수 패킷들의 생성 시에 FEC 코드를 이용할 수 있고, 다음으로, 보수 패킷들은 소스 스트림을 포함하는 본래의 소스 패킷들에 부가하여 전송된다. 보수 패킷들은, 소스 패킷 손실이 발생할 때, 그 손실된 소스 패킷들에 포함된 데이터를 복원하기 위해, 수신된 보수 패킷들이 이용될 수 있는 특성을 갖는다. 보수 패킷들은, 완전히 손실된 손실 소스 패킷들의 컨텐트를 복원하는데 이용될 수 있지만, 또한, 부분적 패킷 손실이 발생한 것으로부터, 완전히 수신된 보수 패킷들 또는 심지어 부분적으로 수신된 보수 패킷들을 복원하는데 이용될 수도 있다. 따라서, 전체적으로 또는 부분적으로 수신된 보수 패킷들은 전체적으로 또는 부분적으로 손실된 소스 패킷들을 복원하는데 이용될 수 있다.

[0061] 또 다른 예들에서, 전송된 데이터에 다른 유형들의 손상이 발생할 수 있어서, 예를 들어, 비트들의 값들이 플립될 수 있고, 따라서, 보수 패킷들은 이러한 손상을 정정하는데 이용될 수 있고, 가능한 한 정확한 소스 패킷들의 복원을 제공할 수 있다. 다른 예들에서, 소스 스트림은 반드시 이산적 패킷들로 전송될 필요는 없고, 대신에, 예를 들어, 연속적 비트 스트림으로 전송될 수 있다.

[0062] 소스 스트림의 보호를 제공하기 위해 이용될 수 있는 FEC 코드들의 다수의 예들이 존재한다. Reed-Solomon 코드들은 통신 시스템들에서 에러 및 소거 정정을 위한 주지의 코드들이다. 예를 들어, 패킷 데이터 네트워크들에 대한 소거 정정의 경우, Reed-Solomon 코드들의 주지의 효율적인 구현은 L. Rizzo의 "Effective Erasure Codes for Reliable Computer Communication Protocols", Computer Communication Review, 27(2):24-36 (1997년 4월)(이하, "Rizzo"), 및 Bloemer 등의 "An XOR-Based Erasure-Resilient Coding Scheme", Technical Report TR-95-48, International Computer Science Institute, Berkeley, California (1995)(이하, "XOR Reed Solomon") 또는 다른 곳에서 설명되는 Cauchy 또는 Vandermonde 행렬들을 이용한다.

[0063] FEC 코드들의 다른 예들은 LDPC 코드들, Luby I에서 설명되는 것과 같은 연쇄 반응 코드들 및 Shokrollahi I에서와 같은 멀티-스테이지 연쇄 반응 코드들을 포함한다.

[0064] Reed-Solomon 코드들의 변형들에 대한 FEC 디코딩 프로세스의 예들은 Rizzo 및 XOR-Reed-Solomon에 설명되어 있다. 이 예들에서, 디코딩은, 충분한 소스 및 보수 데이터 패킷들이 수신된 이후에 적용될 수 있다. 디코딩 프로세스는 계산 집약적일 수 있고, 이용가능한 CPU 자원들에 따라, 블록 내에서 미디어가 걸쳐있는 시간 길이에 비해, 완료하는데 상당한 시간을 소요할 수 있다. 수신기는, 미디어 스트림의 수신의 시작과 미디어의 플레이아웃 사이에서 요구되는 지연을 계산할 때, 디코딩에 요구되는 이 시간 길이를 고려할 수 있다. 디코딩에 기인한 이러한 지연은 특정한 미디어 스트림에 대한 사용자들의 요청과 재생의 시작 사이의 지연으로서 사용자에 의해 인지된다. 따라서, 이러한 지연을 최소화하는 것이 바람직하다.

[0065] 다수의 애플리케이션들에서, 패킷들은, FEC 프로세스가 적용되는 심볼들로 추가로 세분화될 수 있다. 패킷은 하나 또는 그 초과의 심볼을 포함할 수 있다(또는 하나 미만의 심볼을 포함할 수 있지만, 통상적으로 심볼들은, 패킷들의 그룹들 사이에서 에러-조건들이 매우 상관되는 것으로 알려지지 않으면, 패킷들의 그룹들에 걸쳐 스플릿되지 않는다). 심볼은 임의의 사이즈를 가질 수 있지만, 종종 심볼의 사이즈는 최대의 경우에 패킷의 사이즈와 동일하다. 소스 심볼들은 전송될 데이터를 인코딩하는 심볼들이다. 보수 심볼들은 소스 심볼들로부터 직접 또는 간접적으로 생성되는 심볼들이고 소스 심볼들에 부가된다(즉, 소스 심볼들 모두가 이용가능하고 보수 심볼들 모두가 이용가능하지 않으면, 전송될 데이터는 완전히 복원될 수 있다).

[0066] 몇몇 FEC 코드들은, 인코딩 동작들이 일 블록 내의 심볼(들)에 의존하고, 그 블록에 있지 않은 심볼들에는 독립적일 수 있다는 점에서 블록-기반일 수 있다. 블록-기반 인코딩에 있어서, FEC 인코더는 블록에 대한 보수 심볼들을 그 블록 내의 소스 심볼들로부터 생성할 수 있고, 그 후, 다음 블록으로 진행하며, 인코딩되고 있는 현재의 블록에 대한 소스 심볼들이 이외의 소스 심볼들을 참조할 필요가 없다. 전송 시에, 소스 심볼들을 포함하는 소스 블록은 인코딩된 심볼들(일부 소스 심볼들, 일부 보수 심볼들 또는 둘 모두를 포함할 수 있음)을

포함하는 인코딩된 블록에 의해 표현될 수 있다. 보수 심볼들의 존재에 의해, 소스 심볼들 모두가 모든 인코딩된 블록에 요구되지는 않는다.

[0067] 일부 FEC 코드들, 특히 Reed-Solomon 코드들의 경우, 소스 블록 당 인코딩된 심볼들의 수가 증가함에 따라, 인코딩 및 디코딩 시간은 비실용적이 될 수 있다. 따라서, 실제로, 소스 블록 당 생성될 수 있는 인코딩된 심볼들의 총 수에 대해 실용적인 상한이 종종 존재하며(일부 애플리케이션들의 경우 255가 대략적으로 실용적인 제한임), 특히, Reed-Solomon 인코딩 또는 디코딩 프로세스가 종래의 하드웨어에 의해 수행되는 경우, 예를 들어, 스트림들을 패킷 손실로부터 보호하기 위한 DVB-H 표준의 일부로서 포함되는 Reed-Solomon 코드들을 이용하는 MPE-FEC 프로세스들은 소스 블록 당 255개의 Reed-Solomon 총 인코딩된 심볼들로 제한되는 셀 폰 내의 특수한 하드웨어로 구현된다. 심볼들은 종종 별개의 패킷 페이로드들에 배치되도록 요구되기 때문에, 이것은, 인코딩되는 소스 블록의 최대 길이에 대해 실용적인 상한이 된다. 예를 들어, 패킷 페이로드가 1024 바이트 또는 그 미만으로 제한되고, 각각의 패킷이 하나의 인코딩된 심볼을 전달하면, 인코딩된 소스 블록은 최대의 경우 255 킬로바이트일 수 있고, 물론, 이것은 또한 소스 블록 자체의 사이즈에 대한 상한이다.

[0068] 소스 스트리밍 레이트에 뒤쳐지지 않고, FEC 디코딩에 의해 도입되는 디코딩 레이턴시를 최소화하고, FEC 디코딩 동안 임의의 시점에 수신 디바이스 상에서 이용가능한 CPU의 작은 부분만을 이용하도록, 소스 블록들을 충분히 빠르게 디코딩할 수 있는 것과 같은 다른 관심사들은 본 명세서에서 설명되고 취급되는 엘리먼트들에 의해 처리된다.

[0069] 수신기들에 전달되는 스트림들의 품질에 악영향을 주지 않으면서 시스템의 컴포넌트들이 실패하도록 허용하는 견고하고 스케일가능한 스트리밍 전달 솔루션을 제공할 필요가 있다.

[0070] 블록 요청 스트리밍 시스템은, 예를 들어, 이용가능한 미디어 인코딩들의 수에 대한 변경들, 또는 비트 레이트, 해상도, 종횡비, 오디오 또는 비디오 코덱들, 또는 컨텐트 파일들과 연관된 URL들과 같은 다른 메타데이터의 변경들의 코덱 파라미터들과 같은 미디어 인코딩들의 파라미터들에 대한 변화들과 같이, 프리젠테이션의 구조 또는 메타데이터에 대한 변경들을 지원할 필요가 있다. 이러한 변경들은, 더 큰 프리젠테이션의 상이한 세그먼트들 또는 광고와 같은 상이한 소스들로부터의 컨텐트를 함께 편집하는 것, URL들 또는 예를 들어, 구성 변경들, 장비 실패들 또는 장비 실패들로부터의 복원 또는 다른 이유들에 기인한 서빙 인프라구조에서의 변경들의 결과로서 필요해지는 다른 파라미터들의 변형을 포함하는 다수의 이유들 때문에 요구될 수 있다.

[0071] 연속적으로 업데이트된 플레이리스트 파일에 의해 프리젠테이션이 제어될 수 있는 방법들이 존재한다. 이 파일은 연속적으로 업데이트되기 때문에, 전술한 변경들 중 적어도 일부는 이 업데이트들 내에서 행해질 수 있다. 종래의 방법의 단점은, 클라이언트 디바이스들이 플레이리스트 파일을 연속적으로 리트리브하여(또한, "폴링(polling)"으로 지칭됨), 서빙 인프라구조에 부담을 준다는 점, 및 이 파일은 업데이트 인터벌보다 더 오래 캐시되지 못할 수 있어서, 서빙 인프라구조에 대한 작업을 훨씬 더 곤란하게 한다는 점이다. 이것은, 메타데이터 파일에 대한 클라이언트들에 의한 연속적 폴링을 요구하지 않고, 전술된 종류의 업데이트들이 제공되어, 본 명세서의 엘리먼트들에 의해 처리된다.

[0072] 전형적으로 브로드캐스트 분배로부터 알려진, 특히, 라이브 서비스들에서의 다른 문제는, 사용자가 그 프로그램에 참여하는 시간보다 더 일찍 브로드캐스트된 컨텐트를 사용자가 시청할 수 있는 능력이 부족하다는 것이다. 전형적으로, 클라이언트가 그 프로그램에 튜닝되지 않았거나 컨텐트 보호 규칙들에 의해 금지되기 때문에, 로컬 개인 레코딩은 불필요한 로컬 저장부를 소비하거나 불가능하다. 네트워크 레코딩 및 시간-시프트 시청이 선호되지만, 사용자의 서버로의 개별적 접속들, 및 라이브 서비스들과는 별개의 전달 프로토콜 및 인프라구조를 요구하여, 중복적 인프라구조 및 현저한 서비스 비용을 초래한다. 이것은 또한 본 명세서에서 설명되는 엘리먼트들에 의해 처리된다.

[0073] 시스템 개요

[0074] 본 발명의 일 실시예가 도 1을 참조하여 설명되고, 도 1은 본 발명을 구현하는 블록-요청 스트리밍 시스템의 개략도를 도시한다.

[0075] 도 1에는, 컨텐트(102)를 인제스팅하고, 그 컨텐트를 준비하고, 수집 시스템(103) 및 HTTP 스트리밍 서버(104) 모두에 액세스가능한 컨텐트 저장부(110)에 컨텐트를 저장함으로써 HTTP 스트리밍 서버(104)에 의한 서비스를 위해 컨텐트를 패키징하는 수집 시스템(103)을 포함하는 블록 서빙 인프라구조("BSI")(101)를 구비하는 블록-스트리밍 시스템(100)이 도시되어 있다. 도시된 바와 같이, 시스템(100)은 또한 HTTP 캐시(106)를

포함할 수 있다. 동작 시에, HTTP 스트리밍 클라이언트와 같은 클라이언트(108)는 HTTP 스트리밍 서버(104)에 요청들(112)을 전송하고, HTTP 스트리밍 서버(104) 또는 HTTP 캐시(106)로부터 응답들(114)을 수신한다. 각각의 경우에, 도 1에 도시된 엘리먼트들은 적어도 부분적으로 소프트웨어로 구현될 수 있고, 소프트웨어는 프로세서 또는 다른 전자기기들 상에서 실행되는 프로그램 코드를 포함한다.

[0076] 컨텐트는 영화들, 오디오, 2D 평면 비디오, 3D 비디오, 다른 유형들의 비디오, 이미지들, 정기적 텍스트, 정기적 메타데이터 등을 포함할 수 있다. 일부 컨텐트는, 플레이아웃되고 있는 다른 미디어와 함께 보조 정보(스테이션 식별자, 광고, 주식 시세, Flash™ 시퀀스들 등)를 제공하기 위한 데이터와 같이, 정기적 방식으로 제공되거나 소비될 데이터를 수반할 수 있다. 다른 미디어를 결합하고 그리고/또는 단순한 오디오 및 비디오를 넘는 다른 하이브리드 프리젠테이션이 또한 이용될 수 있다.

[0077] 도 2에 도시된 바와 같이, 미디어 블록들은 블록 서빙 인프라구조(101(1)) 내에 저장될 수 있고, 블록 서빙 인프라구조(101(1))는, 예를 들어, HTTP 서버, 컨텐트 전달 네트워크 디바이스, HTTP 프록시, FTP 프록시 또는 서버, 또는 몇몇 다른 미디어 서버 또는 시스템일 수 있다. 블록 서빙 인프라구조(101(1))는 네트워크(122)에 접속되고, 네트워크(122)는 예를 들어, 인터넷과 같은 인터넷 프로토콜("IP") 네트워크일 수 있다. 블록-요청 스트리밍 시스템 클라이언트는 6개의 기능 컴포넌트들, 즉, 전술한 메타데이터가 제공되고, 메타데이터에 의해 표시되는 복수의 이용가능한 블록들 중에서 요청될 블록들 또는 부분적 블록들을 선택하는 기능을 수행하는 블록 선택기(123), 블록 선택기(123)로부터의 요청 명령들을 수신하며, 특정한 블록, 블록의 일부들 또는 다수의 블록들에 대한 요청을 네트워크(122)를 통해 블록 서빙 인프라구조(101(1))에 전달하고, 리턴 시에 그 블록을 포함하는 데이터를 수신하는데 필요한 동작들을 수행하는 블록 요청기(124) 뿐만 아니라 블록 버퍼(125), 버퍼 모니터(126), 미디어 디코더(127) 및 미디어 소비를 용이하게 하는 하나 또는 그 초과의 미디어 트랜스듀서들(128)을 갖는 것으로 도시되어 있다.

[0078] 블록 요청기(124)에 의해 수신되는 블록 데이터는 일시적 저장을 위해, 미디어 데이터를 저장하는 블록 버퍼(125)에 전달된다. 대안적으로, 수신된 블록 데이터는 도 1에 도시된 바와 같이 블록 버퍼(125) 내에 직접 저장될 수 있다. 미디어 디코더(127)는 블록 버퍼(125)에 의해 미디어 데이터를 제공받고, 이 데이터에 대해 미디어 트랜스듀서들(128)에 적합한 입력을 제공하는데 필요한 변환들을 수행하고, 미디어 트랜스듀서들(128)은 미디어를 사용자 또는 다른 소비에 적합한 형태로 렌더링한다. 미디어 트랜스듀서들의 예들은, 모바일 폰들, 컴퓨터 시스템들 또는 텔레비젼들에서 발견되는 것과 같은 시각적 디스플레이 디바이스들을 포함하고, 또한, 스피커들 또는 헤드폰들과 같은 오디오 렌더링 디바이스들을 포함할 수 있다.

[0079] 미디어 디코더의 일예는, H.264 비디오 코딩 표준에서 설명되는 포맷의 데이터를, 각각의 프레임 또는 샘플에 대한 연관된 프리젠테이션 타임스탬프들과의 YUV-포맷 픽셀 맵과 같은, 비디오 프레임들의 아날로그 또는 디지털 리프리젠테이션들로 변환하는 기능일 것이다.

[0080] 버퍼 모니터(126)는 블록 버퍼(125)의 컨텐트들과 관련된 정보를 수신하고, 이 정보 및 가능하게는 다른 정보에 기초하여, 블록 선택기(123)로의 입력을 제공하고, 이 입력은 본 명세서에서 설명되는 바와 같이 요청할 블록들의 선택을 결정하는데 이용된다.

[0081] 본 명세서에서 사용되는 용어에서, 각각의 블록은 "플레이아웃 시간" 또는 "지속시간"을 갖고, 이것은, 수신기가 그 블록에 포함된 미디어를 정규의 속도로 플레이하는데 소요될 시간량을 표현한다. 몇몇 경우들에서, 블록 내의 미디어의 플레이아웃은 이전 블록들로부터 이미 수신된 데이터를 갖는 것에 의존할 수 있다. 드문 경우들에서, 블록 내의 미디어의 일부의 플레이아웃은 후속 블록에 의존할 수 있고, 이 경우, 블록에 대한 플레이아웃 시간은 후속 블록을 참조함이 없이 블록 내에서 플레이아웃될 수 있는 미디어에 대해 정의되고, 후속 블록에 대한 플레이아웃 시간은 그 후속 블록을 수신한 후에만 플레이아웃할 수 있는 이 블록 내의 미디어의 플레이아웃 시간에 의해 증가된다. 후속 블록들에 의존하는 블록 내에 미디어를 포함시키는 것은 드문 경우이기 때문에, 본 명세서의 나머지 부분에서, 하나의 블록의 미디어는 후속 블록들에 의존하지 않는 것으로 가정하지만, 이러한 변형이 후술되는 실시예들에 쉽게 부가될 수 있음을 당업자는 인식할 것을 유의한다.

[0082] 수신기는 "일시중지", "빨리감기", "되감기" 등과 같은 제어들을 가질 수 있고, 이것은 블록이 상이한 레이트의 플레이아웃에 의해 소비되는 것을 초래할 수 있지만, 수신기가, 시퀀스의 마지막 블록을 제외한 통합 플레이아웃 시간과 동일하거나 그보다 작은 통합 시간에서 블록들의 연속적 시퀀스 각각을 획득하고 디코딩할 수 있으면, 수신기는 스톨링없이 그 미디어를 사용자에게 제공할 수 있다. 본 명세서의 일부 설명들에서, 미디어 스트림의 특정한 위치는 미디어 내의 특정한 "시간"으로 지칭되고, 이것은, 미디어 플레이아웃의 시작과 비디오 스트림 내의 특정한 위치에 도달되는 시간 사이에 경과하게 될 시간에 대응한다. 미디어 스트림 내의

시간 또는 위치는 종래의 개념이다. 예를 들어, 비디오 스트림이 초당 24개의 프레임들을 포함하는 경우, 첫 번째 프레임은 t=0.0 초의 위치 또는 시간을 가진다고 말할 수 있고, 241번째 프레임은 t=10.0 초의 위치 또는 시간을 가진다고 말할 수 있다. 물론, 프레임-기반 비디오 스트림에서, 241번째 프레임의 제 1 비트로부터 242번째 프레임의 제 1 비트 직전까지 스트림 내의 비트들 각각은 모두 동일한 시간 값을 가질 수 있기 때문에, 위치 또는 시간은 연속적일 필요가 없다.

[0083] 상기 용어를 채택하면, 블록-요청 스트리밍 시스템(BRSS)은, 하나 또는 그 초과의 컨텐트 서버들(예를 들어, HTTP 서버들, FTP 서버들 등)에 요청하는 하나 또는 그 초과의 클라이언트들을 포함한다. 수집 시스템은 하나 또는 그 초과의 수집 프로세서들을 포함하고, 수집 프로세서는 컨텐트를 (실시간으로 또는 비실시간으로) 수신하고, BRSS에 의한 이용을 위해 컨텐트를 프로세싱하고, 또한 가능하게는 수집 프로세서에 의해 생성된 메타데이터와 함께 이 컨텐트를 컨텐트 서버들에 액세스가능한 저장부에 저장한다.

[0084] BRSS는 또한 컨텐트 서버들과 조정하는 컨텐트 캐시들을 포함할 수 있다. 컨텐트 서버들 및 컨텐트 캐시들은, URL을 포함하는 HTTP 요청들의 형태로 파일들 또는 세그먼트들에 대한 요청들을 수신하는 종래의 HTTP 서버들 및 HTTP 캐시들일 수 있고, 또한, URL에 의해 표시되는 파일 또는 세그먼트 전부보다 덜 요청하기 위해 바이트 범위를 포함할 수 있다. 클라이언트들은, HTTP 서버들의 요청들을 행하고 이 요청들에 대한 응답들을 처리하는 종래의 HTTP 클라이언트를 포함할 수 있으며, 여기서, HTTP 클라이언트는, 요청들을 포뮬레이트하고(formulate), 이들을 HTTP 클라이언트에 전달하고, HTTP 클라이언트로부터 응답들을 획득하고, 이들을 클라이언트 디바이스에 의한 플레이아웃을 위해 프리젠테이션 플레이어에 제공하기 위해 이들을 프로세싱(또는 저장, 변환 등)하는 신규한 클라이언트 시스템에 의해 구동된다. 전형적으로, 클라이언트 시스템은 어떤 미디어가 요구될지를 미리 알지 못하고, 따라서, 미디어가 수신되자마자 또는 수신된 직후 "소비된다"는 점에서 "스트리밍" 시스템으로 불린다. 그 결과, 응답 지연들 및 대역폭 제한들은, 사용자가 프리젠테이션을 소비하고 있는 위치를 스트림이 따라잡을 때 프리젠테이션에서의 일시중지를 유발시키는 것과 같은, 프리젠테이션에서의 지연들을 유발시킬 수 있다.

[0085] 양호한 품질인 것으로 인지되는 프리젠테이션을 제공하기 위해, 다수의 세부사항들이 BRSS에서 클라이언트 단, 수집 단, 또는 둘 모두에서 구현될 수 있다. 몇몇 경우들에서, 구현되는 세부사항들은 네트워크의 클라이언트-서버 인터페이스를 고려하고 이를 처리하도록 행해진다. 몇몇 실시예들에서, 클라이언트 시스템 및 수집 시스템 모두가 향상을 인식할 수 있는 한편, 다른 실시예들에서는, 오직 일측만이 향상을 인식할 수 있다. 이러한 경우들에서, 일측이 향상을 인식할 수 없는 경우에도 전체 시스템은 향상으로부터 이점이 있는 한편, 다른 경우들에서는, 양측 모두가 향상을 인식하면 이점이 오직 누적되지만, 일측이 인식하지 못하는 경우 여전히 실패없이 동작한다.

[0086] 도 3에 도시된 바와 같이, 수집 시스템은 다양한 실시예들에 따라 하드웨어와 소프트웨어 컴포넌트들의 조합으로 구현될 수 있다. 수집 시스템은, 이 시스템으로 하여금 본 명세서에서 설명되는 방법들 중 임의의 하나 또는 그 초과를 수행하도록 실행될 수 있는 명령들의 세트를 포함할 수 있다. 시스템은 컴퓨터의 형태로 특수한 머신으로서 실현될 수 있다. 시스템은 서버 컴퓨터, 개인용 컴퓨터(PC), 또는 그 시스템에 의해 행해지는 동작들을 특정하는 명령들의 세트를 (순차적으로 또는 다른 방식으로) 실행할 수 있는 임의의 시스템일 수 있다. 추가적으로, 오직 단일 시스템이 도시되지만, 용어 "시스템"은 또한, 본 명세서에서 설명되는 방법들 중 임의의 하나 또는 그 초과를 수행하는 명령들의 세트(또는 다수의 세트들)를 개별적으로 또는 공동으로 실행하는 시스템들의 임의의 집합을 포함하는 것으로 사용될 수 있다.

[0087] 수집 시스템은 수집 프로세서(302)(예를 들어, 중앙 처리 장치(CPU)), 실행 동안 프로그램 코드를 저장할 수 있는 메모리(304) 및 디스크 저장부(306)를 포함할 수 있고, 이들 모두는 버스(300)를 통해 서로 통신한다. 시스템은 비디오 디스플레이 유닛(308)(예를 들어, 액정 디스플레이(LCD) 또는 음극선관(CRT))을 더 포함할 수 있다. 시스템은 또한 문자숫자식 입력 디바이스(310)(예를 들어, 키보드); 및 컨텐트 소스로부터 수신하고 컨텐트 저장부(312)로 전달하기 위한 네트워크 인터페이스 디바이스를 포함할 수 있다.

[0088] 디스크 저장 유닛(306)은, 본 명세서에서 설명되는 방법들 또는 기능들 중 임의의 하나 또는 그 초과를 구현하는 명령들(예를 들어, 소프트웨어) 중 하나 또는 그 초과의 세트들이 저장될 수 있는 머신-판독가능 매체를 포함할 수 있다. 명령들은 또한 시스템에 의한 실행 동안 메모리(304) 내에 및/또는 수집 프로세서(302) 내에 완전히 또는 적어도 부분적으로 상주할 수 있고, 메모리(304) 및 수집 프로세서(302)는 또한 머신-판독가능 매체를 구성한다.

[0089] 도 4에 도시된 바와 같이, 클라이언트 시스템은 다양한 실시예들에 따라 하드웨어 및 소프트웨어 컴포넌트들

의 조합으로서 구현될 수 있다. 클라이언트 시스템은, 이 시스템으로 하여금 본 명세서에서 설명되는 방법들 중 임의의 하나 또는 그 초과를 수행하도록 실행될 수 있는 명령들의 세트를 포함할 수 있다. 시스템은 컴퓨터의 형태로 특수한 머신으로서 실현될 수 있다. 시스템은 서버 컴퓨터, 개인용 컴퓨터(PC), 또는 그 시스템에 의해 행해지는 동작들을 특정하는 명령들의 세트를 (순차적으로 또는 다른 방식으로) 실행할 수 있는 임의의 시스템일 수 있다. 추가적으로, 오직 단일 시스템이 도시되지만, 용어 "시스템"은 또한, 본 명세서에서 설명되는 방법들 중 임의의 하나 또는 그 초과를 수행하는 명령들의 세트(또는 다수의 세트들)를 개별적으로 또는 공동으로 실행하는 시스템들의 임의의 집합을 포함하는 것으로 사용될 수 있다.

[0090] 클라이언트 시스템은 클라이언트 프로세서(402)(예를 들어, 중앙 처리 장치(CPU)), 실행 동안 프로그램 코드를 저장할 수 있는 메모리(404) 및 디스크 저장부(406)를 포함할 수 있고, 이들 모두는 버스(400)를 통해 서로 통신한다. 시스템은 비디오 디스플레이 유닛(408)(예를 들어, 액정 디스플레이(LCD) 또는 음극선관(CRT))을 더 포함할 수 있다. 시스템은 또한 문자숫자식 입력 디바이스(410)(예를 들어, 키보드) 및 요청들을 전송하고 응답들을 수신하기 위한 네트워크 인터페이스 디바이스(412)를 포함할 수 있다.

[0091] 디스크 저장 유닛(406)은, 본 명세서에서 설명되는 방법들 또는 기능들 중 임의의 하나 또는 그 초과를 구현하는 명령들(예를 들어, 소프트웨어) 중 하나 또는 그 초과의 세트들이 저장될 수 있는 머신-판독가능 매체를 포함할 수 있다. 명령들은 또한 시스템에 의한 실행 동안 메모리(404) 내에 및/또는 수집 프로세서(402) 내에 완전히 또는 적어도 부분적으로 상주할 수 있고, 메모리(404) 및 수집 프로세서(402)는 또한 머신-판독가능 매체를 구성한다.

[0092] 3GPP 파일 포맷의 이용

[0093] 3GPP 파일 포맷, 또는 MP4 파일 포맷 또는 3GPP2 파일 포맷과 같은 ISO 기반 미디어 파일 포맷에 기초한 임의의 다른 파일은 하기 특성들을 갖는 HTTP 스트리밍에 대한 컨테이너 포맷으로서 이용될 수 있다. 클라이언트가 요구에 따라 파일들 또는 미디어 세그먼트들의 적절한 피스(piece)들을 다운로드할 수 있도록, 시간 오프셋들 및 바이트 범위들을 시그널링하기 위해 각각의 세그먼트에 세그먼트 인덱스가 포함될 수 있다. 전체 미디어 프리젠테이션의 글로벌 프리젠테이션 타이밍 및 각각의 3GP 파일 또는 미디어 세그먼트 내의 로컬 타이밍은 정확하게 정렬될 수 있다. 하나의 3GP 파일 또는 미디어 세그먼트 내의 트랙들은 정확하게 정렬될 수 있다. 리프리젠테이션에 걸친 스위칭이 끊김없이 될 수 있고 상이한 리프리젠테이션들 내의 미디어 컴포넌트들의 공동 프리젠테이션이 동기식이 될 수 있도록, 리프리젠테이션들에 걸친 트랙들은 또한, 이들 각각을 글로벌 타임라인에 할당함으로써 정렬될 수 있다.

[0094] 파일 포맷은 하기 특성들을 갖는 적응형 스트리밍에 대한 프로파일을 포함할 수 있다. 모든 영화 데이터가 영화 프래그먼트들에 포함될 수 있고, ? "moov" 박스는 임의의 샘플 정보를 포함하지 않을 수 있다. TS26.244에서 특정되는 것과 같은 진보된 다운로드 프로파일에 대한 것과 유사한 요건들로, 오디오 및 비디오 샘플 데이터는 인터리빙될 수 있다. "moov" 박스는 파일의 시작부에 배치될 수 있고, 세그먼트 인덱스로도 지칭되는 프래그먼트 오프셋 데이터가 이에 후속하고, 프래그먼트 오프셋 데이터는, 각각의 프래그먼트에 대한 시간에서의 오프셋 정보 및 바이트 범위들 또는 그 포함하는 세그먼트의 프래그먼트들의 적어도 서브세트를 포함한다.

[0095] 미디어 프리젠테이션 디스크립션은 기존의 진보된 다운로드 프로파일에 후속하는 파일들을 참조하는 것이 또한 가능할 수 있다. 이 경우, 클라이언트는 다수의 이용가능한 버전들 중 적절한 대안적 버전을 단순히 선택하기 위해 미디어 프리젠테이션 디스크립션을 이용할 수 있다. 클라이언트들은 또한, 각각의 대안적 버전의 서브세트들을 요청하기 위해 진보된 다운로드 프로파일에 따르는 파일에 의한 HTTP 부분 획득 요청들을 이용하여, 덜 효율적인 형태의 적응형 스트리밍을 구현할 수 있다. 이 경우, 진보된 다운로드 프로파일에 미디어를 포함시키는 상이한 리프리젠테이션들은 여전히 공통 글로벌 타임라인을 준수하여, 리프리젠테이션들에 걸쳐 끊김없는 스위칭을 가능하게 할 수 있다.

[0096] 개선된 방법들의 개관

[0097] 하기 섹션들에서, 개선된 블록-요청 스트리밍 시스템들을 위한 방법이 설명된다. 이 개선들 중 일부는 본 출원의 요구들에 따라 이 개선들의 다른 부분들과 함께 또는 다른 부분들없이 이용될 수 있음을 이해해야 한다. 일반적 동작에서, 수신기는 데이터의 특정 블록들 또는 블록들의 일부들에 대해 서버 또는 다른 전송기들에

요청들을 행한다. 또한 세그먼트들로도 지칭되는 파일들은 다수의 블록들을 포함할 수 있고, 미디어 프리젠테이션의 하나의 리프리젠테이션과 연관된다.

[0098] 바람직하게는, 플레이아웃 또는 디코딩 시간들로부터 세그먼트 내의 대응하는 블록들 또는 프래그먼트들의 바이트 오프셋들로의 매핑을 제공하는, "세그먼트 인덱싱" 또는 "세그먼트 맵"으로도 또한 지칭되는 인덱싱 정보가 생성된다. 이 세그먼트 인덱싱은 세그먼트 내에서 전형적으로 세그먼트의 시작부에 포함될 수 있고(세그먼트 맵의 적어도 일부가 시작부에 있음), 종종 작다. 세그먼트 인덱스는 또한 별개의 인덱스 세그먼트 또는 파일에서 제공될 수 있다. 특히, 세그먼트 인덱스가 세그먼트에 포함되는 경우들에서, 수신기는 이 세그먼트 맵의 일부 또는 전부를 신속하게 다운로드할 수 있고, 후속적으로 이를 이용하여, 시간 오프셋들과, 파일 내의 그 시간 오프셋들과 연관된 프래그먼트들의 대응하는 바이트 위치들 사이의 매핑을 결정할 수 있다.

[0099] 수신기는, 관심있는 시간 오프셋들과 연관되지 않는 다른 프래그먼트들과 연관된 데이터의 전부를 다운로드할 필요없이, 특정한 시간 오프셋들과 연관된 프래그먼트들로부터 데이터를 요청하기 위한 바이트 오프셋을 이용할 수 있다. 이 방식으로, 세그먼트 맵 또는 세그먼트 인덱싱은, 개선된 컨텐트 재핑 시간들, 네트워크 조건들이 변할 때 하나의 리프리젠테이션으로부터 다른 리프리젠테이션으로 신속하게 변경할 수 있는 능력 및 수신기에서 플레이아웃되지 않는 미디어를 다운로드하는 네트워크 자원들의 낭비의 감소를 포함하는 이점들과 함께, 관심있는 현재의 시간 오프셋들과 관련된 세그먼트의 부분들에 수신기가 직접 액세스할 수 있는 능력을 크게 개선할 수 있다.

[0100] 하나의 리프리젠테이션(본 명세서에서는 "스위칭-전" 리프리젠테이션으로 지칭됨)으로부터 다른 리프리젠테이션(본 명세서에서는 "스위칭-후" 리프리젠테이션으로 지칭됨)으로의 스위칭이 고려되는 경우, 세그먼트 인덱스가 또한 이용되어, 스위칭-후 리프리젠테이션의 플레이아웃이 랜덤 액세스 포인트로부터 끊김없이 시작할 수 있도록, 스위칭-전 리프리젠테이션의 미디어가 프리젠테이션 시간까지 다운로드된다는 관점에서 끊김없는 스위칭이 가능한 것을 보장하기 위해, 스위칭-전 리프리젠테이션에서 요청될 데이터량을 식별하기 위해 스위칭-후 리프리젠테이션에서 랜덤 액세스 포인트의 시작 시간을 식별할 수 있다.

[0101] 이 블록들은, 수신기의 사용자를 위한 출력을 생성하기 위해, 요청하는 수신기가 요구하는 비디오 미디어 또는 다른 미디어의 세그먼트들을 표현한다. 미디어의 수신기는, 컨텐트를 전송하는 서버로부터 수신기가 컨텐트를 수신할 때와 같이, 클라이언트 디바이스일 수 있다. 예들은, 셋탑 박스들, 컴퓨터들, 게임 콘솔들, 특수하게 장착된 텔레비젼들, 핸드헬드 디바이스들, 특수하게 장착된 모바일 폰들, 또는 다른 클라이언트 수신기들을 포함한다.

[0102] 다수의 개선된 버퍼 관리 방법들이 본 명세서에 설명된다. 예를 들어, 버퍼 관리 방법은 클라이언트들로 하여금, 시간상 연속적으로 플레이아웃되도록 수신될 수 있는 가장 높은 미디어 품질의 블록들을 요청하게 한다. 가변적 블록 사이즈 특징은 압축 효율을 개선시킨다. 요청들의 빈도를 제한하면서, 요청하는 디바이스에 블록들을 전송하기 위해 다수의 접속들을 갖는 능력은 개선된 전송 성능을 제공한다. 데이터의 부분적으로 수신된 블록들은 미디어 프리젠테이션을 계속하는데 이용될 수 있다. 접속은, 블록들의 특정한 세트에 대한 시작 시에 접속을 실행할 필요없이, 다수의 블록들에 대해 재사용될 수 있다. 다수의 클라이언트들에 의한 다수의 가능한 서버들로부터 서버들의 선택에서의 일관성이 개선되고, 이것은, 인근 서버들에서 복제된 컨텐트의 빈도를 감소시키고, 서버가 전체 파일을 포함할 가능성을 개선시킨다. 클라이언트들은, 미디어 블록들을 포함하는 파일들에 대한 URL들에 내장된 (이용가능한 미디어 인코딩들과 같은) 메타데이터에 기초하여 미디어 블록들을 요청할 수 있다. 시스템은, 미디어 플레이아웃에서 후속적인 일시중지들을 유발시키지 않고 컨텐트의 플레이아웃이 시작할 수 있기 전에 요구되는 버퍼링 시간량의 계산 및 최소화를 제공할 수 있다. 이용가능한 대역폭은 다수의 미디어 블록들 사이에서 공유되고, 각각의 블록의 플레이아웃 시간이 접근할 때 조정될 수 있어서, 필요하다면, 가장 가까운 플레이아웃 시간을 갖는 블록 쪽으로 이용가능한 대역폭의 더 큰 공유가 할당될 수 있다.

[0103] HTTP 스트리밍은 메타데이터를 이용할 수 있다. 프리젠테이션 레벨 메타데이터는, 예를 들어, 스트림 지속시간, 이용가능한 인코딩들(비트레이트들, 코덱들, 공간적 해상도들, 프레임 레이트들, 언어, 미디어 유형들), 각각의 인코딩에 대한 스트림 메타데이터에 대한 포인터들, 및 컨텐트 보호(디지털 권리 관리(DRM) 정보)를 포함한다. 스트림 메타데이터는 세그먼트 파일들에 대한 URL들일 수 있다.

[0104] 세그먼트 메타데이터는 랜덤 액세스 포인트(RAP)들 또는 다른 탐색 포인트들의 식별자 및 세그먼트 내의 요청들에 대한 바이트 범위 대 시간 정보를 포함할 수 있고, 여기서, 이 정보의 일부 또는 전부는 세그먼트 인덱싱 또는 세그먼트 맵의 일부일 수 있다.

[0105] 스트림들은 동일한 컨텐트의 다수의 인코딩들을 포함할 수 있다. 다음으로, 각각의 인코딩은 세그먼트들로 분해될 수 있고, 여기서, 각각의 세그먼트는 저장 단위 또는 파일에 대응한다. HTTP의 경우, 세그먼트는 전형적으로, URL에 의해 참조될 수 있는 자원이고, 이러한 URL의 요청은, 요청 응답 메시지의 엔티티 바디로서 세그먼트의 리턴을 초래한다. 세그먼트들은 다수의 영상 그룹(GoP)들을 포함할 수 있다. 각각의 GoP는 다수의 프래그먼트들을 더 포함할 수 있고, 여기서, 세그먼트 인덱싱은 각각의 프래그먼트에 대한 시간/바이트-오프셋 정보를 제공하는데, 즉, 인덱싱의 단위가 프래그먼트이다.

[0106] 프래그먼트들 또는 프래그먼트들 중 일부들이 스루풋을 증가시키기 위해 병렬 TCP 접속들을 통해 요청될 수 있다. 이는 병목(bottleneck) 링크 상에서 접속들을 공유할 때 또는 정체(congestion)로 인해 접속들이 손실(lost)될 때 발생하는 문제들을 경감시킬 수 있어서 전달의 전체 속도 및 안정성을 증가시키며, 이는 컨텐트 잽핑(zapping) 시간의 속도 및 안정성을 실질적으로 향상시킨다. 대역폭은 과다 요청(over-requesting)에 의한 지연과 교환(trade)될 수 있지만, 고갈의 위험을 증가시킬 수 있는 너무 먼 장래까지 요청하는 것을 방지하기 위해 주의를 기울여야 한다.

[0107] 동일한 서버 상의 세그먼트들에 대한 다수의 요청들이 TCP 시작 지연들을 회피하기 위해 파이프라이닝(현재 요청이 완료되기 전에 다음 요청을 함)될 수 있다. 연속적인 프래그먼트들에 대한 요청들은 하나의 요청으로 애그리게이팅(aggregated)될 수 있다.

[0108] 일부 CDN들은 큰 파일들을 선호하며, 범위 요청을 처음 접할 때 근원(origin) 서버로부터 전체 파일의 백그라운드 페치들을 트리거링할 수 있다. 그러나 대부분의 CDN들은 데이터가 이용가능한 경우 캐시로부터 범위 요청들을 서빙할 것이다. 따라서, 클라이언트 요청들 중 일부가 전체 세그먼트 파일을 위한 것이 되게 하는 것은 유리할 수 있다. 이러한 요청들은 이후에 필요에 따라 취소(cancel)될 수 있다.

[0109] 유효한 스위치 포인트들은 탐색 포인트들, 특히 예를 들어, 타겟 스트림에서 RAP들일 수 있다. (미디어의 시작에 기초하여 또는 GoP들에 기초하여) 스트림들에 걸친 RAP들의 정렬 또는 고정된 GoP 구조와 같은 상이한 구현들이 가능하다.

[0110] 일 실시예에서, 세그먼트들 및 GoP들은 상이한 레이트 스트림들에 걸쳐 정렬될 수 있다. 이러한 실시예에서, GoP들은 가변 크기일 수 있고 다수의 프래그먼트들을 포함할 수 있지만, 프래그먼트들은 상이한 레이트 스트림들 사이에서 정렬되지 않는다.

[0111] 일부 실시예들에서, 파일 리던던시가 이익을 얻기 위해 사용될 수 있다. 이러한 실시예들에서, 삭제(erasure) 코드가 각각의 프래그먼트에 적용되어 데이터의 리던던트 버전들을 생성한다. 바람직하게, 소스 포맷팅은 FEC의 사용으로 인해 변경되지 않으며, 예를 들어, FEC 보수 데이터를 포함하는, 본래의(original) 리프리젠테이션의 종속적인 리프리젠테이션으로서 추가의 보수 세그먼트들이 수집 시스템에서 추가의 단계로서 생성되고 이용가능하게 된다. 상기 프래그먼트에 대해 소스 데이터만을 이용하는 프래그먼트를 재구성할 수 있는 클라이언트는 서버들로부터 세그먼트 내의 프래그먼트에 대해 소스 데이터를 요청할 수만 있다. 서버들이 이용가능하지 않거나 서버들로의 접속들이 늦으면 ?이는 소스 데이터에 대한 요청 이전 또는 이후에 결정될 수 있음?, 추가의 보수 데이터가 보수 세그먼트로부터의 프래그먼트에 대해 요청될 수 있으며, 이는 프래그먼트의 소스 데이터를 복원하기 위해 수신된 소스 및 보수 데이터의 조합을 사용하기 위해 가능하게는 FEC 디코딩을 이용하여 프래그먼트를 복원하기에 충분한 데이터를 신뢰가능하게 전달하는 시간을 감소시킨다. 더욱이, 프래그먼트가 긴급하게 된 경우, 즉 프래그먼트의 플레이아웃 시간이 임박한 경우, 추가의 보수 데이터가 프래그먼트의 복원을 가능하게 하도록 요청될 수 있으며, 이는 링크 상의 상기 프래그먼트에 대한 데이터 공유를 증가시키지만, 대역폭을 넓히기(free up) 위해 링크 상의 다른 접속들을 종결하는 것보다 효율적이다. 이는 또한 병렬 접속들의 사용으로부터의 고갈의 위험을 경감시킬 수 있다.

[0112] 프래그먼트 포맷은 실시간 트랜스포트 제어 프로토콜(RTCP)을 통해 달성되는 오디오/비디오 동기화를 갖는 실시간 트랜스포트 프로토콜(RTP) 패킷들의 저장된 스트림일 수 있다.

[0113] 세그먼트 포맷은 또한 오디오/비디오 동기 달성된 MPEG-2 TS 내부 타이밍을 갖는 MPEG-2 TS 패킷들의 저장된 스트림일 수 있다.

[0114] <u>스트리밍을 더 효율적으로 만들기 위한 시그널링 및/또는 블록 생성의 사용</u>

[0115] 다수의 특징들이 개선된 성능을 제공하기 위해 블록 요청 스트리밍 시스템에서 사용될 수 있거나 그렇지 않을

수 있다. 성능은 클라이언트, 스톨링(stall), 대역폭 제약들 내에서 미디어 데이터의 획득, 및/또는 서버 및/또는 수집 시스템에서 제한된 프로세서 리소스들 내에서 그렇게 하지 않고 프리젠테이션을 플레이아웃하는 능력과 관련될 수 있다. 이러한 특징들 중 일부가 이제 설명될 것이다.

[0116] 세그먼트들 내에서의 인덱싱

[0117] 영화 프래그먼트들에 대한 부분적인 GET 요청들을 포뮬레이팅하기 위해, 클라이언트에는, 파일 또는 세그먼트 내의 프래그먼트들에 포함된 모든 미디어 컴포넌트들의 프리젠테이션 시간 또는 디코딩에서의 시작 시간 및 바이트 오프셋이 통지되며, 또한 상기 프래그먼트들은 랜덤 액세스 포인트들 시작하거나 포함하며(그리고 대안적인 리프리젠테이션들 사이에서 스위치 포인트들로서 사용되는 것이 또한 적절함), 여기서 이러한 정보는 종종 세그먼트 인덱싱 또는 세그먼트 맵으로 지칭된다. 디코딩 또는 프리젠테이션 시간에서 시작 시간은 직접적으로 리프리젠테이션될 수 있거나, 기준 시간과 관련하여 델타들로서 리프리젠테이션될 수 있다.

[0118] 이러한 시간 및 바이트 오프셋 인덱싱 정보는 영화 프래그먼트당 적어도 8 바이트 데이터를 필요로 할 수 있다. 예로써, 500ms 영화 프래그먼트들을 가진 단일 파일 내에 포함된 두 시간짜리 영화의 경우, 이는 총 대략 112 킬로바이트의 데이터일 것이다. 프리젠테이션의 시작 시 이러한 데이터 모두를 다운로딩하는 것은 현저한 추가의 구동 지연을 초래할 수 있다. 그러나 시간 및 바이트 오프셋 데이터는 계층적으로 인코딩될 수 있어서, 클라이언트는 시작하려고 하는 프리젠테이션에서의 포인트에 적절한 작은 크기의 시간 및 오프셋 데이터를 신속하게 발견할 수 있다. 정보는 또한 세그먼트 내에서 분배될 수 있어서 세그먼트 인덱스의 일부 개선이 미디어 데이터와 인터리빙되어 위치설정될 수 있다.

[0119] 리프리젠테이션이 다수의 세그먼트들로 시간에 대해 세그먼트화되면, 각각의 세그먼트에 대한 완전한 시간 및 오프셋 데이터가 이미 상당히 작기 때문에 이러한 계층적 코딩의 사용은 필수적이지 않을 수 있음을 주목해야 한다. 예를 들어, 세그먼트들이 전술한 예에서 두 시간 대신에 1분인 경우, 시간 바이트 오프셋 인덱싱 정보는 약 1킬로바이트의 데이터이며, 이는 단일 TCP/IP 패킷 내에서 전형적으로 적합할 수 있다.

[0120] 상이한 옵션들이 프래그먼트 시간 및 바이트 오프셋 데이터를 3GPP 파일에 부가하기 위해 가능하다:

[0121] 첫 번째, 영화 프래그먼트 랜덤 액세스 박스("MFRA")가 이러한 목적을 위해 사용될 수 있다. MFRA는 영화 프래그먼트들을 이용하여 파일에서 랜덤 액세스 포인트들을 찾는데 있어서 독자들을 지원할 수 있는 테이블을 제공한다. 이러한 기능의 지원에서, MFRA는 랜덤 액세스 포인트들을 포함하는 MFRA 박스들의 바이트 오프셋들을 부수적으로 포함한다. MFRA는 파일의 종단에 또는 종단 부분에 위치설정될 수 있지만, 이는 반드시 그러한 것은 아니다. 영화 프래그먼트 랜덤 액세스 오프셋 박스에 대한 파일의 종단으로부터의 스캐닝 및 그 내부의 크기 정보를 이용함으로써, 영화 프래그먼트 랜덤 액세스 박스의 시작을 위치설정할 수 있다. 그러나 HTTP 스트리밍에 대한 종단에 MFRA를 위치설정하는 것은 원하는 데이터를 액세스하기 위해 전형적으로 3-4 HTTP 요청들을 필요로 한다: 파일의 종단으로부터 MFRA를 요청하기 위한 적어도 하나, MFRA를 획득하기 위한 하나, 그리고 마지막으로 파일에서 원하는 프래그먼트를 획득하기 위한 하나. 따라서, MFRA가 단일 요청에서 제 1 미디어 데이터와 함께 다운로드되기 때문에 시작에서 위치설정하는 것은 바람직할 수 있다. 또한, HTTP 스트리밍을 위해 MFRA를 이용하는 것은 비효율적일 수 있는데, 그 이유는 "MFRA"에서의 정보 중 어떠한 것도 시간 및 moof_offset과 별개로 요구되지 않으며, 길이들 대신에 오프셋들을 특정하는 것은 더 많은 비트들을 필요로 하기 때문이다.

[0122] 두 번째, 아이템 로케이션 박스("ILOC")가 사용될 수 있다. "ILOC"는 메타데이터 자원들이 포함하는 파일, 그 파일 내에서의 이들의 오프셋, 및 이들의 길이를 위치설정함으로써 이러한 또는 다른 파일들에 메타데이터 리소스들의 디렉토리를 제공한다. 예를 들어, 시스템은 외부적으로 참조되는 모든 메타데이터 리소스들을 하나의 파일로 통합해서, 이에 따라 파일 오프셋들 및 파일 참조를 재조정할 수 있다. 그러나 "ILOC"는 메타데이터의 위치를 제공하기 위해 의도되며, 따라서 ILOC가 실제 데이터와 공존하는 것은 어려울 수 있다.

[0123] 마지막은, 그리고 아마도 가장 적절하게, 효율적 방식으로 정확한 프래그먼트 시간들 또는 지속 기간 및 바이트 오프셋을 제공할 목적으로 특히 전용되는, 시간 인덱스 박스("TIDX")로서 지칭되는 새로운 박스의 기술이다. 이는 다음 섹션에서 더욱 상세하게 설명된다. 동일한 기능들을 갖는 대안적인 박스는 세그먼트 인덱스 박스("SIDX")일 수 있다. 여기서, 다르게 지시되지 않는다면, 두 박스들이 효율적인 방식으로 정확한 프래그먼트 시간들 또는 지속 기간들 및 바이트 오프셋을 제공하는 능력을 제공할 수 있기 때문에, 이러한 두 박스는 서로 교환가능하다. TIDX와 SIDX 사이의 차이는 이하에서 제공된다. 두 박스들이 세그먼트 인덱스를 구

현하기 때문에, TIDX 박스들과 SIDX 박스들을 어떻게 교환하는 지는 명확할 것이다.

[0124] 세그먼트 인덱싱

[0125] 세그먼트는 식별된 시작 시간 및 식별된 수의 바이트를 갖는다. 다수의 프래그먼트들이 단일 세그먼트로 연쇄될 수 있으며, 클라이언트는 프래그먼트의 서브셋 또는 요구되는 프래그먼트에 대응하는 세그먼트 내에서 특정 바이트 범위를 식별하는 요청들을 발행할 수 있다. 예를 들어, HTTP가 요청된 프로토콜로서 사용되면, HTTP 범위 헤더가 이러한 목적을 위해 사용될 수 있다. 이러한 방식은 상이한 프래그먼트들의 세그먼트 내의 위치를 특정하는 세그먼트의 "세그먼트 인덱스"에 클라이언트가 액세스하는 것을 필요로 한다. 이러한 "세그먼트 인덱스"는 메타데이터의 일부로서 제공될 수 있다. 이러한 방식은 각 블록마다 개별 파일에 보관되는 방식과 비교하여 훨씬 더 적은 파일들이 생성되고 관리될 필요가 있는 결과를 갖는다. (이를 테면, 1시간 프리젠테이션을 위해 수천 개까지 확장할 수 있는) 매우 큰 수의 파일들의 생성, 전달 및 저장의 관리는 복잡하고 에러가 발생하기 쉬울 수 있으며, 따라서 파일들의 수의 감소는 장점을 나타낸다.

[0126] 클라이언트가 세그먼트의 더 작은 부분의 원하는 시작 시간을 단지 알고 있는 경우, 클라이언트는 전체 파일을 요청할 수 있으며, 그래서 적절한 플레이아웃 시작 위치를 결정하기 위해 파일 전체를 판독할 수 있다. 대역폭 사용을 개선하기 위해, 세그먼트들은 메타데이터로서 인덱스 파일을 포함할 수 있고, 여기서 인덱스 파일은 개별 블록들의 바이트 범위들을 블록들이 대응하는 시간 범위들과 맵핑시키며, 세그먼트 인덱싱 또는 세그먼트 맵으로 불린다. 이러한 메타데이터는 XML 데이터로서 포맷팅될 수 있고, 이들은 예를 들어, 3GPP 파일 포맷의 아톰(atom) 및 박스 구조를 따르는 이진수일 수 있다. 인덱싱은 단순할 수 있는데, 여기서 각 블록의 시간 및 바이트 범위는 파일의 시작에 관하여 절대적이거나, 또는 이들은 계층적일 수 있으며, 여기서 일부 블록들은 부모 블록들로 그룹화되고(부모 블록들은 조부모 블록들로 그룹화되는 등), 주어진 블록에 대한 시간 및 바이트 범위는 블록의 부모 블록의 시간 및/또는 바이트 범위에 관하여 리프리젠테이션된다.

[0127] 예시적인 인덱싱 맵 구조

[0128] 일 실시예에서, 미디어 스트림의 하나의 리프리젠테이션을 위한 본래의 소스 데이터가 본 명세서에서 "미디어 세그먼트"로 불리는 하나 이상의 미디어 파일들에 포함될 수 있으며, 여기서 각각의 미디어 세그먼트는 미디어의 연속한 시간 세그먼트, 예를 들어, 5분의 미디어 플레이백을 플레이백하기 위해 사용되는 미디어 데이터를 포함한다.

[0129] 도 6은 미디어 세그먼트의 예시적인 전체 구조를 도시한다. 시작 또는 소스 세그먼트 전체에 확산된 각각의 세그먼트 내에서, 인덱싱 정보가 존재할 수 있으며, 이는 시간/바이트-오프셋 세그먼트 맵을 포함한다. 일 실시예에서, 시간/바이트-오프셋 세그먼트 맵은 시간/바이트-오프셋 쌍들((T(0),B(0)), (T(1),B(1)), (T(i),B(i)),...,(T(n),B(n)))의 리스트일 수 있으며, 여기서, T(i-1)은 모든 미디어 세그먼트들 중에서 미디어의 초기 시작 시간에 관한 미디어의 i 번째 프래그먼트의 플레이백을 위한 세그먼트 내의 시작 시간을 나타내며, T(i)는 i 번째 프래그먼트에 대한 종료 시간(및 따라서 다음 프래그먼트에 대한 시작 시간)을 나타내며, 바이트-오프셋 B(i-1)은 이러한 소스 세그먼트 내에서 데이터의 시작의 대응하는 바이트 인덱스이며, 여기서, 미디어의 i 번째 프래그먼트는 소스 세그먼트의 처음에 관하여 시작되며, B(i)는 i 번째 프래그먼트의 대응하는 종료 바이트 인덱스(및 따라서 다음 프래그먼트의 첫 번째 바이트의 인덱스)이다. 세그먼트가 다수의 미디어 컴포넌트들을 포함하면, T(i) 및 B(i)는 절대적인 방식으로 세그먼트에서 각각의 컴포넌트에 대해 제공될 수 있거나 또는 이들은 기준 미디어 컴포넌트를 서빙하는 다른 미디어 컴포넌트에 관하여 리프리젠테이션될 수 있다.

[0130] 이러한 실시예에서, 소스 세그먼트에서 프래그먼트들의 수는 n 이며, 여기서, n은 세그먼트마다 변화할 수 있다.

[0131] 다른 실시예에서, 각각의 세그먼트에 대한 세그먼트 인덱스에서의 시간 오프셋은 제 1 프래그먼트의 절대적인 시작 시간과 각각의 세그먼트에 대한 지속 기간들로 결정될 수 있다. 이러한 경우, 세그먼트 인덱스는 세그먼트에 포함되는 모든 프래그먼트들의 지속 기간 및 제 1 프래그먼트의 시작 시간을 기록(document)할 수 있다. 세그먼트 인덱스는 또한 오직 프래그먼트들의 서브셋을 기록할 수 있다. 이러한 경우, 세그먼트 인덱스는, 포함하는 세그먼트의 종단에서 또는 다음 서브세그먼트의 처음에서 종료하는 하나 또는 그 초과의 연속한 프래그먼트들로서 정의되는 서브세그먼트의 지속 기간을 기록한다.

[0132] 각각의 프래그먼트에 대해, 프래그먼트가 탐색 포인트, 즉 상기 포인트 이후에 어떠한 미디어도 상기 포인트에 앞서 임의의 미디어에 의존하지 않는 포인트에서 시작하는지 또는 탐색 포인트를 포함하는지를 결정하는 값이 존재할 수 있고, 따라서 상기 프래그먼트 포워드로부터의 미디어는 앞선 프래그먼트와 독립적으로 플레이아웃될 수 있다. 탐색 포인트들은 일반적으로, 플레이아웃이 모든 앞선 미디어에 독립적으로 시작할 수 있는 이디어의 포인트들이다. 도 6은 또한 소스 세그먼트에 대한 가능한 세그먼트 인덱싱의 단순한 예를 도시한다. 이 예에서, 시간 오프셋 값은 밀리초 단위이며, 따라서, 이러한 소스 세그먼트의 제 1 프래그먼트는 미디어의 처음으로부터 20초에서 시작하며, 제 1 프래그먼트는 485밀리초의 플레이아웃 시간을 갖는다. 제 1 프래그먼트의 시작의 바이트 오프셋은 1이며, 제 2 프래그먼트의 제 1 프래그먼트/시작의 종료의 바이트 오프셋은 50,245이며, 제 1 프래그먼트는 50,245 바이트의 크기이다. 프래그먼트 또는 서브세그먼트가 랜덤 액세스 포인트에서 시작하지 않지만, 랜덤 액세스 포인트는 프래그먼트 또는 서브세그먼트에 포함되면, 디코딩 시간 또는 시작 시간과 실제 RAP 시간 사이의 프리젠테이션 시간 차가 주어질 수 있다. 이는 이러한 미디어 세그먼트로의 스위칭의 경우, 리프리젠테이션으로부터의 스위칭이 제공될 필요가 있을 때까지의 시간을 클라이언트가 정확하게 알 수 있게 한다.

[0133] 단순 또는 계층적 인덱싱에 부가적으로, 또는 이에 대신하여, 데이지 체인식(daisy-chained)의 인덱싱 및/또는 하이브리드 인덱싱이 사용될 수 있다.

[0134] 상이한 트랙들 동안의 샘플 지속 기간들이 동일하지 않을 수 있기 때문에(예를 들어, 비디오 샘플들은 33ms 동안 디스플레이될 수 있는 반면, 오디오 샘플은 80ms 간 지속될 수 있음), 영화 프래그먼트에서의 상이한 트랙들이 정확히 동일한 시간에 시작 및 종료하지 않을 수 있으며, 즉 보상을 위해 오디오는 비디오의 약간 앞 또는 약간 뒤에서 시작할 수 있고, 앞선 프래그먼트에서는 그 반대이다. 모호함을 방지하기 위해, 바이트 오프셋 데이터 및 시간에서 특정된 시간스탬프들이 특정 트랙에 대해 특정될 수 있으며, 이는 각각의 리프리젠테이션에 대해 동일한 트랙일 수 있다. 일반적으로 이는 비디오 트랙일 것이다. 이는 클라이언트가 리프리젠테이션들을 스위칭하고 있을 때 다음 비디오 프레임을 정확하게 식별하게 한다.

[0135] 상기한 문제에도 불구하고 오디오/비디오 동기화의 유지 및 유연한 플레이아웃을 보장하기 위해, 트랙 시간 스케일과 프리젠테이션 시간 사이의 엄격한 관계를 유지하도록 프리젠테이션 동안 주의가 취해질 수 있다.

[0136] 도 7은 단순 인덱스(700) 및 계층적 인덱스(702)와 같은 일부 예들을 도시한다.

[0137] 세그먼트 맵을 포함하는 박스의 두 특정 예는 이하에서 제공되는데, 하나는 시간 인덱스 박스('TIDX')로 지칭되고 하나는 ('SIDX')로 지칭된다. 정의는 ISO 기반 미디어 파일 포맷에 따른 박스 구조를 따른다. 동일한 시멘틱스 및 기능들을 갖고 유사한 신택스를 정의하기 위한 이러한 박스들에 대한 다른 설계들이 독자에게 명백할 것이다.

[0138] 시간 인덱스 박스

[0139] 정의(Definition)

[0140] 박스 타입(Box Tyoe): 'tidx'

[0141] 컨테이너(Container): 파일

[0142] 의무사항(Mandatory): 없음

[0143] 수량(Quantity): 임의의 수의 0 또는 1

[0144] 시간 인덱스 박스는 파일의 어떤 영역들을 프리젠테이션의 어떤 시간 간격들과 관련시키는 바이트 오프셋 인덱스들 및 시간의 세트를 제공한다. 시간 인덱스 박스는 targettype(타겟 타입) 필드를 포함할 수 있고, 이는 참조된(referenced) 데이터의 타입을 나타낸다. 예를 들어, 타겟 타입 "moof"를 갖는 시간 인덱스 박스는 시간 및 바이트 오프셋들 모두의 관점에서 필드에 포함되는 미디어 프래그먼트들로 인덱스를 제공한다. 시간 인덱스 박스의 타겟 타입을 갖는 시간 인덱스 박스는 계층적 시간 인덱스를 구성하도록 사용될 수 있어서, 필드의 사용자들이 신속하게 인덱스의 원하는 부분으로 네비게이팅하게 한다.

[0145] 세그먼트 인덱스는 예를 들어, 이하의 신택스를 포함한다:

```
aligned(8) class TimeIndexBox
extends FullBox('frai') {
unsigned int(32) targettype;
```

[0146]

```
unsigned int(32) time_reference_track_ID;
unsigned int(32) number_of_elements;
unsigned int(64) first_element_offset;
unsigned int(64) first_element_time;
for(i=1; i <= number_of_elements; i++)
{
 bit (1)        random_access_flag;
 unsigned int(31) length;
 unsigned int(32) deltaT;
 }
}
```

[0147]

[0148] 시맨틱스

[0149] targettype: 이러한 시간 인덱스 박스에 의해 참조되는 박스 데이터의 타입. 이는 영화 프래그먼트 헤더("moof") 또는 시간 인덱스 박스("tidx") 중 하나 일 수 있다.

[0150] time_reference_track_id: 이러한 인덱스에서 시간 오프셋들이 특정되는 관련된 트랙을 나타냄.

[0151] number_of_elements: 이러한 시간 인덱스 박스에 의해 인덱싱되는 엘리먼트들의 수.

[0152] first_element_offset: 제 1 인덱싱된 엘리먼트의 필드의 시작으로부터의 바이트 오프셋.

[0153] first_element_time: time_reference_track_id:에 의해 식별되는 트랙의 미디어 헤더 박스에서 특정되는 시간 스케일을 이용하는, 제 1 인덱싱된 엘리먼트의 시작 시간.

[0154] random_access_flag: 엘리먼트의 시작 시간이 랜덤 액세스 포인트인 경우 1. 그렇지 않은 경우 0.

[0155] length: 바이트 단위의 인덱싱된 엘리먼트의 길이.

[0156] deltaT: 이러한 엘리먼트의 시작 시간과 다음 엘리먼트의 시작 시간 사이의 time_reference_track_id에 의해 식별되는 트랙의 미디어 헤더 박스에서 특정되는 시간 스케일의 관점에서의 차이.

[0157] 세그먼트 인덱스 박스

[0158] 세그먼트 인덱스 박스('sidx')는 세그먼트에서 영화 프래그먼트들의 컴팩트 인덱스 및 다른 세그먼트 인덱스 박스들을 제공한다. 세그먼트 인덱스 박스에는 두 개의 루프 구조들이 존재한다. 제 1 루프는 서브세그먼트의 제 1 샘플, 즉 제 2 루프에 의해 참조되는 제 1 영화 프래그먼트에서의 샘플을 기록한다. 제 2 루프는 서브세그먼트의 인덱스를 제공한다. 'sidx' 박스를 위한 컨테이너는 바로 파일 또는 세그먼트이다.

[0159] 신택스

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

[0160]

[0161] 시맨틱스

[0162] reference_track_ID는 기준 트랙에 대한 track_ID를 제공한다.

[0163] track_count: 다음 루프에서 인덱싱되는 트랙들의 수(1 또는 그 초과);

[0164] reference_count: 제 2 루프에 의해 인덱싱되는 엘리먼트들의 수(1 또는 그 초과);

[0165] track_ID: 트랙의 ID로서, 이에 대해 트랙 프래그먼트가 이러한 인덱스에서 식별되는 제 1 영화 프래그먼트에 포함됨; 이러한 루프에서 정확하게 하나의 트랙 ID는 reference_track_ID와 동일함;

[0166] decoding_time: (트랙의 미디어 헤더 박스의 시간 스케일 필드에서 기록되는 바와 같이) 트랙의 시간 스케일에서 리프리젠테이션되는, 제 2 루프에서 제 1 아이템에 의해 참조되는 영화 프래그먼트에서의 track_ID에 의해 식별되는 트랙에서 제 1 샘플에 대한 디코딩 시간;

[0167] reference_type: 0으로 설정할 때, 참조가 영화 프래그먼트('moof') 박스에 대한 것임을 나타내고, 1로 설정될 때, 참조가 세그먼트 인덱스('sidx') 박스에 대한 것임을 나타냄;

[0168] reference_offset: 포함하는 세그먼트 인덱스 박스 이후의 제 1 바이트로부터 참조된 박스의 제 1 바이트까지의 바이트 단위의 거리;

[0169] subsegment_duration: 참조가 세그먼트 인덱스 박스에 대한 것일 때, 이 필드는 상기 박스의 제 2 루프의 subsegment_duration 필드들의 합을 전달; 참조가 영화 프래그먼트일 때, 이러한 필드는 루프에서 다음 엔트리에 의해 기록되는 제 1 영화 프래그먼트나 서브세그먼트의 종단 중 어느 쪽이든 더 이른 것까지 표시된 영화 프래그먼트 및 후속 영화 프래그먼트들에서, 기준 트랙의 샘플들의 샘플 지속 기간들의 합을 전달; 지속 기간은 (트랙의 미디어 헤더 박스의 시간 스케일 필드에 기록된 바와 같이) 트랙의 시간 스케일로 리프리젠테이션된다;

[0170] contains_RAP: 참조가 영화 프레임에 대한 것일 때, 참조 트랙 ID와 동일한 트랙 ID를 갖는 트랙에 대해 상기 영화 프래그먼트 내의 트랙 프래그먼트가 적어도 하나의 랜덤 액세스 포인트를 갖는 경우 이 비트는 1일 수 있고; 참조가 세그먼트 인덱스에 대한 것일 때, 세그먼트 인덱스의 임의의 참조들이 1로 설정된 이러한 비트를 갖기만 하면 이 비트는 1로 설정되며, 그렇지 않으면 0이다;

[0171] RAP_delta_time: contains_RAP가 1인 경우, 랜덤 액세스 포인트(RAP)의 리프리젠테이션(컴포지션) 시간을 제공함; contains_RAP가 0인 경우 값 0 으로 보존됨. 시간은 reference_track_ID와 동일한 track_ID를 갖는 트

랙에서, 이러한 엔트리에 의해 기록되는 세그먼트의 제 1 샘플의 디코딩 시간과 랜덤 액세스 포인트의 리프리젠테이션(컴포지션) 시간 사이의 차로서 리프리젠테이션된다.

[0172] TIDX와 SIDX 사이의 차

[0173] TIDX와 SIDX는 인덱싱에 관하여 동일한 기능을 제공한다. SIDX의 제 1 루프는 부가로 제 1 영화 프래그먼트에 대한 글로벌 타이밍을 제공하지만, 글로벌 타이밍은 또한 참조 트랙에 대해 절대적으로 또는 상대적으로 영화 프래그먼트 그 자체에 포함될 수 있다.

[0174] SIDX의 제 2 루프는 TIDX의 기능을 구현한다. 구체적으로, SIDX는 reference_type 의해 인용되는 각각의 인덱스에 대한 참조를 위해 타겟들의 혼합을 갖도록 허용하는 반면, TIDX는 오직 TIDX 또는 오직 MOOF 중 하나만을 참조한다. TIDX에서의 number_of_elements는 SIDX에서의 참조 카운트에 대응하며, TIDX에서의 time_reference_track_id는 SIDX에서의 reference_track_ID에 대응하며, TIDX에서의 first_element_offset은 제 2 루프의 제 1 엔트리에서의 reference_offset에 대응하며, TIDX에서의 first_element_time은 제 1 루프에서의 reference_track의 decoding_time에 대응하며, TIDX에서의 the random_access_flag는 SIDX에서 RAP가 프래그먼트의 시작에 필수적으로 위치하지 않을 수 있고 따라서 RAP_delta_time을 필요로 하는 추가의 자유도를 갖는 SIDX에서의 contains_RAP에 대응하며, TIDX에서의 길이는 SIDX에서의 reference_offset에 대응하며, 끝으로 TIDX에서의 deltaT는 SIDX에서의 subsegment_duration에 대응한다. 따라서, 두 박스들의 기능들은 등가이다.

[0175] 가변 블록 크기 설정 및 서브-GoP 블록들

[0176] 비디오 미디어의 경우, 요청들을 위한 블록 구조와 비디오 인코딩 구조 사이의 관계가 중요할 수 있다. 예를 들어, 각각의 블록이 탐색 포인트, 이를 테면, 랜덤 액세스 포인트("RAP")에서 시작하고, 각각의 블록이 비디오 시간의 동일한 지속 기간을 리프리젠테이션하면, 비디오 미디어의 적어도 일부의 탐색 포인트들의 위치(positioning)는 고정되고 탐색 포인트들은 비디오 인코딩 내에서 규칙적인 간격들로 발생할 것이다. 비디오 인코딩 기술 분야의 당업자에게 잘 열려져 있듯이, 탐색 포인트들이 비디오 프레임들 사이의 관계에 따라 배치되고, 특히 이들이 이전 프레임들과 공통점이 별로 없는 프레임들에 배치될 경우 압축 효율은 개선될 수 있다. 따라서, 블록들이 동일한 양의 시간을 나타내는 이러한 요건은 비디오 인코딩에 제한을 두어서, 압축이 준최적(suboptimal)이 될 수 있다.

[0177] 고정된 위치들에서의 탐색 포인트들을 요구하는 것보다, 비디오 프리젠테이션 내의 탐색 포인트들의 위치가 비디오 인코딩 시스템에 의해 선택되게 하는 것이 바람직하다. 비디오 인코딩 시스템이 탐색 포인트들을 선택가능 하게 하는 것은 개선된 비디오 압축을 초래하고 따라서 비디오 미디어의 더 높은 품질이 주어진 이용가능한 대역폭을 이용하여 제공될 수 있어서, 개선된 사용자 경험을 초래한다. 현재의 블록-요청 스트리밍 시스템들은 모든 블록들이 (비디오 시간에서) 동일한 지속 기간을 가지며, 각각의 블록이 탐색 포인트에서 시작해야 하는 것을 요구할 수 있으며, 따라서 이는 현존 시스템들의 장점이다.

[0178] 전술한 것에 비해 장점을 제공하는 신규한 블록-요청 스트리밍 시스템이 이제 설명된다. 일 실시예에서, 비디오 컴포넌트의 제 1 버전의 비디오 인코딩 프로세스는, 압축 효율을 최적화하기 위해 탐색 포인트들의 위치들을 선택하도록 구성될 수 있지만, 탐색 포인트들 사이의 지속 기간에 최대치가 존재한다는 요건을 갖는다. 후자의 요건은 인코딩 프로세스에 의한 탐색 포인트들의 선택을 제한하고 결국 압축 효율을 감소시킨다. 그러나 탐색 포인트들 사이의 지속 기간의 최대치가 너무 작지 않음(예를 들어, 약 1초 초과)을 전제로, 규칙적인 고정된 위치들이 탐색 포인트들에 대해 요구되면, 압축 효율의 감소는 초래된 효율에 비해 작다. 더욱이, 탐색 포인트들 사이의 지속 기간의 최대치가 수 초이면, 탐색 포인트들의 완전한 자유 위치와 비교하여 압축 효율에서의 감소는 일반적으로 매우 적다.

[0179] 본 실시예를 포함하는 많은 실시예들에서, 일부 RAP들은 탐색 포인트들이 아닐지 모르는데, 즉 예를 들어, RAP가 인근의 탐색 포인트들에 시간상으로 너무 근접하기 때문에, 또는 RAP에 선행 또는 후속하는 탐색 포인트와 RAP 사이의 미디어 데이터의 양이 너무 작기 때문에, 두 개의 연속한 탐색 포인트들 사이에 있고 탐색 포인트로 선택되지 않은 RAP인 프레임이 존재할 수 있다.

[0180] 미디어 프리젠테이션의 모든 다른 버전들 내의 탐색 포인트들의 위치는 제 1 (예를 들어, 가장 높은 미디어 데이터 레이트) 버전에서 탐색 포인트들과 동일하도록 제한될 수 있다. 이는 탐색 포인트들의 인코더 자유

선택을 허용하는 것과 비교하여 이러한 다른 버전에 대한 압축 효율을 감소시킨다.

[0181] 탐색 포인트들의 사용은 프레임이 독립적으로 디코딩 가능할 것을 전형적으로 요구하며, 이는 일반적으로 그 프레임에 대한 낮은 압축 효율을 초래한다. 독립적으로 디코딩 가능하도록 요구되지 않는 프레임들은 다른 프레임들의 데이터를 참조하여 인코딩될 수 있으며, 이는 일반적으로 인코딩될 프레임과 참조 프레임들 사이의 공통성의 양에 의존하는 양만큼 그 프레임에 대한 압축 효율을 증가시킨다. 탐색 포인트 위치의 효율적인 선택은 앞선 프레임들과 낮은 공통성을 갖는 프레임을 탐색 포인트 프레임으로서 선택하며, 그로 인해 독립적으로 디코딩 가능한 방식으로 프레임을 인코딩함으로써 유발되는 압축 효율 불이익을 최소화한다.

[0182] 그러나 프레임과 잠재적인 참조 프레임들 사이의 공통성의 레벨은, 본래의 컨텐트가 동일하기 때문에, 컨텐트의 상이한 리프리젠테이션들에 걸쳐 높게 상관된다. 결과로서, 제 1 변형에서 탐색 포인트들과 동일한 위치들이 되도록 다른 변형들에서 탐색 포인트들을 제한하는 것은 압축 효율에 큰 차이를 낳지 않는다.

[0183] 탐색 포인트 구조는 바람직하게 블록 구조를 결정하는데 사용된다. 바람직하게, 각각의 탐색 포인트는 블록의 시작을 결정했고, 연속한 탐색 포인트들 사이에서 데이터를 압축하는 하나 이상의 블록들이 존재할 수 있다. 탐색 포인트들 사이의 지속 기간이 우수한 압축성을 갖는 인코딩을 위해 고정되지 않기 때문에, 모든 블록들이 동일한 플레이아웃 지속 기간을 갖도록 요구되는 것은 아니다. 일부 실시예들에서, 블록들은 컨텐트의 버전들 사이에서 정렬되는데, 즉 컨텐트의 하나의 버전에서 특정 그룹의 프레임들에 걸치는 블록이 존재하면, 컨텐트의 다른 버전에서 동일한 그룹의 프레임들에 걸치는 블록이 존재한다. 컨텐트의 주어진 버전의 블록들은 중첩하지 않으며, 컨텐트의 프레임 하나 하나가 각각의 버전의 정확하게 하나의 블록 내에 포함된다.

[0184] 탐색 포인트들 사이의 가변 지속 기간들, 결국 가변 지속 기간 GoP들의 효율적인 사용을 가능하게 하는 인에이블 특징은 세그먼트에 포함될 수 있거나 다른 수단에 의해 클라이언트에 제공될 수 있는 세그먼트 인덱싱 또는 세그먼트 맵이며, 즉 이는 프리젠터이션의 각각의 블록의 시작 시간 및 지속 기간을 포함하는, 제공될 수 있는 이러한 리프리젠테이션의 이러한 세그먼트와 관련되는 메타데이터이다. 사용자가 세그먼트 내의 특정 포인트에서 프리젠테이션이 시작하게 요청한 경우 프리젠테이션을 시작할 블록을 결정할 때 클라이언트는 이러한 세그먼트 인덱싱 데이터를 사용할 수 있다. 이러한 메타데이터가 제공되지 않으면, (예를 들어, 시작 블록의 인덱스를 제공하기 위해 평균 블록 지속 기간에 의해 (시간상) 요청된 시작 포인트를 분할하여 시작 블록을 선택함으로써) 프리젠테이션은 컨텐트의 시작에서만 또는 원하는 포인트에 근접한 랜덤 또는 대략적인 포인트에서 시작될 수 있다.

[0185] 일 실시예에서, 각각의 블록은 개별 파일로서 제공될 수 있다. 다른 실시예에서, 다수의 연속한 블록들이 세그먼트를 형성하기 위해 단일 파일로 애그리게이팅될 수 있다. 이러한 제 2 실시예에서, 각각의 블록의 시작 시간 및 지속 기간 및 블록이 시작하는 파일 내의 바이트 오프셋을 포함하는 각각의 버전에 대한 메타데이터가 제공될 수 있다. 이러한 메타데이터는 초기 프로토콜 요청에 응답하여 제공될 수 있거나, 즉 세그먼트 또는 파일로부터 개별적으로 이용가능하거나, 예를 들어, 파일의 처음에서 블록들 그 자체로서 동일한 파일 또는 세그먼트 내에 포함될 수 있다. 기술 분야의 당업자에게 명백하듯이, 메타데이터를 클라이언트에게 전달하기 위해 요구되는 네트워크 리소스들을 감소시키기 위해, 이러한 메타데이터는 압축된 형태, 이를 테면 gzip 또는 델타 인코딩 또는 2진 형태로 인코딩될 수 있다.

[0186] 도 6은 세그먼트 인덱싱의 예를 도시하며, 여기서 블록들은 가변 크기이며, 블록들의 범위(scope)는 부분적인 GoP, 즉 하나의 RAP와 다음 RAP 사이의 미디어 데이터의 부분적인 양이다. 이러한 예에서, 탐색 포인트들은 RAP 표시자(indicator)에 의해 표시되며, 여기서 1의 RAP 표시자 값은 블록이 RAP 또는 탐색 포인트에서 시작하거나 RAP 또는 탐색 포인트를 포함하는 것을 나타내며, 0의 RAP 표시자는 블록이 RAP 또는 탐색 포인트를 포함하지 않음을 나타낸다. 이러한 예에서, 처음 3개의 블록들, 즉 바이트 0 내지 157,033은 제 1 GoP를 포함하며, 이는 1.623초의 프리젠테이션 지속 기간을 가지며, 컨텐트의 프리젠테이션 시간은 20초에서 21.623초까지 진행한다. 이러한 예에서, 처음 3개의 블록들 중 첫 번째는 .485초의 프리젠테이션 시간을 포함하며, 세그먼트에서 처음 50,245바이트의 미디어 데이터를 포함한다. 이러한 예에서, 블록4, 블록5 및 블록6은 제 2 GoP를 포함하며, 블록7 및 블록8은 제3 GoP를 포함하고, 블록9, 블록 10 및 블록 11은 제4 GoP를 포함한다. 탐색 포인트들로서 지정되지 않으며 따라서 세그먼트 맵에서 RAP들로서 표시되지 않는, 미디어 데이터에 다른 RAP들이 존재할 수 있음을 주목해야 한다.

[0187] 도 6을 다시 참조하면, 클라이언트 또는 수신기가 대략 22초의 시간 오프셋에서 미디어 프리젠테이션을 시작하는 컨텐트에 액세스하기를 원하면, 클라이언트는 관련된 미디어 데이터가 이러한 세그먼트 내에 있는 지를 우선 결정하기 위해 이하에서 더욱 상세하게 설명되는 MPD와 같은 다른 정보를 우선 사용할 수 있다. 클라이

언트는 세그먼트의 제 1 부분을 다운로드하여 세그먼트 인덱싱을 획득할 수 있으며, 이 경우, 세그먼트 인덱싱은 예를 들어, HTTP 바이트 범위 요청을 이용하는 단지 수 바이트이다. 세그먼트 인덱싱을 이용하여, 클라이언트는 자신이 다운로드해애 하는 제 1 블록이 기껏해야 22초이고 RAP, 즉 탐색 포인트에서 시작하는 시간 오프셋을 갖는 제 1 블록이라는 것을 결정할 수 있다. 이러한 예에서, 비록 블록5가 22초보다 작은 시간 오프셋을 가지만, 즉 블록5의 시간 오프셋은 21.965초이지만, 세그먼트 인덱싱은 블록5가 RAP에서 시작하지 않음을 나타내고, 따라서 대신에, 세그먼트 인덱싱에 기초하여, 블록4의 시작 시간이 기껏해야 22초이고, 즉 블록4의 시작 시간 오프셋이 21.623초이고 블록4가 RAP에서 시작하기 때문에, 클라이언트는 블록4를 다운로드하도록 선택한다. 따라서, 세그먼트 인덱싱에 기초하여, 클라이언트는 바이트 오프셋 157,034에서 시작하는 HTTP 범위 요청을 할 것이다.

[0188] 만일 세그먼트 인덱싱이 이용가능하지 않으면, 클라이언트는 이러한 데이터를 다운로드하기 전에 모든 앞선 157,034 바이트의 데이터를 다운로드해야 할 수도 있으며, 이는 훨씬 긴 시작 시간, 또는 채널 잽핑 시간을 초래하고 그리고 이용가능하지 않은 데이터의 낭비적인 다운로딩을 초래한다. 대안적으로, 세그먼트 인덱싱이 이용가능하지 않으면, 클라이언트는 원하는 데이터가 세그먼트 내에서 시작하는 위치를 근사화할 수 있지만, 근사화는 조악할 수 있으며 대략적인 시간을 누락할 수 있고 따라서 다시 시동(start up) 지연을 증가시키는 퇴보를 요구한다.

[0189] 일반적으로, 각각의 블록은, 앞선 블록들과 함께 미디어 플레이어에 의해 플레이아웃될 수 있는 미디어 데이터의 부분을 포함한다. 따라서, 세그먼트 내에 포함되거나 또는 다른 수단을 통해 클라이언트에 제공되는, 세그먼트 인덱싱 블록 구조의 클라이언트로의 시그널링 및 블록 구조는 네트워크 변화 및 방해에도 불구하고 끊김 없는 플레이아웃 및 신속한 채널 잽핑을 제공하기 위해 클라이언트의 능력을 현저하게 개선할 수 있다. 가변 지속 기간 블록들의 지원 및 세그먼트 인덱싱에 의해 인에이블되는 GoP의 단지 일부들을 포함하는 블록들은 스트리밍 경험을 현저하게 개선할 수 있다. 예를 들어, 클라이언트가 대략 22초에서 프리젠테이션의 플레이아웃을 시작하기를 원하는 전술한 예 및 도 6을 다시 참조하면, 하나 이상의 요청들을 통해 클라이언트는 블록4 내의 데이터를 요청하고, 그리고 나서 데이터가 플레이백을 시작하도록 이용가능하자마자 이를 미디어 플레이어에 제공할 수 있다. 따라서, 이러한 예에서, 블록4의 42,011 바이트가 클라이언트에서 수신되자마자 플레이아웃이 시작되고, 따라서 신속한 채널 잽핑 시간을 가능케한다. 대신에 플레이아웃이 개시되기 전에 클라이언트가 전체 GoP를 요청할 필요가 있다면, 이는 144,211 바이트의 데이터이기 때문에 채널 잽핑 시간은 더 길어 질 것이다.

[0190] 다른 실시예들에서, RAP들 또는 탐색 포인트들은 또한 블록의 중간에서 발생할 수 있으며, RAP 또는 탐색 포인트가 블록 또는 프래그먼트 내의 어디에 있는지를 나타내는 세그먼트 인덱싱에 데이터가 존재할 수 있다. 다른 실시예들에서, 시간 오프셋은, 블록 내의 제 1 프레임의 프리젠테이션 시간 대신에, 블록 내의 제 1 프레임의 디코딩 시간일 수 있다.

[0191] 도 8의 (a) 및 (b)는 다수의 버전들 또는 리프리젠테이션에 걸쳐 정렬된 탐색 포인트 구조를 크기설정(sizing)하는 가변 블록의 예를 도시한다; 도 8의 (a)는 미디어 스트림의 다수의 버전들에 대한 정렬된 탐색 포인트들을 이용한 가변 블록 크기설정을 도시하며, 도8의 (b)는 미디어 스트림의 다수의 버전들에 대한 비정렬된 탐색 포인트들을 이용한 가변 블록 크기설정을 도시한다.

[0192] 시간은 초 단위로 상부에 가로질러 표시되고, 두 리프리젠테이션을 위한 두 개의 세그먼트의 블록들 및 탐색 포인트들이 이 시간 라인에 대한 이들의 타이밍 관점에서 좌에서 우로 표시되며, 따라서 도시된 각각의 블록의 길이는 블록의 플레이아웃 시간에 비례하며 블록의 바이트들의 수에 비례하지 않는다. 이러한 예에서, 두 리프리젠테이션들의 두 세그먼트들에 대한 세그먼트 인덱싱은 탐색 포인트들에 대해 동일한 시간이지만, 잠재적으로 탐색 포인트들 사이의 프래그먼트들 또는 블록들의 상이한 수들을 가질 것이며 각 블록의 미디어 데이터의 상이한 양으로 인해 블록들에 대해 상이한 바이트 오프셋들을 가질 것이다. 이러한 예에서, 클라이언트가 대략 23초인 프리젠테이션 시간에서 리프리젠테이션 1로부터 리프리젠테이션 2로 스위칭하기를 원하면, 클라이언트는 리프리젠테이션 1에 대한 세그먼트에서 블록 1.2를 통해 요청할 수 있고, 블록 2.2에서 시작하는 리프리젠테이션 2에 대해 세그먼트의 요청을 시작할 수 있으며, 따라서 스위칭은 리프리젠테이션 2의 탐색 포인트 2.2 와 동일한 시간에 있는 리프리젠테이션 1의 탐색 포인트 1.2와 일치하는 프리젠테이션에서 발생할 것이다.

[0193] 전술한 사항으로부터 명백하듯이, 설명된 블록-요청 스트리밍 시스템은 컨텐트 내의 특정 위치들에 탐색 포인트들을 배치시키도록 비디오 인코딩을 강제하지 않으며, 이는 현존 시스템들의 문제들 중 하나를

경감시킨다.

[0194] 전술한 실시예들에서, 동일한 컨텐트의 다양한 리프리젠테이션들에 대한 탐색 포인트들이 정렬되도록 시스템이 구성된다. 그러나, 많은 경우, 이러한 정렬 요건을 완화하는 것이 바람직하다. 예를 들어, 인코딩 툴들이 탐색 포인트 정렬된 리프리젠테이션들을 생성하기 위한 능력들을 갖지 않은 리프리젠테이션들을 생성하기 위해 사용되는 경우가 때때로 있다. 다른 예로서, 컨텐트 프리젠테이션은 상이한 리프리젠테이션들 사이에 어떠한 탐색 포인트 정렬도 없이, 독립적으로 상이한 리프리젠테이션들로 디코딩될 수 있다. 다른 예로서, 리프리젠테이션은 더 낮은 레이트들 갖거나 및 더 통상적으로 스위칭될 필요가 있기 때문에 더 많은 탐색 포인트들을 포함하거나 고속 빨리 감기 또는 되감기 또는 고속 탐색과 같은 트릭 모드들을 지원하기 위해 탐색 포인트들을 포함한다. 따라서, 블록-요청 스트리밍 시스템을 컨텐트 프리젠테이션을 위한 다양한 리프리젠테이션들에 걸쳐 비정렬된 탐색 포인트들을 효율적으로 그리고 끊김 없이 처리하기 할 수 있게 하는 방법들을 제공하는 것이 바람직하다.

[0195] 이러한 실시예에서, 리프리젠테이션들에 걸친 탐색 포인트들의 위치들은 정렬되지 않을 수 있다. 새로운 블록이 각각의 탐색 포인트에서 시작하도록 블록들이 구성되며, 따라서 프리젠테이션의 상이한 버전들의 블록들 사이에 정렬이 존재하지 않을 수 있다. 상이한 리프리젠테이션들 사이의 이러한 비정렬된 탐색 포인트 구조의 예는 도 8(b)에 도시된다. 시간은 초 단위로 상부에 가로질러 도시되며, 두 리프리젠테이션들에 대한 두 세그먼트들의 블록들 및 탐색 포인트들이 이러한 시간 라인에 대해 이들의 타이밍 관점에서 좌에서 우로 도시되며, 따라서, 도시된 각 블록의 길이는 블록의 플레이아웃 시간에 비례하고 블록의 바이트들의 수에 비례하지 않는다. 이러한 예에서, 두 리프리젠테이션들의 두 세그먼트들에 대한 세그먼트 인덱싱은 탐색 포인트들에 대해 잠재적으로 상이한 시간 오프셋들, 및 또한 탐색 포인트들 사이의 프래그먼트들 또는 블록들의 잠재적으로 상이한 수들, 및 각 블록의 미디어 데이터의 상이한 양으로 인해 블록들에 대한 상이한 바이트 오프셋들을 가질 것이다. 이러한 예에서, 클라이언트가 대략 25초인 프리젠테이션 시간에서 리프리젠테이션 1로부터 리프리젠테이션 2로 스위칭하기를 원하면, 클라이언트는 리프리젠테이션 1에 대한 세그먼트에서 블록 1.3을 통해 요청할 수 있고, 블록 2.3에서 시작하는 리프리젠테이션 2에 대해 세그먼트의 요청을 시작할 수 있으며, 따라서 스위칭은 리프리젠테이션 1의 탐색 포인트 1.3의 플레이아웃의 중간에 있는 리프리젠테이션 2의 탐색 포인트 2.3과 일치하는 프리젠테이션에서 발생할 것이다.

[0196] 이러한 실시예에서, 블록 선택기(123)의 동작은 변경될 수 있어서, 앞서 선택된 버전과 상이한 리프리젠테이션으로부터 블록을 선택하는 것이 요구될 때마다, 마지막 선택된 블록의 마지막 프레임에 후속하는 프레임보다 첫 번째 프레임이 더 늦지 않은 가장 최근의 블록이 선택된다.

[0197] 마지막에 설명된 이러한 실시예는 제 1 버전이 아닌 버전들 내에서 탐색 포인트들의 위치들을 제한하기 위한 요건을 제거할 수 있으며, 따라서 이러한 버전들에 대한 압축 효율을 증가시켜서 주어진 이용가능한 대역폭에 대한 더 높은 품질의 프리젠테이션 및 결국 개선된 사용자 경험을 초래한다. 추가의 고려 사항은 컨텐트의 다수의 인코딩들(버전들)에 걸쳐 탐색 포인트 정렬의 기능을 수행하는 비디오 인코딩 툴들이 널리 사용가능하지 않을 수 있으며, 따라서 맨 마지막에 설명된 이러한 실시예의 장점은 현재 이용가능한 비디오 인코딩 툴들이 사용될 수 있다는 것이다. 다른 장점은 컨텐트의 상이한 버전들의 인코딩이 상이한 버전들에 대한 인코딩 프로세스들 사이의 조정에 대한 어떠한 필요성 없이 동시에 진행될 수 있다는 것이다. 다른 장점은, 인코딩 툴들에 특정 탐색 포인트 위치들의 리스트를 제공하지 않고 컨텐트의 추가의 버전들이 인코딩되어 더 늦은 시간에 프리젠테이션에 부가된다는 것이다.

[0198] 일반적으로, 픽쳐들이 픽쳐들의 그룹(GoP)들로 인코딩되는 경우, 시퀀스의 첫 번째 픽쳐는 탐색 포인트일 수 있지만, 언제나 그러할 필요는 없다.

[0199] 선택적 블록 분할

[0200] 블록-요청 스트리밍 시스템에서 하나의 우려 사항은 인코딩된 미디어의 구조, 예를 들어 비디오 미디어와 블록 요청들에 대해 사용된 블록 구조 사이의 상호 작용이다. 비디오 인코딩 기술 분야의 당업자에게 알려져 있듯이, 각각의 비디오 프레임의 인코딩된 리프리젠테이션에 대해 요구되는 비트들의 수가, 때때로 실질적으로, 프레임마다 변화하는 것은 흔히 있는 일이다. 결과로서, 수신된 데이터의 양과 그 데이터에 의해 인코딩된 미디어의 지속 기간 사이의 관계는 간단하지 않을 수 있다. 더욱이, 미디어 데이터를 블록-요청 스트리밍 시스템 내의 블록으로 분할하는 것은 복잡도의 추가의 디멘존을 부가한다. 특히, 일부 시스템에서, 블록의 미디어 데이터는 전체 블록이 수신될 때까지 플레이아웃되지 않을 수 있으며, 예를 들어, 블록 내의 미디어

데이터의 배열 또는 보장(ensure) 코드들의 사용의 블록 내에서 미디어 샘플들 사이의 의존성은 이러한 특성을 초래할 수 있다. 블록 크기와 블록 지속 기간 사이의 이러한 복잡한 상호작용 및 플레이아웃을 시작하기 전에 전체 블록을 수신하기 위한 가능한 필요성의 결과로서, 클라이언트 시스템들이 미디어 데이터가 플레이아웃이 시작하기 전에 버퍼링되는 보수적인 방식을 채택하는 것은 일반적이다. 이러한 버퍼링은 긴 채널 잽핑 시간 및 결국 열악한 사용자 경험을 초래한다.

[0201] 파크자드(Pakzad)는 데이터 스트림의 하부 구조에 기초하여 데이터 스트림을 인접한 블록들로 어떻게 분할할지를 결정하는 새롭고 효율적인 방법들인, "block partitioning methods"를 기술했으며, 스트리밍 시스템의 상황에서 이러한 방법들의 몇몇 장점들을 추가로 기술하였다. 파크자드의 블록 분할 방법들을 블록-요청 스트리밍 시스템에 적용하기 위한 본 발명의 추가의 실시예가 이제 설명된다. 이러한 방법은 대략 프리젠테이션 시간 순서로 제공될 미디어 데이터를 배열하여, 미디어 데이터(예를 들어, 비디오 프레임 또는 오디오 샘플)의 임의의 주어진 엘리먼트의 플레이아웃 시간이 제공된 임계치 미만만큼 임의의 인접함 미디어 데이터의 플레이아웃 시간과 상이하다. 이렇게 정렬된(ordered) 미디어 데이터는 파크자드의 리프리젠테이션을 빌리면 데이터 스트림으로 간주될 수 있으며 이러한 데이터 스트림에 적용되는 파크자드의 임의의 방법들은 블록 경계들을 데이터 스트림과 동일시한다. 임의의 쌍의 인접한 블록 경계들 사이의 데이터는 본 명세서의 리프리젠테이션에서 "블록"으로 간주되며, 본 명세서의 방법들은 블록-요청 스트림 시스템 내의 미디어 데이터의 프리젠테이션을 제공하기 위해 적용된다. 본 명세서를 판독시 기술 분야의 당업자에게 명백하듯이, 파크자드에서 설명된 방법들의 몇몇 장점들은 이후에 블록-요청 스트리밍 시스템의 상황에서 실현될 수 있다.

[0202] 파크자드에서 설명된 바와 같이, 부분적인 GoP들 또는 GoP상에서 보다 더 많은 부분들을 포함하는 블록들을 포함하는 세그먼트의 블록 구조의 결정은 신속한 채널 잽핑 시간들을 가능하게 하도록 클라이언트의 능력에 영향을 줄 수 있다. 파크자드에서, 목표 시동 시간(target startup time)이 주어지면, 목표 시동 시간이 경과된 이후 클라이언트가 임의의 탐색 포인트에서 리프리젠테이션의 다운로딩을 시작하고 플레이아웃을 시작하면, 시간상 각 포인트에서 클라이언트가 다운로드한 데이터의 양이 적어도 목표 다운로드 레이트 × 다운로드의 시작으로부터 경과한 시간인 한, 플레이아웃이 끊김 없이 계속되는 것을 보장하는 목표 다운로드 레이트 및 블록 구조를 제공하는 방법이 제공되었다. 클라이언트가 목표 시동 시간 및 타겟 다운로드 레이트에 액세스하는 것은 유리한데, 이는 이러한 것이 가장 이른 시점에서 리프리젠테이션의 플레이아웃을 시작할 때를 결정하는 수단을 클라이언트에게 제공하고, 다운로드가 전술한 조건을 충족시키는 한 클라이언트가 리프리젠테이션을 계속 플레이아웃하게 허용하기 때문이다. 따라서, 이후에 설명되는 방법은 "미디어 프리젠테이션 디스크립션" 내에서 목표 시동 시간 및 목표 다운로드 레이트를 포함하기 위한 수단을 제공하여, 전술한 목적을 위해 사용될 수 있다.

[0203] 미디어 프리젠테이션 데이터 모델

[0204] 도 5는 도 1에 도시된 컨텐트 저장의 가능한 구조를 도시하며, 세그먼트들 및 미디어 프리젠테이션 기술("MPD") 파일들, 및 세그먼트들의 브레이크다운, 타이밍 및 MPD 파일 내의 다른 구조를 포함한다. MPD 구조들 또는 파일들의 가능한 구현들의 세부사항들이 이제 설명될 것이다. 많은 예들에서, MPD는 파일로서 설명되지만, 비-파일 구조들이 또한 사용될 수 있다.

[0205] 도 5에 도시된 바와 같이, 컨텐트 저장(110)은 다수의 소스 세그먼트들(510), MPD들(510) 및 보수 세그먼트들(512)을 포함한다. MPD는 기간 기록들(501)을 포함할 수 있고, 이는 차례로 초기화 세그먼트들(504) 및 미디어 세그먼트들(505)에 대한 참조들과 같이 세그먼트 정보(503)를 포함하는 리프리젠테이션 기록들(502)을 포함할 수 있다.

[0206] 도 9(a)는 예시적인 메타데이터 테이블(900)을 도시하는 한편, 도 9의 (b)는 HTTP 스트리밍 서버(906)에 대한 접속을 통해 미디어 블록들(904) 및 메타데이터 테이블(900)을 HTTP 스트리밍 클라이언트(902)가 어떻게 획득하는 지의 예를 도시한다.

[0207] 본 명세서에 설명된 방법들에서, 클라이언트에 이용가능한 미디어 프리젠테이션의 리프리젠테이션에 관한 정보를 포함하는 "미디어 프리젠테이션 디스크립션"이 제공된다. 리프리젠테이션들은 클라이언트가 상이한 대안들 중 하나를 선택한다는 점에서 대안적일 수 있거나, 이들은 클라이언트가 몇몇 리프리젠테이션들을 선택하고, 각각이 대안들의 세트로부터 또한 가능하며 이들을 연합하여 제공한다는 점에서 상보적일 수 있다. 리프리젠테이션들은 그룹들에 유리하게 할당될 수 있으며, 클라이언트는 하나의 그룹에서의 리프리젠테이션들에 대해 이들이 서로에 대해 각각의 대안들임을 이해하도록 프로그래밍 또는 구성되는 반편, 상이한 그룹들로부

터의 리프리젠테이션은 하나보다 많은 리프리젠테이션이 연합하여 제공되는 그러한 것이다. 다시 말해서, 클라이언트는 프리젠테이션을 형성하기 위해 그룹에 하나보다 많은 리프리젠테이션이 존재하면, 그 그룹으로부터 하나의 리프리젠테이션을 선택하고, 다음 그룹에서 하나의 리프리젠테이션을 선택하는 식이다.

[0208] 리프리젠테이션들을 설명하는 정보는, 리프리젠테이션, 비디오 프레임 레이트, 비디오 해상도 및 데이터 레이트를 디코딩하기 위해 요구되는 적용되는 미디어 코덱들의 레벨들 및 프로파일들을 포함하는 이러한 코덱들의 세부사항들을 유리하게 포함할 수 있다. "미디어 프리젠테이션 디스크립션"을 수신하는 클라이언트는 리프리젠테이션이 디코딩 또는 프리젠테이션에 대해 적절한 지를 앞서 결정하기 위해 이러한 정보를 사용할 수 있다. 이는 장점을 나타내는데, 그 이유는 리프리젠테이션의 이진 데이터에 차별 정보(differentiating information)가 유일하게 포함되면, 그 적합함에 대한 정보를 찾아내기 위해 모든 리프리젠테이션들로부터 이진 데이터를 요청하고 관련된 정보를 분석 및 추출하는 것이 필요할 것이다. 데이터의 이러한 다수의 요청들 및 분석 부가 추출(parsing annex extraction)은 다소의 시간이 걸리고 이는 긴 시동 시간 및 결국 열악한 사용자 경험을 초래할 것이다.

[0209] 부가적으로, "미디어 프리젠테이션 디스크립션"은 하루 중의 시간에 기반하여 클라이언트 요청들을 제한하는 정보를 포함할 수 있다. 예를 들어, 라이브 서비스의 경우, 클라이언트는 프리젠테이션의 요청 부분들에 대해 제한될 수 있으며, 이는 "현재 브로드캐스트 시간"에 가깝다. 이는 장점을 나타내는데 그 이유는 라이브 브로드캐스트의 경우 현재 브로드캐스트 시간 이전에 제공되는 임계치보다 많이 브로드캐스트되는 컨텐트에 대해 서빙 인프라구조로부터 데이터를 제거하는 것이 바람직할 수 있기 때문이다. 이는 서빙 인프라구조 내에서 저장 리소스들의 재사용을 위해 바람직할 수 있다. 이는 또한 제공된 서비스 타입에 의존하여 또한 바람직할 수 있으며, 예를 들어, 일부 경우들에서, 수신 클라이언트 디바이스들의 어떤 가입(subscription) 모델로 인해 프리젠테이션은 단지 라이브만 이용가능할 수 있는 반면, 다른 미디어 프리젠테이션들이 라이브 및 주문형(on-demand)에 이용가능하게 될 수 있으며, 다른 프리젠테이션들은 클라이언트 디바이스들의 1등급(first class)에 대해서 단지 라이브만이 이용가능하고, 클라이언트 디바이스의 2등급에 대해서는 주문형만 가능하고, 클라이언트 디바이스들의 3등급에 대해서는 라이브 또는 주문형의 조합이 가능하게 될 수 있다. (이하의) "미디어 프리젠테이션 디스크립션"에서 설명되는 방법들은, 서빙 인프라구조에서 이용가능하지 않을 수 있는 데이터에 대해 클라이언트가 사용자에 대한 제공들을 조정하고 요청들 회피할 수 있도록 클라이언트에 이러한 정책들이 통보되게 한다. 대안으로서, 예를 들어, 클라이언트는 이러한 데이터가 이용가능하지 않다는 통보를 사용자에게 제공할 수 있다.

[0210] 본 발명의 추가의 실시예에서, 미디어 세그먼트들은 ISO/IEC 14496-12 또는 파생 상세(specificatoin)들(이를테면, 3GPP 기술 상세 26.244에 설명된 3GP 파일 포맷)에서 설명된 ISO 기반 미디어 파일 포맷을 준수할 수 있다. (전술한) 3GPP 파일 포맷 섹션의 사용은 블록-요청 스트리맹 시스템 내에서 이러한 파일 포맷의 데이터 구조의 효율적인 사용을 허용하는 ISO 기반 미디어 파일 포맷에 대한 새로운 개선을 설명한다. 이러한 참조에서 설명되듯이, 정보가 파일 내에 제공될 수 있어서 파일 내의 바이트 범위들과 미디어 프리젠테이션의 세그먼트들 사이의 신속하고 효율적인 맵핑을 허용한다. 미디어 데이터 그 자체는 ISO/IEC14496-12에 정의된 영화 프래그먼트 구성에 따라 구조화될 수 있다. 시간 및 바이트 오프셋들을 제공하는 이러한 정보는 계층적으로 또는 정보의 단일 블록으로서 구조화될 수 있다. 이러한 정보는 파일의 시작에서 제공될 수 있다. 3GPP 파일 포맷 섹션의 사용에서 설명된 바와 같이 효율적인 인코딩을 사용하는 이러한 정보의 제공은, 블록 요청 스트리밍 시스템에 의해 사용된 파일 다운로드 프로토콜이 HTTP인 경우, 예를 들어, HTTP 부분 GET 부분 요청들을 이용하여 이러한 정보를 클라이언트가 신속하게 리트리브(retrieve)할 수 있게 하며, 이는 짧은 시동, 탐색 또는 스트리밍 스위치 시간을 초래하고 따라서 개선된 사용자 경험을 초래한다.

[0211] 미디어 프리젠테이션의 리프리젠테이션들은, 전형적으로 대안들인, 리프리젠테이션들에 걸쳐 끊김 없는 스위칭을 보장하고, 그리고 두 개 또는 그 보다 많은 리프리젠테이션들의 동기 프리젠테이션을 보장하도록 글로벌 시간 라인에서 동기화된다. 따라서, 적응형 HTTP 스트리밍 미디어 프리젠테이션 내에서 리프리젠테이션들의 함유된 미디어의 샘플 타이밍은 다수의 세그먼트들에 걸친 연속한 글로벌 시간 라인에 관련될 수 있다.

[0212] 다수의 타입들의 미디어를 포함하는 인코딩된 미디어의 블록, 예를 들어 오디오 및 비디오는 상이한 타입들의 미디어에 대해 상이한 프리젠테이션 종료 시간들을 가질 수 있다. 블록 요청 스트리밍 시스템에서, 각각의 미디어 블록들은, 각각의 미디어 타입이 연속적으로 플레이되고 따라서 하나의 블록으로부터의 하나의 타입의 미디어 샘플들이 다른 타입의 선행 블록의 미디어 샘플들 이전에 플레이 아웃될 수 있는 방식으로 연속적으로 플레이아웃될 수 있으며, 이는 본 명세서에서 "연속한 블록 스플라이싱"으로 지칭된다. 대안으로서, 이러한 미디어 블록들은 임의의 타입의 하나의 블록의 가장 이른 샘플이 임의의 타입의 선행 블록의 가장 늦은 샘플

이후에 플레이되는 식으로 플레이아웃될 수 있으며, 이는 본 명세서에서 "불연속 블록 스플라이싱"으로 지칭된다. 두 블록들이 순서대로 또는 다른 경우들로 인코딩된 연속한 컨텐트 아이템 및 동일한 리프리젠테이션으로부터 미디어를 포함할 때 연속한 블록 스플라이싱이 적절할 수 있다. 전형적으로, 하나의 리프리젠테이션 내에서, 연속한 블록 스플라이싱이 두 블록들을 스플라이싱할 때 적용될 수 있다. 이는 유리한데, 그 이유는 현존 인코딩이 적용될 수 있고 세그멘테이션이 블록 경계들에서 미디어 트랙들을 정렬할 필요 없이 수행될 수 있기 때문이다. 이는 도 10에 도시되면, 여기서, 비디오 스트림(1000)은 RAP(1204)와 같은 RAP들을 갖는 블록(1202) 및 다른 블록들을 포함한다.

[0213] 미디어 프리젠테이션 디스크립션

[0214] 미디어 프리젠테이션은 HTTP-스트리밍 서버 상의 파일들의 구조화된 컬렉션(collection)으로서 여겨질 수 있다. HTTP-스트리밍 클라이언트는 스트리밍 서비스를 사용자에게 제공하기 위해 충분한 정보를 다운로드할 수 있다. 대안적인 리프리젠테이션들은 3GP 파일로부터 또는 3GP 파일로 쉽게 변환될 수 있는 적어도 잘 정의된 데이터 구조들의 세트에 또는 3GPP 파일 포맷에 따르는 3GP 파일의 일부들 또는 하나 또는 그보다 많은 3GP 파일들로 구성될 수 있다.

[0215] 미디어 프리젠테이션은 미디어 프리젠테이션 디스크립션에 의해 설명될 수 있다. 미디어 프리젠테이션 디스크립션(MPD)은, 적절한 시간에 데이터에 액세스하고 스트리밍 서비스를 사용자에게 제공하기 위해 클라이언트가 적절한 파일 요청들, 예를 들어, HTTP GET 요청들을 구성하도록 사용할 수 있는 메타데이터를 포함할 수 있다. 미디어 프리젠테이션 디스크립션은 적절한 3GPP 파일들 및 파일들의 부분(piece)들을 선택하도록 HTTP 스트리밍 클라이언트에 충분한 정보를 제공할 수 있다. 클라이언트에게 액세스 가능하다고 시그널링되는 유닛들은 세그먼트들로 지칭된다.

[0216] 특히, 미디어 프리젠테이션 디스크립션은 다음과 같이 엘리먼트들 및 속성들을 포함할 수 있다.

[0217] 미디어 프리젠테이션 디스크립션 에리먼트(MediaPresentationDescription Element)

[0218] 엘리먼트 캡슐화 메카데이터는 스트리밍 서비스를 최종 사용자에게 제공하기 위해 HTTP 스트리밍 클라이언트에 의해 사용된다. MediaPresentationDescription Element는 이하의 속성들 및 엘리먼트들 중 하나 이상을 포함할 수 있다.

[0219] Version: 확장 가능을 보장하기 위한 프로토콜에 대한 버전 번호.

[0220] PresentationIdentifier: 프리젠테이션이 다른 프리젠테이션들 사이에서 고유하게 식별되게 하는 정보. 개별적인 필드들 또는 네임들을 또한 포함할 수 있음.

[0221] UpdateFrequency: 미디어 프리젠테이션 디스크립션의 업데이트 빈도, 즉 얼마나 자주 클라이언트가 실제 미디어 프리젠테이션 디스크립션을 리로드할 수 있는 지를 나타냄. 존재하지 않으면, 미디어 프리젠테이션은 고정적일 수 있음. 미디어 프리젠테이션의 업데이트는 미디어 프리젠테이션이 캐싱될 수 없음을 의미할 수 있음.

[0222] MediaPresentationDescriptionURI: 미디어 프리젠테이션 디스크립션을 업데이팅하기 위한 URI.

[0223] 스트리밍: 스트림 또는 미디어 프리젠테이션의 타입을 설명: 비디오, 오디오 또는 텍스트. 비디오 스트림 타입은 오디오를 포함할 수 있고 텍스트를 포함할 수 있음.

[0224] Service: 추가의 속성들을 갖는 서비스 타입을 설명. 서비스 타입들은 라이브 및 주문형일 수 있음. 이는 어느 정도의 현재 시간을 초월하여 탐색 및 액세스가 허용되지 않음을 클라이언트에게 통보하기 위해 사용될 수 있음.

[0225] MaximumClientPreBufferTime: 클라이언트가 미디어 스트림을 프리버퍼링할 수 있는 최대량의 시간. 클라이언트가 이러한 최대 프리버퍼 시간을 초과하여 다운로드하는 것이 제한되는 경우, 이러한 타이밍은 점진적인 다운로드로부터의 스트리밍을 식별할 수 있다. 프리버퍼링의 관점에서 어떠한 제한들도 적용될 수 없음을 나타내는 값이 제공되지 않을 수 있다.

[0226] SafetyGuardIntervalLiveService: 서버에서 라이브 서비스의 최대 턴어라운드(turn-around) 시간에 대한 정보. 현재 시간에서 이미 액세스 가능한 정보가 무엇인지의 표시를 클라이언트에게 제공. 이 정보는, 클라

이언트 및 서버가 UTC 시간 상에서 동작하도록 예상되고 어떠한 긴밀한 시간 동기화도 제공되지 않은 경우 필수적일 수 있다.

[0227] TimeShiftBufferDepth: 클라이언트가 현재 시간에 대해 라이브 서비스에 어디까지 이동할 수 있는 지에 대한 정보. 이러한 깊이의 확장에 의해, 시간 시프트 뷰잉(viewing) 및 캐치업(catch-up) 서비스는 서비스 제공에서 특정 변경들 없이 허용될 수 있다.

[0228] LocalCachingPermitted: 이러한 플래그는 HTTP 클라이언트가 다운로드된 데이터가 플레이된 후 이를 국부적으로 캐싱할 수 있는 지를 나타냄.

[0229] LivePresentationInterval: 시간 간격들을 포함하며, 이 시간 간격들 동안 프리젠테이션은 시작 시간들 및 종료 시간들을 특정함으로써 이용가능할 수 있다. 시작 시간은 서비스들의 시작 시간을 나타내며, 종료 시간은 서비스의 종료 시간을 나타낸다. 종료 시간이 특정되지 않으면, 종료 시간은 현재 시간에서 알려지지 않으며, 업데이트 빈도는 서비스의 실제 종료 시간 이전에 종료 시간에 대한 액세스를 클라이언트들이 획득하는 것을 보장할 수 있다.

[0230] OnDemandAvailabilityInterval: 프리젠테이션 간격은 네트워크 상에서의 서비스의 이용가능성을 나타낸다. 다수의 프리젠테이션 간격들이 제공될 수 있다. HTTP 클라이언트는 임의의 특정 시간 윈도우 밖의 서비스를 액세스하지 못할 수 있다. 주문형 간격의 제공에 의해, 추가의 시간 시프트 뷰잉이 특정될 수 있다. 이러한 속성은 또한 라이브 서비스에 대해 존재할 수 있다. 속성이 라이브 서비스에 대해 존재하는 경우, 서버는 모든 제공되는 이용가능성 간격들 동안 클라이언트가 주문형 서비스로서 서비스에 액세스할 수 있음을 보장할 수 있다. 따라서, 라이브 프리젠테이션 간격은 임의의 주문형 이용가능성 간격과 중첩하지 않을 수 있다.

[0231] MPDFileInfoDynamic: 이는 미디어 프리젠테이션에서 파일들의 디폴트 동적 구성을 설명한다. 더욱 상세한 사항들은 이하에서 제공된다. 몇몇 또는 모든 대안적인 리프리젠테이션에 대해 동일한 규칙들이 사용되면 MPD 레벨에 대한 디폴트 상세는 불필요한 반복을 면하게 할 수 있다.

[0232] MPDCodecDescription: 이는 미디어 프리젠테이션에서 메인 디폴트 코덱들을 설명한다. 더욱 상세한 사항들은 이하에서 설명된다. MPD 레벨에 대한 디폴트 상세는, 몇몇 또는 모든 리프리젠테이션들에 대해 동일한 코덱들이 사용되는 경우 불필요한 반복을 면하게 할 수 있다.

[0233] MPDMoveBoxHeaderSizeDoesNotChange: 이는 전체 미디어 프리젠테이션 내의 개별 파일들 중에서 MoveBox Header가 크기에서 변화하는 지를 나타내기 위한 플래그이다. 이러한 플래그는 다운로드를 최적화하기 위해 사용될 수 있으며, 특정 세그먼트 포맷들의 경우, 특히 세그먼트들이 moov 헤더를 포함하는 경우 유일하게 존재할 수 있다.

[0234] FileURIPattern: 이는, 미디어 프리젠테이션 내의 파일들에 대해 요청 메시지들을 생성하기 위해 클라이언트에 의해 사용되는 패턴이다. 상이한 속성들이 미디어 프리젠테이션 내의 파일들 각각에 대해 유일한 URI들의 생성을 허용한다. 베이스 URI는 HTTP URI일 수 있다.

[0235] Alternative Representation: 이는 리프리젠테이션들의 리스트를 설명한다.

[0236] 대안적인 리프리젠테이션 엘리먼트(AlternativeRepresentation Element)

[0237] 하나의 리프리젠테이션에 대한 모든 메타데이터를 캡슐화하는 XML 엘리먼트. AlternativeRepresentation Element는 이하의 속성들 및 엘리먼트들을 포함할 수 있다.

[0238] RepresentationID: 미디어 프리젠테이션 내에서 이러한 특정한 Alternative Representation에 대한 고유한 ID.

[0239] FilesInfo Static: 하나의 대안적인 프리젠테이션의 모든 파일들의 URI 및 시작 시간들의 명시적인 리스트를 제공. 파일들의 리스트의 고정적인 제공은 미디어 디스크립션의 정확한 타이밍 디스크립션의 장점을 제공할 수 있지만, 특히 대안적인 리프리젠테이션이 많은 파일들을 포함하는 경우, 그렇게 컴팩트하지 않을 수 있다. 또한, 파일 네임들은 임의의 네임들을 가질 수 있다.

[0240] FilesInfoDynamic: 하나의 대안적인 프리젠테이션의 URI 및 시작 시간들의 리스트를 구성하기 위한 암시적 방식을 제공한다. 파일들의 리스트의 동적 제공은 더 많은 컴팩트 리프리젠테이션의 장점을 제공할 수 있다. 시작 시간들의 시퀀스만이 제공되면, 타이밍 장점들이 또한 여기서 유지되지만, 파일 네임들은

FilePatternURI에 동적으로 기반을 두고 구성될 것이다. 각각의 세그먼트의 지속 기간이 제공되기만 하면, 리프리젠테이션은 컴팩트하며, 라이브 서비스들 내에서 사용하기에 적절할 수 있지만, 파일들의 생성은 글로벌 타이밍에 의해 좌우될 수 있다.

[0241] APMoveBoxHeaderSizeDoesNotChange: Alternative Description 내에서 개별 파일들 중에서 MoveBox Header가 크기에서 변화하는 경우를 나타내는 플래그. 이러한 플래그는 다운로드를 최적화하기 위해 사용될 수 있고, 특정 세그먼트 포맷들의 경우, 특히 세그먼트들이 moov 헤더를 포함하는 경우 유일하게 존재할 수 있다.

[0242] APCodecDescription: 대안적인 프리젠테이션에서 파일들의 메인 코덱들을 설명.

[0243] 미디어 디스크립션 엘리먼트(Media Description Element)

[0244] MediaDescription: 이러한 리프리젠테이션에 포함된 미디어에 대한 모든 메타데이터를 캡슐화할 수 있는 엘리먼트. 특히, 이는, 적용가능한 경우, 추천된 트랙들의 그룹은 물론 이러한 대안적인 프리젠테이션의 트랙들에 대한 정보를 포함할 수 있다. MediaDescription Attribute은 이하의 속성들을 포함한다:

[0245] TrackDescription: 이러한 리프리젠테이션에 포함된 미디어에 대한 모든 메타데이터를 캡슐화하는 XML 속성. TrackDescription Attribute는 이하의 속성들을 포함한다:

[0246] TrackID: 대안적인 리프리젠테이션 내의 트랙에 대한 고유한 ID. 이는 트랙이 그룹 디스크립션의 일부인 경우 사용될 수 있다.

[0247] Bitrate : 트랙의 비트레이트.

[0248] TrackCodecDescription: 이러한 트랙에 사용된 코덱에 대한 모든 속성들을 포함하는 XML 속성. TrackCodecDescription Attribute는 이하의 속성들을 포함한다:

[0249] MediaName: 미디어 타입을 정의하는 속성. 미디어 타입들은 "오디오", "비디오", "텍스트", "애플리케이션" 및 "메시지"를 포함한다.

[0250] Codec: CodecType은 프로파일 및 레벨을 포함한다.

[0251] LanguageTag: 적용가능한 경우, LanguageTag.

[0252] Max Width, MaxHeight: 비디오의 경우, 픽셀에서 포함된 비디오의 높이 및 폭.

[0253] SamplingRate: 오디오의 경우, 샘플링 레이트.

[0254] GroupDescription: 상이한 파라미터들에 기반하여 적절한 그룹핑을 위해 클라이언트에 추천을 제공하는 속성.

[0255] GroupType: 클라이언트가 트랙들을 어떻게 그룹핑할지를 결정하는데 기반하는 타입.

[0256] 미디어 프리젠테이션 디스크립션에서의 정보는, 예를 들어, 액세스 대역폭, 디스플레이 능력들, 코덱 능력들 등은 물론 언어 등과 같은 사용자의 선호들의 관점에서 그 능력들을 매칭하는 적절한 리프리젠테이션으로부터 세그먼트들을 선택하는, 적절한 시간들에서 파일들/세그먼트들 또는 그들의 부분들에 대한 요청들을 수행하기 위해 HTTP 스트리밍 클라이언트에 의해 유리하게 사용된다. 더욱이, 미디어 프리젠테이션 디스크립션은 글로벌 시간라인에 시간 정렬되고 맵핑되는 리프리젠테이션을 기술하기 때문에, 클라이언트는 연합하여 리프리젠테이션을 제공하거나 미디어 프리젠테이션 내에서 탐색하기 위해, 리프리젠테이션들에 걸쳐 스위칭하기 위한 적절한 동작들을 개시하기 위한 진행중인 미디어 프리젠테이션 동안 MPD에서 정보를 사용할 수 있다.

[0257] 시그널링 세그먼트 시작 시간들(Signalling Segment Start Times)

[0258] 리프리젠테이션은 시간에 대해 다수의 세그먼트들로 분할될 수 있다. 인터-트랙 타이밍 문제가 하나의 세그먼트의 마지막 프래그먼트와 다음 세그먼트의 다음 프래그먼트 사이에 존재한다. 또한, 다른 타이밍 문제가 일정한 지속시간의 세그먼트들이 사용되는 경우에 존재한다.

[0259] 각각의 세그먼트에 대해 동일한 지속 기간을 사용하는 것은 MPD가 컴팩트하고 고정적이라는 장점을 가질 수 있다. 그러나 각각의 세그먼트는 여전히 랜덤 액세스 포인트에서 시작할 수 있다. 따라서, 비디오 인코딩이 이러한 특정 포인트들에서 랜덤 액세스 포인트들을 제공하도록 제한될 수 있거나, 실제 세그먼트 지속시간들

이 MPD에서 특정되는 것처럼 정밀하지 않을 수 있다. 스트리밍 시스템이 비디오 인코딩 프로세스를 불필요하게 제한하지 않는 것이 바람직할 수 있으며, 따라서 두 번째 옵션이 바람직할 수 있다.

[0260] 구체적으로, 파일 지속시간이 d초로 MPD에서 특정되면, n 번째 파일은 시간 (n-1)d 에서 또는 그 직후 랜덤 액세스 포인트부터 시작할 수 있다.

[0261] 이러한 해결책(approach)에서, 각각의 파일은 글로벌 프리젠테이션 시간의 면에서는 세그먼트의 정확한 시작(start) 시간에 관한 정보를 포함할 수 있다. 이를 시그널링하기 위한 3개의 가능한 방식들은 하기의 것들을 포함한다.

[0262] (1) 첫째, MPD에 지정된(specified) 것처럼 정확한 타이밍에 대해 각각의 세그먼트의 시작 시간을 제한한다. 그러나, 이후 미디어 인코더(media encoder)는 IDR 프레임들의 배치(placement)에 대한 임의의 탄력성(flexibility) 갖지 않을 수 있고 파일 스트리밍을 위한 특정한 인코딩을 요구할 수 있다.

[0263] (2) 둘째, 각각의 세그먼트에 대한 MPD에 정확한 시작 시간을 부가한다. 온-디멘드(on-demand) 경우에 대해, MPD의 압축성(compactness)은 감소될 수 있다. 라이브(live) 경우에 대해, 이는 MPD의 정규(regular) 업데이트를 요구할 수 있으며 이는 확장성(scalability)을 감소시킬 수 있다.

[0264] (3) 셋째, 세그먼트가 이 정보를 포함한다는 점에서, 글로벌 시간 또는 리프리젠테이션(representation)의 발표된(announced) 시작 시간에 관한 정확한 시작 시간 또는 MPD의 세그먼트의 발표된 시작 시간을 세그먼트에 부가한다. 이는 적응식 스트리밍(adaptive streaming)에 전용된 새로운 박스(new box)에 부가될 수 있다. 이 박스는 또한 "TIDX" 또는 "SIDX" 박스에 의해 제공되는 것과 같은 정보를 포함할 수 있다. 이러한 세번째 해결책의 결과는, 세그먼트들 중 하나의 시작부(beginning) 부근의 특정 위치를 탐색(seeking)할 때, 클라이언트는 MPD에 기초하여, 요구되는 탐색 포인트를 포함하는 것에 대한 차후의 세그먼트를 선택할 수 있다는 것이다. 이 경우 간단한 응답은 리트리브된 세그먼트의 시작으로(즉, 탐색 포인트 이후 다음 랜덤 액세스 포인트로) 탐색 포인트 전방(forward) 이동시키는 것일 수 있다. 통상적으로, 랜덤 액세스 포인트들은 적어도 몇 초마다 제공되며 (그리고 여기에는 종종 이들을 덜 빈번하게 만드는 적은(little) 인코딩 이득이 제공되며) 최악의 경우 지정된 것보다 몇 초 늦게 탐색 포인트가 이동될 수 있다. 대안적으로, 클라이언트는 요청된 탐색 포인트가 사실상 이전 세그먼트에 있다는 세그먼트에 대한 헤더 정보를 리트리빙시에 결정할 수 있고 대신 그 세그먼트를 요청할 수 있다. 이는 탐색 동작(operation)을 실행하기 위해 요구되는 시간에서의 간헐적 증가(occasional increase)를 야기할 수 있다.

[0265] 액세스가능한 세그먼트 리스트

[0266] 미디어 프리젠테이션(media presentation)은, 각각이 오리지널 미디어 컨텐트에 대한 일부 상이한 버전의 인코딩을 제공하는 리프리젠테이션들의 세트를 포함한다. 리프리젠테이션들 자체는, 유리하게 다른 파라미터들에 비해 리프리젠테이션을 구별하는 파라미터들에 대한 정보를 포함한다. 이들은 또한, 액세스가능한 세그먼트들의 리스트를 명시적으로 또는 암시적으로 포함한다.

[0267] 세그먼트들은 메타데이터만을 포함하는 타임-리스(time-less) 세그먼트들 및 주로(primarily) 미디어 데이터를 포함하는 미디어 세그먼트들에서 구별될 수 있다. 미디어 프리젠테이션 디스크립션(MPD)은 유리하게 암시적으로 또는 명시적으로, 세그먼트들 각각을 식별하고 세그먼트들 각각에 서로다른 속성들을 할당한다. 유리하게 각각의 세그먼트에 할당된 속성들은 세그먼트가 액세스가능한 기간, 그를 통해 세그먼트들이 액세스가능한 자원들 및 프로토콜들을 포함한다. 또한, 미디어 세그먼트들에는 유리하게, 미디어 프리젠테이션에서 세그먼트의 시작 시간, 및 미디어 프리젠테이션에서 세그먼트의 지속시간과 같은 속성들이 할당된다.

[0268] 미디어 프리젠테이션이, 유리하게 OnDemandAvailabilityInterval과 같은 미디어 프리젠테이션 디스크립션에서의 속성에 의해 표시되 것처럼 "온-디멘드(on-demand)" 타입이면, 통상적으로 미디어 프리젠테이션 디스크립션은 전체 세그먼트들을 기술하며 또한 세그먼트들이 액세스가능할 때 그리고 세그먼트들이 액세스가능하지 않을 때의 표시를 제공한다. 세그먼트들의 시작 시간들은 유리하게, 동일한 미디어 프리젠테션들(그러나 상이한 시간들에서)의 플레이-백(play-back)을 시작하는 2 클라이언트들이 동일한 미디어 세그먼트들 뿐만 아니라 동일한 미디어 프리젠테이션 디스크립션을 이용할 수 있도록, 미디어 프리젠테이션의 시작에 관해 표현된다. 이는 유리하게 세그먼트들을 캐시(cache)하는 능력을 개선한다.

[0269] 미디어 프리젠테이션이, 유리하게 속성 *Service*와 같은 미디어 프리젠테이션 디스크립션에서의 속성에 의해

표시되는 것처럼 "라이브(live)" 타입이면, 하루 중 실제 시간을 지난(beyond) 미디어 프리젠테이션을 포함하는 세그먼트들은, 세그먼트들이 MPD에 완전히 기술됨에도 불구하고 일반적으로 생성되지 않거나 또는 적어도 액세스가능하지 않다. 그러나, 미디어 프리젠테이션 서비스가 "라이브" 타입이라는 표시를 이용하여, 클라이언트는 MPD의 다운로드 시간 및 MPD에 포함된 정보에 기초하여 월-클록 타임(wall-clock time)에서 클라이언트 내부 시간 NOW에 대한 타이밍 속성들과 함께 액세스가능한 세그먼트들의 리스트를 생성할 수 있다. 서버는 월-클록 타임 NOW에서 MPD의 인스턴스(instance)로 동작하는 레퍼런스 클라이언트가 자원들을 액세스할 수 있도록, 자원을 액세스가능하게 만든다는 점에서 유리하게 동작한다.

[0270] 특정하게, 레퍼런스 클라이언트는 MPD의 다운로드 시간 및 MPD에 포함된 정보에 기초하여 월-클록 타임에서 클라이언트 내부 시간 NOW에 대한 타이밍 속성들과 함께 액세스가능한 세그먼트들의 리스트를 생성한다. 시간이 진행(advancing)됨에 따라, 클라이언트는 동일한 MPD를 이용할 것이며 미디어 프리젠테이션을 연속적으로 플레이아웃(playout)하기 위해 사용될 수 있는 새로운 액세스가능한 세그먼트 리스트를 생성할 것이다. 따라서, 서버는 이러한 세그먼트들이 실제적으로 액세스가능하기 전에 MPD에서 세그먼트들을 발표할 수 있다. 이는, MPD의 빈번한 업데이팅 및 다운로딩을 감소시키기 때문에 유리하다.

[0271] 각각이 시작 시간(tS)을 갖는 세그먼트들의 리스트가 FileInfoStatic 과 같은 엘리먼트들의 플레이 리스트에 의해 명시적으로 또는 FileInfoDynamic 과 같은 엘리먼트를 사용함으로써 암시적으로 기술된다고 가정한다. FileInfoDynamic 를 이용하는 세그먼트 리스트를 생성하는 유리한 방법이 하기에 기술된다. 이러한 구성 룰(construction rule)에 기초하여, 클라이언트는 본 명세서에서 $\mathrm{FileURI}(r,i)$ 로서 지칭되는 각각의 리프리젠테이션(r)에 대한 URL들의 리스트에 대한 액세스, 및 인덱스(i)를 갖는 각각의 세그먼먼트에 대한 시작 시간 $tS(r,i)$ 을 갖는다.

[0272] 세그먼트들의 액세스가능한 시간 윈도우를 생성하기 위한 MPD의 정보 사용은 하기의 룰들을 이용하여 수행될 수 있다.

[0273] "온-디멘드" 타입의 서비스에 대해, *Service*와 같은 속성에 의해 유리하게 표시되는 것처럼, 클라이언트에서 현재의 월-클록 타임 NOW가 OnDemandAvailabilityInterval 과 같은 MPD 엘리먼트에 의해 유리하게 표현되는 가용성(availability)의 임의의 범위 내에 있다면, 이러한 온-디멘드 프리젠테이션의 기술된 모든 세그먼트들은 액세스가능하다. 클라이언트에서 현재의 월-클록 타임 NOW가 가용성의 임의의 범위를 벗어나면, 이러한 온-디멘드 프리젠테이션의 기술된 세그먼트들 어느 것도 액세스될 수 없다.

[0274] "라이브" 타입의 서비스에 대해, *Service*와 같은 속성에 의해 유리하게 표시되는 것처럼, 시작 시간 $tS(r,i)$ 은 월-클록 타임에 가용성의 시간을 유리하게 표현한다. 가용성 시작 시간은 이벤트의 라이브 서비스 시간 및 캡처링(capturing), 인코딩, 및 공개(publishing)를 위한 서버에서의 소정의 턴-어라운드(turn-around) 시간의 조합으로서 유추(derive)될 수 있다. 예를 들어, 이러한 프로세스에 대한 시간은, 예를 들어 MPD에 SafetyGuardIntervalLiveService 로서 지정된 예에 대해 지정된 안전 가드(safety guard) 인터벌 tG를 사용하여 MPD에 지정될 수 있다. 이는 HTTP 스트리밍 서버에 대한 데이터의 가용성과 UTC 시간 간의 최소 차를 제공할 수 있다. 또 다른 실시예에서, MPD는 이벤트 라이브 시간 및 턴-어라운드 시간 간의 차로서 턴-어라운드 시간을 제공하지 않고도 MPD의 세그먼트의 가용성 시간을 명시적으로 지정한다. 하기 설명들에서, 임의의 글로벌 시간들이 가용성 시간들로서 지정된다는 것이 가정된다. 라이브 미디어 브로드캐스팅 기술분야의 업자들은 본 설명을 판독한 후 미디어 프리젠테이션 디스크립션에서의 적절한 정보로부터 이러한 정보를 유추할 수 있다.

[0275] 클라이언트에서 현재의 월-클록 타임 NOW이 LivePresentationInterval 와 같은 MPD 엘리먼트에 의해 유리하게 표현되는 라이브 프리젠테이션 인터벌의 임의의 범위를 벗어나면, 이 라이브 프리젠테이션의 기술된 세그먼트들 어느 것도 액세스될 수 없다. 클라이언트에서 현재의 월-클록 타임 NOW이 라이브 프리젠테이션 인터벌 내에 있다면, 이 라이브 프리젠테이션의 기술된 세그먼트들의 적어도 특정 세그먼트들은 액세스가능할 수 있다.

[0276] 액세스가능한 세그먼트들의 제약(restriction)은 하기의 값들에 의해 제어된다:

[0277] (클라이언트에 대해 이용가능한 것으로서의) 월-클록 타임 NOW.

[0278] 미디어 프리젠테이션 디스크립션에 TimeShiftBufferDepth 로서 지정된 예에 대해 허용된 시간-이동 버퍼

깊이 tTSB.

[0279] 관련 이벤트 시간 t_1에서의 클라이언트는 단지, (NOW - tTSB) 및 NOW의 인터벌에서의 또는 지속시간 d을 갖는 세그먼트의 마지막 시간이 또한 포함되어 $(NOW - tTSB - d)$ 및 NOW의 인터벌을 야기하게 하는 인터벌에서의 시작 시간들 $tS(r, i)$ 을 갖는 세그먼트들을 요청하도록 허용될 수 있다.

[0280] MPD 업데이팅

[0281] 일부 실시예들에서, 서버는, 예를 들어 서버 로케이션이 변경할 때, 또는 미디어 프리젠테이션이 서로 다른 서버로부터의 일부 광고를 포함할 때, 또는 미디어 프리젠테이션의 지속시간이 미지(unknown)일 때, 또는 서버가 이후의(following) 세그먼트들에 대한 로케이터를 혼란시키길(to obfuscate) 원할 때, 세그먼트들의 시작 시간들 및 파일 또는 세그먼트 로케이터를 사전에 인지할 수 없다.

[0282] 이러한 실시예들에서, 서버는 MPD의 이러한 인스턴스가 공개된 이후 짧게 액세스가능한 또는 이미 액세스가능한 세그먼트들만을 기술할 수 있다. 또한, 일부 실시예들에서, 클라이언트는 사용자가 미디어 컨텐트의 생성을 위해 가능한 근접하게 포함된 미디어 프로그램을 경험하도록, MPD에 기술된 미디어에 근접한 미디어를 유리하게 소모한다. 클라이언트가 MPD의 기술된 미디어 세그먼트들의 마지막에 도달한다는 것을 예상하자마자, 클라이언트는 새로운 미디어 세그먼트들을 기술하는 새로운 MPD를 서버가 공개한다는 예상으로 연속적 플레이-아웃을 지속하기 위해 MPD의 새로운 인스턴스를 유리하게 요청한다. 서버는 MPD의 새로운 인스턴스들을 유리하게 생성하며 클라이언트들이 연속적 업데이트들에 대한 프로시저들에 대해 의지할 수 있도록 MPD를 업데이트한다. 서버는, 공통 클라이언트로서의 동작들이 동작할 수 있는 레퍼런스 클라이언트의 프로시저들에 대한 세그먼트 생성 및 공개와 함께 그의 MPD 업데이트 프로시저들을 적응할 수 있다.

[0283] MPD의 새로운 인스턴스가 단지 짧은 시간 진행(short time advance)을 기술하면, 클라이언트들은 MPD의 새로운 인스턴스들을 빈번하게 요청할 것을 필요로 한다. 이는 불필요한 빈번한 요청들로 인해 확장성 문제들 및 불필요한 업링크 및 다운링크 트래픽을 야기할 수 있다.

[0284] 따라서, 한편으로는 세그먼트들을 아직 액세스가능하게 만들 필요없이 가능한 추후에 그 세그먼트들을 기술하고, 다른 한편으로는 MPD 내의 예측하지 못한 업데이트들이 새로운 서버 위치들을 표현하거나, 광고들과 같은 새로운 컨텐트의 삽입을 허용하거나, 또는 코덱 파라미터들에서의 변화들을 제공하는 것을 가능케하는 것이 적절하다.

[0285] 또한, 일부 실시예들에서, 미디어 세그먼트들의 지속시간은 몇 초 범위로 작을 수 있다. 미디어 세그먼트들의 지속시간은 전달 또는 캐싱 특성들에 대해 최적화될 수 있는 적절한 세그먼트 크기들로 조절되고, 세그먼트들의 저장 또는 전달을 다루는 다른 양상들 또는 라이브 서비스들에서의 종대종 지연(end-to-end), 또는 다른 요인들에 대해 보상하도록 유리하게 탄력적이다. 특히 세그먼트들이 미디어 프리젠테이션 지속시간에 비해 작은 경우들이면, 상당한 양의 미디어 세그먼트 자원들 및 시작 시간들이 미디어 프리젠테이션 디스크립션에 기술될 필요가 있다. 결과적으로, 미디어 프리젠테이션 디스크립션의 크기는 클 수 있고 이는 미디어 프리젠테이션 디스크립션의 다운로드 시간에 악영향을 미칠 수 있고 이로 인해 미디어 프리젠테이션의 시동(start-up) 지연 및 또한 액세스 링크에 대한 대역폭 사용에 영향을 미칠 수 있다. 따라서, 플레이리스트들을 이용하여 미디어 세그먼트들의 리스트의 디스크립션을 허용할 뿐 아니라, 템플릿들 또는 URL 구성 룰들을 이용함으로써 디스크립션을 허용하는 것이 유리하다. 템플릿들 및 URL 구성 룰들은 이러한 디스크립션에서 동의어로 사용된다.

[0286] 또한, 템플릿들은 현재 시간을 지난(beyond) 라이브 경우들에서 세그먼트 로케이터들을 기술하는데 유리하게 이용될 수 있다. 이러한 경우들에서, MPD의 업데이트 그 자체는 로케이터들처럼 불필요하며 뿐만 아니라 세그먼트 리스트는 템플릿들에 의해 기술된다. 그러나, 프리젠테이션들 또는 세그먼트들의 디스크립션에서 변화들을 요구하는 예측하지 못한 이벤트들이 여전히 발생할 수 있다. 적응식(adaptive) HTTP 스트리밍 미디어 프리젠테이션 디스크립션에서의 변화들은 다수의 서로 다른 소스들로부터의 컨텐트가 서로 결합될 때, 예를 들면 광고가 삽입되었을 때, 요구될 수 있다. 서로 다른 소스들로부터의 컨텐트는 다양한 방식들로 상이할 수 있다. 또 다른 원인은, 라이브 프리젠테이션들 동안, 하나의 라이브 원서버(origin server)로부터 다른 서버에 페일-오버(fail-over)를 제공하기 위해 컨텐트 파일들에 대해 사용되는 URL들을 변화시킬 필요가 있을 수 있다는 것이다.

[0287] 일부 실시예들에서, MPD가 업데이트되면, MPD에 대한 업데이트들은, 레퍼런스 클라이언트 및 이에 따른 임의의 구현된 클라이언트가 MPD의 이전 인스턴스로부터 행해질 수 있는 것처럼, 이전의 PMD의 유효 시간까지 임의의 시간 동안 업데이트된 MPD로부터 액세스가능한 세그먼트들의 동일한 기능 리스트를 생성한다는 점에따라 업데이트된 MPD가 이전의 PMD와 호환되도록 실행되는 것이 유리하다. 이러한 요구조건은, (a) 클라이언트는, 업데이트 시간 이전에 구(old) MPD와 호환되기 때문에, 구(old) MPD와의 동기화 없이 새로운 MPD를 이용하여 즉시 시작될 수 있고; (b) 업데이트 시간이 MPD에 대한 실제 변화가 발생하는 시간과 동기화될 필요가 없다는 것을 보장한다. 다른 말로, MPD에 대한 업데이트들은 사전에 광고될 수 있으며 서버는, MPD의 서로 다른 버전들을 유지하지 않고도 새로운 정보가 이용될 수 있다면 MPD의 구 인스턴스를 교체할 수 있다.

[0288] 2개의 가능성들은 모든 프리젠테이션들 또는 프리젠테이션들의 세트에 대한 MPD 업데이트에 대한 미디어 타이밍에 대해 존재할 수 있다. (a) 존재하는 글로벌 타임라인이 MPD 업데이트에 대해 지속되거나(본 명세서에서 "연속적 MPD 업데이트"로서 지칭됨), 또는 (b) 현재 타임라인이 종료(end)되고 새로운 타임라인이 변화에 따라 세그먼트로 시작된다(본 명세서에서 "불연속 MPD 업데이트"로 지칭됨).

[0289] 이러한 옵션들 간의 차는 미디어 프래그먼트(Media Fragment) 및 이에 따른 세그먼트의 트랙들이 일반적으로 시작되지 않고 트랙들에 대한 상이한 샘플 입상도(granularity) 때문에 동일한 시간에 종료되는 것이 고려될 때 증명될 수 있다. 정규(normal) 프리젠테이션 동안, 프래그먼트의 하나의 트랙의 샘플들은 이전 프래그먼트의 또 다른 트랙의 일부 샘플들 이전에 렌더링될 수 있다, 즉 싱글 트랙 내에서는 오버랩되지 않을 수 있지만 프래그먼트들 사이에서는 일부 형태(kind)의 오버랩이 있다.

[0290] (a)와 (b) 사이의 차는, 이러한 오버랩이 MPD 업데이트에 대해 인에이블될 수 있는지 여부에 있다. MPD 업데이트가 완전한 분리 컨텐트의 결합(splicing) 때문인 경우, 이러한 오버랩은, 새로운 컨텐트가 이전의 컨텐트와 결합될 새로운 인코딩을 필요로함에 따라 일반적으로 달성하기 어렵다. 따라서, 특정 세그먼트들에 대한 타임라인을 재시작함으로써 미디어 프리젠테이션을 불연속적으로 업데이트하는 능력을 제공하고 가능하면 또한 업데이트 이후 프리젠테이션들의 새로운 세트를 한정하는 것이 유리하다. 또한, 컨텐트가 독립적으로 인코딩되고 세그먼트화되었다면, 컨텐트의 이전 피스(piece)의 글로벌 타임라인 내에 핏팅되게 타임스탬프들를 조정하는 것이 회피된다.

[0291] 업데이트가 기술된 미디어 세그먼트들의 리스트에 단지 새로운 미디어 세그먼트들을 부가하는 것과 같은 극소 원인들(lesser reason)에 대한 것일 때, 또는 URL들의 로케이션이 변경되면, 오버랩 및 연속적 업데이트가 허용될 수 있다.

[0292] 불연속 MPD 업데이트의 경우, 이전 프리젠테이션의 최종(lsat) 세그먼트의 타임라인은 세그먼트의 임의의 샘플의 가장 최종의 프리젠테이션 마지막 시간(latest presentation end time)에 종료된다. 다음 프리젠테이션의 타임라인(또는 보다 정확하게는, 미디어 프리젠테이션의 새로운 파트의 제 1 미디어 세그먼트의 제 1 프리젠테이션 시간, 또한 새로운 기간(period)으로 지칭됨)은 시임리스(seamless) 및 연속적 플레이아웃이 보장되도록 최종 기간의 프리젠테이션의 마지막과 동일한 이 인스턴트에서 통상적으로 그리고 유리하게 시작한다.

[0293] 2가지 경우들이 도 11에 예시된다.

[0294] 세그먼트 경계들(boundaries)에 대해 MPD 업데이트들을 제한하는 것이 바람직하고 유리하다. 세그먼트 경계들에 대한 이러한 변화들 또는 업데이트들을 제한하는 이유는 다음과 같다. 첫째, 각각의 리프리젠테이션, 통상적으로 무비 헤더(Movie Header)에 대한 바이너리 메타데이터(binary metadata)에 대한 변화들은, 적어도 세그먼트 경계들에서 발생할 수 있다. 둘째, 미디어 프리젠테이션 디스크립션은 세그먼트들에 대한 포인터들(URL들)을 포함할 수 있다. 어떤 의미에서, MPD는 미디어 프리젠테이션과 연관된 모든 세그먼트 파일들과 함께 그룹화되는 "엄브렐라(umbrella)" 데이터 구조이다. 이러한 포함 관계를 유지하기 위해, 각각의 세그먼트는 싱글 MPD에 의해 참조될 수 있으며 MPD가 업데이트될 때, 이는 유리하게 세그먼트 경계에서만 업데이트된다.

[0295] 세그먼트 경계들은 일반적으로 정렬될 것이 요구되지 않지만, 서로 다른 소스들로부터 결합되는 컨텐트의 경우에 대해, 그리고 일반적으로 불연속 MPD 업데이트들에 대해, (특정하게, 각각의 리프리젠테이션의 최종(last) 세그먼트는 동일한 비디오 프레임에서 종료될 수 있으며 그 프레임의 프리젠테이션 보다 늦은 프리젠테이션 시작 시간을 갖는 오디오 샘플들을 포함하지 않을 수 있는) 세그먼트 경계들을 정렬하는 것이 합리적이다. 이후 불연속 업데이트는 기간(period)으로 지칭되는 공통 시간 인스턴트(instant)에서 리프리젠테이션들의 새로운 세트를 시작할 수 있다. 이러한 리프리젠테이션들의 새로운 세트의 유효(validity)의 시작 시간은, 예를 들어 기간 시작 시간에 의해 제공된다. 각각의 리프리젠테이션의 상대 시작 시간은 제로로 리셋되

며 기간의 시작 시간은 글로벌 미디어 프리젠테이션 타임라인의 이러한 새로운 기간에서의 리프리젠테이션들의 세트를 배치한다(places).

[0296] 연속적 MPD 업데이트들에 대해, 세그먼트 경계들은 정렬되게 요구되지 않는다. 각각의 대안적 리프리젠테이션의 각각의 세그먼트는 싱글 미디어 프리젠테이션 디스크립션에 의해 제어될 수 있으며, 따라서 추가의 미디어 세그먼트들이 동작하는 MPD에서 기술되지 않는다는 예측(anticipation)에 의해 일반적으로 트리거되는 미디어 프리젠테이션 디스크립션의 새로운 인스턴스들에 대한 업데이트 요청들은 소모될 것으로 예측되는 리프리젠테이션들의 세트를 포함하는 리프리젠테이션들의 소모된 세트에 따라 서로 다른 시간들에서 발생할 수 있다.

[0297] 보다 일반적인 경우에서의 속성들 및 MPD 엘리먼트들에서의 업데이트들을 지원하기 위해, 리프리젠테이션들의 세트 또는 프리젠테이션들만이 아닌 임의의 엘리먼트들은 유효 시간(validity time)과 연관될 수 있다. 따라서, MPD의 특정 엘리먼트들이 업데이트될 필요가 있다면, 예를 들어 리프리젠테이션들의 수가 변경되거나 URL 구성 룰들이 변경되는 경우, 이러한 엘리먼트들은 엘리먼트들의 다수의 카피들을 해체(disjoint) 유효 시간들에 제공함으로써, 지정된 시간들에서 개별적으로 각각 업데이트될 수 있다.

[0298] 유효성은 글로벌 미디어 시간과 바람직하게 연관되어, 유효 시간과 관련하여 연관되어 기술된 엘리먼트는 미디어 프리젠테이션의 글로벌 타임라인의 기간에서 유효하다.

[0299] 앞서 논의된 것처럼, 일 실시예에서, 유효 시간들은 리프리젠테이션들의 풀 세트에 단지 추가된다. 이후 각각의 풀 세트는 기간을 형성한다. 이후 유효 시간은 기간의 시작 시간을 형성한다. 다른말로, 유효 엘리먼트를 이용하는 특정 경우에서, 리프리젠테이션들의 풀 세트는 리프리젠테이션들의 세트에 대한 글로벌 유효 시간에 의해 표시되는 시간에 관한 기간에 대해 유효할 수 있다. 리프리젠테이션들의 세트의 유효 시간은 기간으로 지칭된다. 새로운 기간의 시작시, 이전 세트 리프리젠테이션의 유효성은 만료되며 리프리젠테이션들의 새로운 세트는 유효하다. 기간들의 유효 시간들은 바람직하게 해체된다는 것이 다시 주목된다.

[0300] 앞서 논의된 것처럼, 미디어 프리젠테이션 디스크립션에 대한 변화들은 세그먼트 경계들에서 발생하며, 각각의 리프리젠테이션에 대해, 엘리먼트 변화는 실제로 다음 세그먼트 경계에서 발생한다. 이후 클라이언트는 미디어의 프리젠테이션 시간 내의 시간의 각각의 인스턴트에 대해 세그먼트들의 리스트를 포함하는 유효 MPD를 형성할 수 있다.

[0301] 불연속 블록 결합은, 블록들이 서로 다른 리프리젠테이션들로부터, 또는 서로 다른 컨텐트로부터, 예를 들어 컨텐트의 세그먼트 및 광고로부터 미디어 데이터를 포함하는 경우들에서, 또는 다른 경우들에서 적절할 수 있다. 이는 블록 경계들에서만 프리젠테이션 메타데이터에 대한 변화들이 발생하는 블록 요청 스트리밍 시스템에서 요구될 수 있다. 이는, 블록 내의 미디어 디코더 파라미터들을 업데이트하는 것은 블록들 사이에서만 이들을 업데이트하는 것보다 더 복잡할 수 있기 때문에, 구현 이유들에 대해 유리할 수 있다. 이 경우, 앞서 기술된 것처럼 유효 인터벌들은 근사치(approximate)로서 해석될 수 있고, 이로 인해 엘리먼트는 지정된 유효 인터벌의 시작 보다 앞서지 않은 제 1 블록 경계로부터 지정된 유효 인터벌의 마지막 보다 앞서지 않는 제 1 블록 경계까지 유효한 것으로 고려되게 유리하게 지정될 수 있다.

[0302] 블록-요청 스트리밍 시스템에 대한 새로운 강화들을 기술하는 상기 예시적 실시예는 미디어 프리젠테이션들에 대한 변화들이란 명칭의 이후 제시되는 섹션에 기술된다.

[0303] 세그먼트 지속시간 시그널링(Segment Duration Signalling)

[0304] 불연속 업데이트들은 프리젠테이션을 기간으로서 지칭되는 일련의 해체 인터벌들로 효과적으로 분할한다. 각각의 기간은 미디어 샘플 타이밍에 대한 그 자신의 타임라인을 갖는다. 기간 내의 리프리젠테이션의 미디어 타이밍은 각각의 기간 동안 또는 기간내의 각각의 리프리젠테이션 동안 세그먼트 지속시간들의 개별(separate) 콤팩트한 리스트를 지정함으로써 유리하게 표시될 수 있다.

[0305] 예를 들어 기간 시작 시간으로 지칭되며, MPD 내의 엘리먼트들과 연관되는 속성은 미디어 프리젠테이션 시간 내의 특정 엘리먼트들의 유효 시간을 지정할 수 있다. 이러한 속성은 MPD의 임의의 엘리먼트들에 부가될 수 있다(유효성에 할당될 수 있는 속성들은 엘리먼트들에 대해 변경될 수 있다).

[0306] 불연속 MPD 업데이트들에 대해, 모든 리프리젠테이션들의 세그먼트들은 불연속성으로 중단될 수 있다. 이는 일반적으로 적어도, 불연속성이 이전의 최종 세그먼트가 이전의 것들과 서로 다른 지속시간을 갖는다는 것을

의미한다. 시그널링 세그먼트 지속시간은 각 세그먼트에 대한 개별 지속시간을 표시하거나 또는 모든 세그먼트들이 동일한 지속시간을 갖는다는 것을 표시하는 것을 수반할 수 있다. 이들 중 다수가 동일한 지속시간을 갖는 경우 효과적인 세그먼트 지속시간들의 리스트에 대한 콤팩트한 리프리젠테이션을 갖는 것이 바람직할 수 있다.

[0307] 하나의 리프리젠테이션 또는 리프리젠테이션들의 세트에서 각각의 세그먼트의 지속시간들은 불연속 업데이트의 시작, 즉 MPD에 최종 미디어 세그먼트가 기술될 때까지 기간의 시작로부터의 싱글 인터벌 동안 모든 세그먼트 지속시간들을 지정하는 싱글 스트링으로 유리하게 실행될 수 있다. 일 실시예에서, 이러한 엘리먼트의 포맷은 세그먼트 지속시간 엔트리들의 리스트를 포함하는 프로덕션(production)에 따르는 텍스트 스트링이며, 각각의 엔트리는 지속시간 속성 *dur*, 및 이러한 리프리젠테이션이 제 1 엔트리의 지속시간 *<dur>*의 제 1 엔트리 세그먼트들의 *<mult>*를, 이후에 제 2 엔트리의 지속시간 *<dur>*의 제 2 엔트리 세그먼트들의 *<mult>* 등을 포함하는 것을 표시하는 속성의 선택적 멀티플라이어 *mult*를 포함한다.

[0308] 각각의 지속시간 엔트리는 하나 이상의 세그먼트들의 지속시간을 지정한다. *<dur>* 값이 "*" 문자 및 숫자로 이어지면, 이러한 숫자는 이러한 지속시간(초 단위)을 갖는 연이은(consecutive) 세그먼트들의 수를 지정한다. 멀티플라이어 부호 "*"가 없다면, 세그먼먼트들의 수는 1이다. "*"가 존재하고 숫자가 후속하지 않은다면, 모든 후속 세그먼트들은 지정된 지속시간을 가지며 리스트에는 추가 엔트리들이 없을 수 있다. 예를 들어, 스트링 "30*"은 30초의 지속시간을 갖는 모든 세그먼트들을 의미한다. 스트링 "30* 12 10.5"은 30초의 지속시간을 갖는 12개의 세그먼트들에 이어 10.5초의 지속시간을 갖는 1개의 세그먼트를 표시한다.

[0309] 세그먼트 지속시간들이 각각의 대안적 리프리젠테이션에 대해 개별적으로 지정되는 경우, 각각의 인터벌 내의 세그먼트 지속시간들의 합은 각각의 리프리젠테이션에 대해 동일할 수 있다. 비디오 트랙들의 경우, 인터벌은 각각의 대안적 리프리젠테이션에서 동일한 프레임에 대해 종료될 수 있다.

[0310] 통상의 당업자들은, 이 개시물을 판독시에, 콤팩트한 방식으로 세그먼트 지속시간들을 표현하는 유사한 그리고 등가적인 방식들을 찾을 수 있을 것이다.

[0311] 다른 실시예에서, 세그먼트의 지속시간은 신호 지속시간 속성 <duration>에 의한 최종의 것을 제외하고 리프리젠테이션에서의 모든 세그먼트들에 대해 일정하게 시그널링된다. 불연속 업데이트 이전의 최종 세그먼트의 지속시간은 다음 불연속 업데이트의 시작 포인트 또는 새로운 기간의 시작이 제공되는 한 더 짧을 수 있고, 이는 이후 다음 기간의 시작에 이르는 최종 세그먼트의 지속시간을 암시한다.

[0312]

[0313] 리프리젠테이션 메타데이터에 대한 변화들 및 업데이트들

[0314] 무비 헤더 "moov" 변화들과 같이 바이너리 코딩된 리프리젠테이션 메타데이터의 변화들을 표시하는 것은 서로 다른 방식들로 달성될 수 있다: (a) MPD에서 참조되는 개별 파일에서 모든 리프리젠테이션에 대한 하나의 moov 박스가 있을 수 있다, (b) 각각의 대안적 리프리젠테이션(Alternative Representation)에 참조되는 개별 파일에서 각각의 대안적 리프리젠테이션에 대한 하나의 moov 박스가 있을 수 있다, (c) 각각의 세그먼트는 moov 박스를 포함할 수 있고 따라서 자급식(self-contained)이며, (d) MPD와 함께 하나의 3GP 파일에서 모든 리프리젠테이션에 대한 moov 박스가 있을 수 있다.

[0315] (a) 및 (b)의 경우에, 싱글 'moov'는 보다 많은 'moov' 박스들이 이들의 유효성이 해제되는 한 MPD에서 참조될 수 있다는 점에서, 상기한 유효성 개념과 유리하게 조합될 수 있다는 것이 주목된다. 예를 들어, 기간 경계의 정의로, 구(old) 기간에서 'moov'의 유효성은 새로운 기간의 시작으로 만료될 수 있다.

[0316] 옵션 (a)의 경우, 싱글 moov 박스에 대한 레퍼런스가 유효 엘리먼트에 대해 할당될 수 있다. 다중 프리젠테이션 헤더들이 허용될 수 있지만 한번에는 단지 하나만이 유효할 수 있다. 다른 실시예에서, 앞서 정의된 것처럼 전체 기간 또는 기간의 리프리젠테이션들의 전체 세트의 유효 시간은, 통상적으로 moov 헤더로서 제공되는 이러한 리프리젠테이션 메타데이터에 대한 유효 시간으로서 사용될 수 있다.

[0317] 옵션 (b)의 경우, 각각의 리프리젠테이션의 moov 박스에 대한 레퍼런스가 유효 엘리먼트에 할당될 수 있다. 다중 리프리젠테이션 헤더들이 허용될 수 있지만, 한번에 단지 하나만이 유효할 수 있다. 다른 실시예에서, 앞서 정의된 것처럼 전체 기간 또는 전체 리프리젠테이션의 유효 시간은, 통상적으로 moov 헤더로서 제공되는 이러한 리프리젠티에션 메타데이터에 대한 유효 시간으로서 사용될 수 있다.

[0318] 옵션들 (c)의 경우에, MPD에 시그널링이 부가되지 않을 수 있지만, moov 박스가 임의의 향후(upcoming) 세그

먼트들에 대해 변할 것임을 표시하도록 미디어 스트림의 추가 시그널링이 부가될 수 있다. 이는 "세그먼트 메타데이터 내의 시그널링 업데이트"란 문맥으로 아래에서 추가로 설명된다.

[0319] 세그먼트 메타데이터 내의 시그널링 업데이트

[0320] 잠재적 업데이트들에 대한 지식(knowledge)을 얻기 위해 미디어 프리젠테이션 디스크립션의 빈번한 업데이트들을 방지하기 위해, 미디어 세그먼트들과 함께 임의의 이러한 업데이트들을 시그널링하는 것이 유리하다. 미디어 프리젠테이션 디스크립션과 같이 업데이트된 메타데이터가 이용가능하며 액세스가능한 세그먼트 리스트들의 생성을 성공적으로 지속하기 위해 특정량의 시간 내로 액세스될 것임을 표시할 수 있는 그들 자신의 미디어 세그먼트들 내의 추가 엘리먼트 또는 엘리먼트들이 제공될 수 있다. 또한, 이러한 엘리먼트들은 파일 식별자, 예컨대 URL 또는 업데이트된 메타데이터 파일에 대한 파일 식별자를 구성하는데 이용될 수 있는 정보를 제공할 수 있다. 업데이트된 메타데이터 파일은 유효 인터벌들에 의해 동반되는 추가 메타데이터와 함께 유효 인터벌들을 표시하기 위해 변형되는 프리젠테이션에 대한 오리지널 메타데이터 파일에 제공되는 것과 동일한 메타데이터를 포함할 수 있다. 이러한 표시는 미디어 프리젠테이션에 대해 이용가능한 리프리젠테이션들 모두의 미디어 세그먼트들에 제공될 수 있다. 미디어 블록 내의 이러한 표시 검출시, 블록 요청 스트리밍 시스템을 액세스하는 클라이언트는 파일 다운로드 프로토콜 또는 업데이트된 메타데이터 파일을 리트리브하기 위한 다른 수단을 이용할 수 있다. 이로써, 클라이언트에게는 변화들이 발생할 또는 발생하는 시간 및 미디어 프리젠테이션 디스크립션에서의 변화들에 대한 정보가 제공된다. 유리하게, 각각의 클라이언트는 "폴링"보다는 이러한 변화가 발생하고 가능한 업데이트들 또는 변화들에 대한 파일이 다수번 수신될 때만 업데이트된 미디어 프리젠테이션 디스크립션을 요청한다.

[0321] 변화들의 예들은, 트랙들 또는 코덱 파라미터들이 포함되는, 리프리젠테이션들의 부가 또는 제거, 비트-레이트에서의 변화와 같은 하나 이상의 리프리젠테이션에 대한 변화들, 레졸루션, 어스펙트 비, 그리고 URL 구성 룰들에 대한 변화들, 예를 들어 광고를 위한 서로 다른 원서버(origin server)를 포함한다. 일부 변화들은 리프리젠테이션과 연관된 무비 헤더("moov") 어톰(atom)과 같은 초기화(initialization) 세그먼트에만 영향을 미치는 반면, 다른 변화들은 미디어 프리젠테이션 디스크립션(MPD)에 영향을 미칠 수 있다.

[0322] 온-디멘드 컨텐트의 경우, 이러한 변화들 및 이들의 타이밍은 사전에 인지될 수 있으며 미디어 프리젠테이션 디스크립션에서 시그널링될 수 있다.

[0323] 라이브 컨텐트에 대해, 변화들은 이들이 발생할 때까지는 인지될 수 없다. 하나의 솔루션은, 특정 URL에서 이용가능한 미디어 프리젠테이션 디스크립션이 동적으로 업데이트되게 허용하고 클라이언트들이 변화들을 검출하기 위해 이러한 MPD를 정상적으로 요청하게 요구하는 것이다. 이러한 솔루션은 확장성(원서버 부하 및 캐시 효율성)에 관해 단점을 갖는다. 다수의 뷰어들을 갖는 시나리오에서, 캐시들은,이전 버전이 캐시로부터 만료된 이후에 그리고 새로운 버전이 수신되기 이전에 MPD에 대한 다수의 요청들을 수신하며 이들 모두가 원서버로 포워딩될 수 있다. 원서버는 MPD의 각각의 업데이트된 버전에 대한 캐시들로부터 요청들을 일정하게 프로세스할 것을 요구할 수 있다. 또한, 업데이트들은 미디어 프리젠테이션에서의 변화들로 쉽게 시간-정렬되지 않을 수 있다.

[0324] HTTP 스트리밍의 장점들 중 하나는 확장성에 대한 표준 웹 인트라스트럭처 및 서비스들을 이용하는 능력이기 때문에, 선호되는 솔루션은 "정적"(즉, 캐시가능) 파일들만을 수반하며 이들이 변경되는 경우 볼 수 있는 클라이언트 "폴링" 파일들에 의존하지 않는다.

[0325] 솔루션들은 적응식(Adaptive) HTTP 스트리밍 미디어 프리젠테이션에서 "moov" 어톰들과 같은 바이너리 리프리젠테이션 메타데이터 및 미디어 프리젠테이션 디스크립션을 포함하는 메타데이터의 업데이트를 해결(resolve)하도록 논의되고 제안된다.

[0326] 라이브 컨텐트의 경우에 대해, MPD 또는 "moov"가 변할 수 있는 포인트들은 MPD가 구성되는 시기를 인지하지 못할 수 있다. 업데이트들을 위한 체크에 대한 MPD의 빈번한 "폴링"이 일반적으로 방지되어야 하기 때문에, 대역폭 및 확장성 이유들에 대해, MPD에 대한 업데이트들은 세그먼트 파일들 자신들에서 "대역내(in band)"로 표시된다, 즉 각각의 미디어 세그먼트는 업데이트들을 표시하기 위한 옵션을 가질 수 있다. 상기의 세그먼트 포맷들 (a) 내지 (c)에 따라, 상이한 업데이팅이 시그널링될 수 있다.

[0327] 일반적으로, 하기 표시는 세그먼트 내의 신호에 유리하게 제공될 수 있다: 현재 세그먼트의 시작 시간 보다 더 큰 시작 시간을 갖는 임의의 다음 세그먼트 또는 이러한 리프리젠테이션 내의 다음 세그먼트를 요청하기

이전에 MPD가 업데이트될 수 있는 표시자(indicator). 다음의 것 보다 늦은 임의의 세그먼트에서만 업데이트가 발생하도록 요구된다는 것을 표시하는 업데이트가 사전에 발표될 수 있다. 이러한 MPD 업데이트는 또한, 미디어 세그먼트의 로케이터가 변경되는 경우 무비 헤더들과 같은 바이너리 리프리젠테이션 메타데이터를 업데이트하는데 이용될 수 있다. 또 다른 신호는 이러한 세그먼트의 완료로, 시간이 진행한 더 이상의 어떠한 세그먼트들도 요청될 수 없다는 것을 표시할 수 있다.

[0328] 세그먼트들이 세그먼트 포맷 (c)에 따라 포맷되는 경우, 즉 각각의 미디어 세그먼트가 무비 헤더와 같은 자체-초기화(self-initialising) 메타데이터를 포함할 수 있고, 그러면 차후 세그먼트가 업데이트된 무비 헤더(moov)를 포함하는 것을 표시하는 또 다른 신호가 부가될 수 있다. 이는 유리하게 무비 헤더가 세그먼트에 포함되게 허용하지만, 무비 헤더는 리프리젠테이션들의 스위칭시 랜덤 액세스 또는 검색(seeking)의 경우에 또는 이전의 세그먼트가 무비 헤더 업데이트를 표시하는 경우 클라이언트에 의해서만 요청될 필요가 있다. 다른 경우들에서, 클라이언트는 무비 헤더를 다운로드로부터 배제하는 세그먼트에 대한 바이트 범위 요청을 발행할 수 있으며, 이로 인해 대역폭이 유리하게 절감된다.

[0329] 또 다른 실시예에서, MPD 업데이트 표시가 시그널링되면, 신호는 또한 업데이트된 미디어 프리젠테이션 디스크립션에 대한 URL과 같은 로케이터를 포함할 수 있다. 업데이트된 MPD는, 불연속 업데이트들의 경우에 새로운(new) 그리고 구(old) 기간과 같은 유효 속성들을 이용하여, 업데이트 이전 및 업데이트 이후 모두의 프리젠테이션을 기술할 수 있다. 이는 하기 추가로 시작되는 것처럼 시간-이동 뷰잉(viewing)을 허용하도록 유리하게 사용될 수 있으나, 또한 MPD 업데이트가 그것을 포함하는 변화들이 시행(take effect)되기 이전의 임의의 시간에서 시그널링되게 유리하게 허용한다. 클라이언트는 새로운 MPD를 즉시 다운로드할 수 있고 이를 진행중인(ongoing) 프리젠테이션에 적용할 수 있다.

[0330] 특정 구현에서, 미디어 프리젠테이션 디스크립션에 대한 임의의 변화들의 시그널링, moov 헤더들 또는 프리젠테이션의 마지막은 ISO 베이스 미디어 파일 포맷의 박스 구조를 이용하여 세그먼트 포맷의 룰들을 따라 포맷되는 스트리밍 정보 박스에 포함될 수 있다. 이러한 박스는 서로 다른 업데이트들의 임의의 것에 대한 특정 신호를 제공한다.

[0331] *스트리밍 정보 박스(Streaming Information Box)*

[0332] 정의(Definition)

[0333] 박스 타입 : 'sinf'

[0334] 컨테이너: 없음

[0335] 의무사항(mandatory) : 없음

[0336] 양 : 0 또는 1

[0337] 스트리밍 정보 박스는 파일이 그의 일부인 스트리밍 프리젠테이션에 관한 정보를 포함한다.

[0338] 신택스(Syntax)

```
aligned(8) class StreamingInformationBox
```

[0339]

```
extends FullBox('sinf') {
```

[0340]

```
unsigned int(32) streaming_information_flags;
```

[0341]

```
    /// The following are optional fields

    string mpd_location
}
```

[0342]

[0343] 씨맨틱스(Semantics)

[0344] streaming_information_flags 는 하기의 것의 제로 또는 그 이상의 논리적 OR을 포함한다.

[0345] 0x00000001 무비 헤더 업데이트 따름

[0346] 0x00000002 프리젠테이션 디스크립션 업데이트

[0347] 0x00000004 프리젠테이션 종료

[0348] mpd_location 은 *Presentation Description update* 플래그가 설정되는 경우 그리고 이 경우에만 존재하며 새로운 미디어 프리젠테이션 디스크립션에 대한 균일한 자원 로케이터를 제공한다.

[0349] 라이브 서비스들에 대한 MPD 업데이트들에 대한 예시적 사용 경우

[0350] 서비스 제공자들이 본 명세서에 기술된 강화된 블록-요청 스트리밍을 이용하여 라이브 풋볼 이벤트를 제공하길 원한다고 가정한다. 아마도 수백만명의 사용자들은 이벤트의 프리젠테이션을 액세스하길 원할 것이다. 라이브 이벤트는 광고가 부가되어야 하는, 타임 아웃이 호출될 때 또는 동작시 다른 소강상태(lull)에 있을 때 브레이크들(breaks)에 의해 우발적으로 중단된다. 통상적으로, 브레이크들에 대한 정확한 타이밍의 사전 통지가 없거나 거의 없다.

[0351] 서비스 제공자들은 라이브 이벤트 동안 임의의 컴포넌트 페일(components fail)의 경우 심리스 스위치-오버를 가능케하는 리던던트 인프라구조(예를 들면, 인코더들 및 서버들)을 제공할 필요가 있을 수 있다.

[0352] 사용자가 안나가 그녀의 모바일 디바이스가 있는 버스 상에서의 서비스를 액세스하고 서비스를 즉시 이용가능하다 가정한다. 다음 그녀의 옆에는 그의 랩톱 상의 이벤트를 주시(watche)하는 또 다른 사용자 폴이 앉아있다. 골이 득점되고 둘(both)은 동시에 이 이벤트를 축하한다. 폴은, 게임에서의 첫 번째 골이 훨씬 흥분되었다는 것을 안나에게 말하며, 안나는 그녀가 시간상 30분 전 이벤트를 시청할 수 있도록 서비스를 이용한다. 골을 본 후에, 그녀는 라이브 이벤트로 다시 복귀한다.

[0353] 이러한 사용의 경우를 어드레스하기 위해, 서비스 제공자는 MPD를 업데이트할 수 있어야 하며, 업데이트된 MPD가 이용가능하다는 것을 클라이언트들에게 시그널링해야 하며 실시간 근접한 데이터가 제시될 수 없도록 클라이언트들이 스트리밍 서비스를 액세스하도록 허용해야 한다.

[0354] MPD를 업데이트하는 것은, 본 명세서의 다른 곳에서 설명된 것처럼, 세그먼트들의 전달을 위해 비동기식 방식으로 실행할 수 있다. 서버는 MPD가 일부 시간 동안 업데이트되지 않는다는 보장들(guarantees)을 수신기에게 제공할 수 있다. 서버는 현재의 MPD에만 의존할 수 있다. 그러나, 일부 최소 업데이트 기간 이전에 MPD가 업데이트될 때 명시적 시그널링은 요구되지 않는다.

[0355] 완벽한 동기식 플레이아웃은 클라이언트가 서로 다른 MPD 업데이트 인스턴스들에 대해 동작할 수 있기 때문에 거의 달성될 수 없고, 따라서 클라이언트들은 드리프트(drift)를 가질 수 있다. MPD 업데이트들을 사용으로, 서버는 변화들을 전달할 수 있고 클라이언트들은 프리젠테이션 동안에도, 변화들에 대해 변경될 수 있다. 세그먼트-바이-세그먼트(segment-by-segment) 베이시스에 대한 대역내 시그널링은 MPD의 업데이트를 표시하기 위해 이용될 수 있으며, 이러한 업데이트들은 세그먼트 경계들로 제한될 수 있지만, 대부분의 애플리케이션들을 허용해야 한다.

[0356] MPD 업데이트가 요구된다는 것을 시그널링하기 위해 세그먼트들의 시작시에 부가되는 선택적 MPD 업데이트 박스 뿐만 아니라 MPD의 월-클록 타임의 공개 시간(publishing time)을 제공하는 MPD 엘리먼트가 부가될 수 있다. 업데이트들은 MPD들처럼 계층적으로 수행될 수 있다.

[0357] MPD "공개 시간"은 MPD에 대한 고유 식별자들 및 MPD가 발행되는 시기를 제공한다. 또한 이는 업데이트 프로시저들(procedures)에 대한 앵커를 제공한다.

[0358] MPD 업데이트 박스는 "styp" 박스 이후 MPD에서 발견될 수 있으며 컨테이너가 없고, 의무사항이 없고 0 또는 1의 양(quantity)을 필요로 하는, 박스 타입="mupe"에 의해 한정된다. MPD 업데이트 박스는 세그먼트가 그의 일부인 미디어 프리젠테이션에 관한 정보를 포함한다.

[0359] 예시적 syntax는 다음과 같다:

```
aligned(8) class MPDUpdateBox
        extends FullBox('mupe') {
        unsigned int(3) mpd information flags;
        unsigned int(l) new-location flag;
```

[0360]

```
        unsigned int(28) latest_mpd_update time;
        /// The following are optional fields
        string mpd_location
    }
```

[0361]

[0362] 클래스 MPDUpdateBox의 다양한 오브젝트들의 시맨틱들은 다음과 같을 수 있다:

[0363] mpd_information_flags : 하기의 것들의 제로 또는 그 이상의 논리 OR

[0364] 0x00 미디어 프리젠테이션 디스크립션 지금 업데이트

[0365] 0x01 미디어 프리젠테이션 디스크립션 사전(ahead) 업데이트

[0366] 0x02 프리젠테이션 종료

[0367] 0x03-0x07 예비됨

[0368] new_location flag : 1로 설정되면, 새로운 미디어 프리젠테이션 디스크립션이 mpd_location에 지정된 새로운 위치에서 이용가능하다.

[0369] latest_mpd_update time : MPD 업데이트가 가장늦은 MPD의 MPD 발행 시간에 관해 필요할 때 시간(ms 단위)을 지정한다. 클라이언트는 현재중의(between now) 임의의 시간에서 MPD를 업데이트하도록 선택할 수 있다.

[0370] mpd_location : new_location_flag가 설정되는 경우 그리고 이 경우에만 제시되며, 만약 그렇다면 mpd_location은 새로운 미디어 프리젠테이션 디스크립션에 대한 균일한 자원 로케이터를 제공한다.

[0371] 업데이트들에 의해 사용되는 대역폭이 이슈이면, 서버는 단지 이러한 부분들만이 업데이트되도록 특정 디바이스 용량들에 MPD들을 제공할 수 있다.

[0372] 시간-이동 뷰잉 및 네트워크 PVR

[0373] 시간-이동 뷰잉이 지원될 때, 세션의 라이프-타임에 대해 2개 또는 그 이상의 MPD들 또는 무비 헤더들이 유효하게 발생할 수 있다. 이러한 경우, 필요할 때 MPD를 업데이트하나, 유효 메커니즘 또는 기간 콘셉을 부가함으로써, 유효 MPD가 전체 타임-윈도우에 존재할 수 있다. 이는 서버가, 임의의 MPD 및 무비 헤더가 타임-이동 뷰잉을 위한 유효 시간-윈도우 내에 있는 임의의 시간 기간 동안 발표되는 것을 보장할 수 있다. 그의 현재 프리젠테이션 시간에 대한 그의 이용가능한 MPD 및 메타데이터가 유효하다는 것을 보장하는 것은 클라이언트에게 달려있다. 최소 MPD 업데이트들만을 이용하는 네트워크 PVR 세션에 대한 라이브 세션의 마이그래이션(migration)이 또한 지원될 수 있다.

[0374]

[0375] 특정 미디어 세그먼트들

[0376] ISO/IEC 14496-12의 파일 포맷이 블록 요청 스트리밍 시스템 내에서 사용될 때의 이슈(issue)는 앞서 기술된 것처럼, 연이은 시간 세그먼트들에 배열된 다수의 파일들에서의 프리젠테이션의 싱글 버전에 대한 미디어 데이터를 저장하는 것이 유리할 수 있다는 것이다. 더구나, 각각의 파일이 랜덤 액세스 포인트로 시작되게 정렬하는데 유리할 수 있다. 또한, 비디오 인코딩 프로세스 동안 탐색 포인트들의 위치들을 선택하고 인코딩 프로세스 동안 구성된 탐색 포인트들의 선택에 기초하여 탐색 포인트를 각각 시작하는 다수의 파일들로 프리젠테이션을 세그먼트화하는 것이 유리할 수 있으며, 각각의 랜덤 액세스 포인트는 파일의 시작시 배치되거나 또는 배치되지 않을 수 있지만 각각의 파일은 랜덤 액세스 포인트로 시작한다. 앞서 기술된 특성들을 갖는 하나의 실시예에서, 프리젠테이션 메타데이터, 또는 미디어 프리젠테이션 디스크립션은 각각의 파일의 정확한 지속시간을 포함할 수 있으며, 여기서 지속시간은 예를 들어 파일의 비디오 미디어의 시작 시간과 다음 파일의 비디오 미디어의 시작 시간 간의 차를 의미하도록 취해진다. 프리젠테이션 메타데이터에서의 이러한 정보에 기초하여, 클라이언트는 각각의 파일 내의 미디어에 대한 로컬 타임라인 및 미디어 프리젠테이션에 대한 글로벌 타임라인 간의 맵핑을 구성할 수 있다.

[0377] 다른 실시예에서, 프리젠테이션 메타데이터의 크기는 각 파일 또는 세그먼트가 동일한 지속시간을 갖는다는 것을 대신 지정함으로써 유리하게 감소될 수 있다. 그러나, 이 경우 그리고 미디어 파일들이 상기 방법에 따라 구성되는 경우, 각각의 파일의 지속시간은 미디어 프리젠테이션 디스크립션에 지정된 지속시간과 정확히 동일하지 않을 수 있고, 이는 랜덤 액세스 포인트가 파일의 시작으로부터 정확하게 지정된 지속시간인 포인트에 존재하지 않을 수 있기 때문이다.

[0378] 앞서 언급된 모순(discrepancy)에도 불구하고 블록-요청 스트리밍 시스템의 정확한 동작을 위해 제공되는 본 발명의 추가 실시예가 이제 기술된다. 이 방법에서는, 글로벌 프리젠테이션 타임라인에 대한 (파일내의 미디어 샘플들의 디코딩 및 컴포지션 타임스탬프들(composition timestamps)이 ISO/IEC 14496-12에 따라 지정되는 타임스탬프 제로로부터 시작하는 타임라인을 의미하는) 파일 내의 미디어의 로컬 타임라인의 맵핑을 지정하는 엘리먼트를 각각의 파일 내에 제공할 수 있다. 이러한 맵핑 정보는 로컬 파일 타임라인 내의 제로 타임스탬프에 해당하는 글로벌 프리젠테이션 시간내의 싱글 타임스탬프를 포함할 수 있다. 맵핑 정보는 프리젠테이션 메타데이터내에 제공된 정보에 따라 파일의 시작에 해당하는 글로벌 프리젠테이션 시간 및 로컬 파일 타임라인내의 제로 타임스탬프에 해당하는 글로벌 프리젠테이션 시간 간의 차를 지정하는 오프셋 값을 대안적으로 포함할 수 있다.

[0379] 이러한 박스들의 예는 예를 들어, 트랙 프래그먼트 미디어 조절 박스('tfma')와 함께 트랙 프래그먼트 조절 박스('tfad') 또는 트랙 프래그먼트 디코드 시간('tfdt') 박스일 수 있다.

[0380] 세그먼트 리스트 생성을 포함하는 예시적 클라이언트

[0381] 예시적 클라이언트가 이제 기술될 것이다. 이는 MPD의 적절한 생성 및 업데이트들을 보장하기 위해 서버에 대한 레퍼런스(reference) 클라이언트로서 이용될 수 있다.

[0382] HTTP 스트리밍 클라이언트는 MPD에 제공되는 정보에 의해 가이드된다. 클라이언트는 시간 T, 즉 MPD를 성공적으로 수신할 수 있었던 시간에서 수신된 MPD에 대한 액세스를 갖는 것으로 가정된다. 성공적인 수신을 결정하는 것은, 이전의 성공적인 수신 이래로 MPD가 업데이트되지 않았다는 것을 클라이언트가 검증하거나 또는 클라이언트가 업데이트된 MPD를 획득하는 것을 포함할 수 있다.

[0383] 예시적 클라이언트 작용(behaviour)이 도입된다. 사용자에 대한 연속적인 스트리밍 서비스를 제공하기 위해, 클라이언트는 먼저 MPD를 분석(parse)하고 현재 시스템 시간에서 클라이언트-로컬 시간에 대한 각각의 리프리젠테이션에 대한 액세스가능 세그먼트들의 리스트를 생성하며, 가능한 플레이-리스트들 또는 URL 구성 룰들을 사용하여 하기에 상세히 기재되는 것처럼 세그먼트 리스트 생성 프로시저들을 고려한다. 다음, 클라이언트는 리프리젠테이션 속성들의 정보 및 다른 정보, 예를 들어 이용가능한 대역폭 및 클라이언트 능력들에 기초하여 하나 또는 다수의 리프리젠테이션들을 선택한다. 그룹화에 따라 리프리젠테이션들은 다른 프리젠테이션들과 함께 또는 독립적으로 제시될 수 있다.

[0384] 각각의 리프리젠테이션에 대해, 클라이언트는, 존재할 경우, 리프리젠테이션에 대한 "moov" 헤더와 같은 바이너리 메타데이터 및 선택된 리프리젠테이션의 미디어 세그먼트들을 획득한다. 클라이언트는 가능한 세그먼트 리스트를 이용하여, 세그먼트들 또는 바이트 범위들의 세그먼트들을 요청함으로써 미디어 컨텐트를 액세스한다. 클라이언트는, 프리젠테이션 시작 이전에 초기에 미디어를 버퍼링하며 일단 프리젠테이션이 시작되면,

클라이언트는 MPD 업데이트 프로시저들을 고려하여 세그먼트들 또는 세그먼트들의 일부들(parts)을 연속적으로 요청함으로써 미디어 컨텐트 소모를 지속한다.

[0385] 클라이언트는, 업데이트된 MPD 정보 및/또는 그의 환경으로부터 업데이트된 정보, 예를 들어 이용가능한 대역폭의 변화를 고려하여 리프리젠테이션들을 스위치할 수 있다. 랜덤 액세스 포인트를 포함하는 미디어 세그먼트에 대한 임의의 요청으로, 클라이언트는 서로 다른 리프리젠테이션으로 스위치될 수 있다. 전방 이동시, 즉, 현재 시스템 시간(프리젠테이션에 관한 시간을 표현하기 위한 "NOW 시간"으로 지칭됨)이 진행중(advancing)일 때, 클라이언트는 액세스가능한 세그먼트들을 소모한다. NOW 시간에서의 각각의 진행(advance)으로, 클라이언트는 본 명세서에서 지정된 프로시저들에 따라 각각의 리프리젠테이션에 대한 액세스가능한 세그먼트들의 리스트를 아마도 확장할 것이다.

[0386] 미디어 프리젠테이션의 마지막이 아직 도달되지 않았고 현재의 플레이백 시간이 클라이언트가 임의의 소모를 위해 MPD에 시작된 미디어의 미디어를 없앨 것으로 예상되거나 또는 프리젠테이션이 소모될 것으로 예상되는 쓰레숄드(threshold) 내에서 취해지면(get), 클라이언트는 새로운 페치(fetch) 시간 수신 시간 T로, MPD의 업데이트를 요청할 수 있다. 일단 수신되면, 클라이언트는 가능한 업데이트된 MPD 및 액세스가능한 세그먼트 리스트들의 생성시 새로운 시간 T을 고려한다. 도 29는 클라이언트에서의 서로 다른 시간들에서 라이브 서비스들에 대한 프로시저를 예시한다.

[0387] 액세스가능한 세그먼트 리스트 생성

[0388] HTTP 스트리밍 클라이언트가 MPD에 대한 액세스를 갖고 월-클록 타임 NOW 동안 액세스가능한 세그먼트 리스트를 생성하기 원할 수 있다고 가정한다. 클라이언트는 특정한 정확성으로 글로벌 시간 레퍼런스에 대해 동기화되나, 요구되는 HTTP 스트리밍 서버에 대한 동기화는 유리하게 지시하지 않는다.

[0389] 각각의 리프리젠테이션에 대한 액세스가능한 세그먼트 리스트는 세그먼트 로케이터 및 세그먼트 시작 시간의 리스트 쌍으로서 바람직하게 정의되며, 여기서 세그먼트 시작 시간은 일반성(generality)의 손실 없이, 리프리젠테이션의 시작과 관련되는 것으로 정의될 수 있다. 리프리젠테이션의 시작은 이러한 콘셉이 적용되기 이전에 또는 이러한 콘셉이 적용되는 경우의 시작과 정렬될 수 있다. 그렇지 않다면, 리프리젠테이션 시작은 미디어 프리젠테이션의 시작일 수 있다.

[0390] 클라이언트는 예를 들어 본 명세서에 추가로 정의된 것처럼, URL 구성 룰들 및 타이밍을 이용한다. 시작된 세그먼트들의 리스트가 일단 획득되면, 이러한 리스트는 액세스 가능한 것들에 대해 추가로 제한되며, 이는 완벽한 미디어 프리젠테이션의 세그먼트들의 서브세트일 수 있다. 구성은 클라이언트 NOW 시간에서 클록의 현재 값에 의해 제어된다. 일반적으로, 세그먼트들은 이용가능성 시간들의 세트 내에서의 임의의 시간 NOW에 대해서만 이용될 수 있다. 이러한 윈도우를 벗어난 시간들 NOW에 대해, 어떠한 세그먼트들도 이용될 수 없다. 또한, 라이브 서비스들에 대해, 일부 시간 체크타임이 어디까지(how far into) 추가 미디어가 기술되는지에 대한 정보를 제공하는 것으로 가정한다. 체크타임은 MPD-문서화(documented) 미디어 시간 축상에 정의된다; 클라이언트의 플레이백 시간이 체크타임에 도달할 때, 이는 유리하게 새로운 MPD를 요청한다.

[0391] 클라이언트의 플레이백 시간이 체크타임에 도달할 때, 이는 유리하게 새로운 MPD를 요청한다.

[0392] 다음, 세그먼트 리스트는 MPD 속성 TimeShiftBufferDepth 과 함께 체크타임에 의해 추가로 제한되어, 단지 이용가능한 미디어 세그먼트들은 미디어 세그먼트의 시작 시간의 합에 대한 것들이고 리프리젠테이션 시작 시간은 NOW 마이너스 timeShiftBufferDepth 마이너스 체크타임 또는 NOW 중 어느 하나의 더 작은 값 및 최종 시작되는 세그먼트의 지속시간에 속할 수 있다(falls in).

[0393] 확장가능 블록들

[0394] 이용가능한 대역폭은 때로, 수신기에서 현재 수신되고 있는 블록 또는 블록들이 리프리젠테이션의 정지(pausing) 없이 플레이 아웃될 시간에서 완전히 수신될것 같지 않게 되는 너무 낮은 범위에 속한다. 수신기는 사전에 이러한 상황들을 검출할 수 있다. 예를 들어, 수신기는 6 유닛들의 시간 마다 5 유닛들의 미디어를 인코딩하는 블록들을 수신하고, 4 유닛들의 미디어의 버퍼를 갖는다는 것을 결정할 수 있어, 수신기는 이후 약 24 유닛들의 시간에, 프리젠테이션의 정지(stall) 또는 일시중지(pause)을 예상할 수 있다. 충분한 통

지(notice)로, 수신기는 예를 들어, 플레이아웃 시간의 유닛당 더 적은 대역폭을 사용하는 것과 같은, 컨텐트의 서로 다른 리프리젠테이션으로부터 블록 또는 블록들을 요청하는 시작 및 블록들의 현재 스트림을 포기함으로써 이러한 상황에 반응할 수 있다. 예를 들어, 동일한 크기 블록들에 대한 적어도 20% 보다 많은 비디오 시간 동안 블록들이 인코딩되는 리프리젠테이션으로 수신기가 스위치된 경우, 수신기는 대역폭 상황이 개선될 때까지 중지되는 것에 대한 필요성을 소거시킬 수 있다.

[0395] 그러나, 수신기가 포기된 리프리젠테이션으로부터 이미 수신된 데이터를 완전히 폐기하는 것은 낭비적일 수 있다. 본 명세서에 기술된 블록-스트리밍 시스템의 실시예에서, 각각의 블록 내의 데이터는, 블록 내의 데이터의 특정 프리픽스들이 수신되었던 나머지 블록 없이, 프리젠테이션을 지속하기 위해 이용될 수 있는 방식으로 인코딩 및 정렬될 수 있다. 예를 들어, 확장가능 비디오 인코딩의 공지된 기술들이 사용될 수 있다. 이러한 비디오 인코딩 방법들의 예들은 H.264 확장가능 비디오 코딩(SVC) 또는 H.264 개선된 비디오 코딩(AVC)의 시간 확장성(temporal scalability)을 포함한다. 유리하게, 이 방법은 예를 들어 이용가능한 대역폭에서의 변화들로 인해, 블록 또는 블록들의 수신이 포기될 수 있을 때에도 수신되었던 블록의 일부에 기초하여 프리젠테이션을 지속적으로 허용한다. 다른 장점은 컨텐트의 다수의 서로 다른 리프리젠테이션들에 대한 소스로서 싱글 데이터 파일이 이용될 수 있다는 것이다. 이는 예를 들어, 요구되는 리프리젠테이션에 해당하는 블록의 서브세트를 선택하는 HTTP 부분 GET 요청들의 사용을 구성함으로써 달성된다.

[0396] 본 명세서에 기술된 일 개선안은 강화된 세그먼트, 확장가능 세그먼트 맵이다. 확장가능 세그먼트 맵은 클라이언트가 그에 따라 세그먼트들의 일부들을 액세스하고 계층들을 추출할 수 있도록 세그먼트 내의 서로 다른 계층들의 로케이션들을 포함한다. 다른 실시예에서, 세그먼트 내의 미디어 데이터는 세그먼트의 품질이 세그먼트의 시작으로부터 점차적으로 데이터를 다운로딩하는 동안 증가되도록 오더링된다. 다른 실시예에서, 품질의 점차적 증가는 세그먼트내에 포함된 각각의 블록 또는 프래그먼트에 대해 적용되며, 프래그먼트 요청들은 확장가능 해결책을 해결하기 위해 수행될 수 있다.

[0397] 도 12는 확장가능 블록들의 양상을 도시하는 도면이다. 이 도면에서, 전송기(1200)는 메타데이터(1202), 확장가능 계층 1(1204), 확장가능 계층 2(1206), 및 확장가능 계층 3(1208)을 출력하며, 이후 지연된다. 다음, 수신기(1210)는 미디어 프리젠테이션(1212)을 제시하기 위해 메타데이터(1202), 확장가능 계층 1(1204), 확장가능 계층 2(1206)를 이용할 수 있다.

[0398] 독립적인 확장성 계층들

[0399] 위에서 설명된 바와 같이, 수신기가 미디어 데이터의 특정 리프리젠테이션의 요청된 블록들을 그것의 플레이아웃을 위한 시간에 수신할 수 없을 때 블록-요청 스트리밍 시스템을 스톨링시켜야 하는 것은 바람직하지 않은데, 그 이유는 그것은 종종 나쁜 사용자 경험을 발생시키기 때문이다. 프리젠테이션의 임의의 정해진 부분이 적시에 수신되지 않을 가능성이 거의 없게 되도록 하기 위해서, 선택되는 리프리젠테이션들의 데이터 레이트를 이용가능한 대역폭보다 훨씬 작게 되도록 제약함으로써 스톨들이 회피, 감소 또는 완화될 수 있지만, 이러한 전략은 이용가능한 대역폭에 의해 원칙적으로 지원될 수 있는 것보다 미디어 품질이 반드시 훨씬 낮다는 단점을 갖는다. 가능한 것보다 더 낮은 품질의 프리젠테이션은 나쁜 사용자 경험으로도 해석될 수 있다. 따라서, 블록-요청 스트리밍 시스템의 설계자는 이용가능한 대역폭보다 훨씬 낮은 데이터 레이트를 갖는 컨텐트 버전을 요청하거나(이 경우에는, 사용자는 나쁜 미디어 품질을 경험할 수 있음) 또는 이용가능한 대역폭에 근접한 데이터 레이트를 갖는 컨텐트 버전을 요청하기 위해(이 경우에는, 사용자는 이용가능한 대역폭이 변하기 때문에 프리젠테이션 동안 높은 중지 확률을 경험할 수 있음) 클라이언트 절차의 설계, 클라이언트의 프로그래밍 또는 하드웨어의 구성에 있어 선택에 직면하게 된다.

[0400] 이러한 상황들을 처리하기 위해서, 여기서 설명되는 블록-스트리밍 시스템들은 다수의 확장성 계층을 독립적으로 처리하도록 구성될 수 있고, 그럼으로써 수신기는 계층화된 요청들을 수행할 수 있고 전송기는 계층화된 요청들에 응답할 수 있다.

[0401] 이러한 실시예들에서, 각각의 블록에 대한 인코딩된 미디어 데이터는 여기서 "계층들"로서 지칭되는 다수의 해제된 피스들로 분할될 수 있고, 그럼으로써 계층들의 조합이 블록에 대한 미디어 데이터 전체를 포함하고, 또한 그럼으로써 계층들의 특정 서브세트들을 수신한 클라이언트가 컨텐트의 리프리젠테이션의 디코딩 및 프리젠테이션을 수행할 수 있다. 이러한 해결책에 있어서, 스트림에서 데이터의 정렬은 연속적인 영역들이 품질에 있어 증가하고 있고 메타데이터가 이를 반영하도록 이루어진다.

[0402] 위의 특성을 갖는 계층들을 생성하기 위해 사용될 수 있는 기술의 예로는 예컨대 ITU-T Standards H.264/SVC에 설명된 바와 같은 Scalable Video Coding의 기술이 있다. 위의 특성을 갖는 계층들을 생성하기 위해 사용될 수 있는 기술의 다른 예로는 ITU-T Standard H.264/AVC에 제공된 바와 같은 임시 확장성 계층들의 기술이 있다.

[0403] 이러한 실시예들에 있어서, 메타데이터는 MPD 또는 세그먼트 자체로 제공될 수 있어서, 임의의 정해진 블록의 개별적인 계층들 및/또는 계층들의 조합들 및/또는 다수의 블록들의 정해진 계층 및/또는 다수의 블록들의 계층들의 조합에 대한 요청들의 구성을 가능하게 한다. 예컨대, 블록을 포함하는 계층들은 단일 파일 내에 저장될 수 있고, 메타데이터는 개별적인 계층들에 상응하는 파일 내의 바이트 범위들을 명시하도록 제공될 수 있다.

[0404] 바이트 범위들을 명시할 수 있는 파일 다운로드 프로토콜, 예컨대 HTTP 1.1은 개별적인 계층들 또는 다수의 계층들을 요청하는데 사용될 수 있다. 게다가, 본 발명을 검토함으로써 당업자에게 자명하게 될 바와 같이, 가변적인 크기를 갖는 블록들 및 블록들의 가변적인 조합들의 구성, 요청 및 다운로드에 관련하여 위에서 설명된 기술들은 이러한 상황에서도 잘 적용될 수 있다.

[0405] 조합들

[0406] 위에서 설명된 바와 같이 계층들로 분할되는 미디어 데이터의 사용에 의해 기존 기술들에 비교해서 서빙 인프라구조 용량 요건들의 감소 및/또는 사용자 경험의 향상을 달성하기 위해, 블록-요청 스트리밍 클라이언트에 의해 유리하게 이용될 수 있는 다수의 실시예들이 이제 설명된다.

[0407] 제 1 실시예에 있어서, 블록 요청 스트리밍 시스템의 공지된 기술들에는 컨텐트의 상이한 버전들이 일부 경우들에서 계층들의 상이한 조합들에 의해 대체되는 변경이 적용될 수 있다. 바꿔 말하면, 기존 시스템이 컨텐트의 2개의 별개 리프리젠테이션을 제공할 수 있는 경우에는, 여기서 설명된 개선된 시스템이 2개의 계층들을 제공할 수 있고, 기존 시스템에서 컨텐트의 한 리프리젠테이션이 개선된 시스템에서 제 1 계층에 대해 비트-레이트, 품질 및 어쩌면 다른 메트릭들에 있어 유사한 경우에는, 기존 시스템에서 컨텐트의 제 2 리프리젠테이션이 개선된 시스템에서 2개의 계층들의 조합에 대해 비트-레이트, 품질 및 어쩌면 메트릭들에 있어 유사하다. 그 결과, 개선된 시스템 내에서 요구되는 저장 용량이 기존 시스템에서 요구되는 것과 비교해서 감소된다. 게다가, 기존 시스템의 클라이언트가 하나의 리프리젠테이션 또는 다른 리프리젠테이션의 블록들에 대한 요청을 발행할 수 있는데 반해, 개선된 시스템의 클라이언트들은 블록의 제 1 계층 또는 두 계층들 모두 중 어느 하나에 대한 요청들을 발행할 수 있다. 그 결과, 두 시스템들에서의 사용자 경험은 유사하다. 게다가, 심지어 상이한 품질들의 경우에도 더 높은 가능성을 가지고 캐싱되는 공통 세그먼트들이 사용되기 때문에 향상된 캐싱이 제공된다.

[0408] 제 2 실시예에 있어서, 이제 설명되는 계층들의 방법을 이용하는 개선된 블록-요청 스트리밍 시스템에서 클라이언트는 미디어 인코딩의 수 개의 계층들 각각을 위한 별도의 데이터 버퍼를 유지할 수 있다. 클라이언트 디바이스들 내에서의 데이터 관리에 대해 당업자들에게 자명할 바와 같이, 이러한 "별도"의 버퍼들은 별도의 버퍼들에 대한 물리적으로 또는 논리적으로 별도의 메모리 영역들의 할당에 의해서, 또는 버퍼링된 데이터가 단일 또는 다수의 메모리 영역들에 저장되고 상이한 계층들로부터의 데이터의 분리가 별도의 계층들로부터의 데이터의 저장 위치들에 대한 참조들을 포함하는 데이터 구조들의 사용을 통해 논리적으로 달성되는 다른 기술들에 의해 구현될 수 있고, 그럼으로써 아래에서 "별도의 버퍼들"이란 용어는 별개 계층들의 데이터가 개별적으로 식별되는 임의의 방법을 포함하는 것으로 이해되어야 한다. 클라이언트는 예컨대 각각의 버퍼의 점유에 기초하여 각 블록의 개별적인 계층에 대한 요청을 발생하고, 그 계층들은 우선순위 순서에 따라 정렬됨으로써 만약 우선순위 순서에서 하위 계층에 대한 임의의 버퍼의 점유가 그 하위 계층에 대한 임계치 미만이라면 한 계층으로부터의 데이터에 대한 요청이 발행될 수 없다. 이러한 방법에 있어서, 만약 이용가능한 대역폭이 우선순위 순서에서 상위 계층들을 또한 수신하기 위해 요구되는 것보다 아래로 떨어진다면 그때는 단지 하부 계층만이 요청되도록 하기 위해서, 우선순위 순서에서 하위 계층들로부터의 데이터를 수신하도록 우선순위가 제공된다. 게다가, 상이한 계층들과 연관된 임계치들이 상이할 수 있고, 그럼으로써 예컨대 하위 계층들이 더 높은 임계치들을 갖는다. 상위 계층에 대한 데이터가 블록의 플레이아웃 시간 이전에는 수신될 수 없도록 하기 위해 이용가능한 대역폭이 변하는 경우, 하위 계층들에 대한 데이터는 반드시 이미 수신되었을 것이고, 따라서 프리젠테이션이 하위 계층들에서만 단지 계속될 수 있다. 버퍼 점유에 대한 임계치들은 데이터의 바이트들, 버퍼에 보유되는 데이터의 플레이아웃 지속시간, 블록들의 수 또는 임의의 다른 적절한 척도

를 통해 정의될 수 있다.

[0409] 제 3 실시예에 있어서, 제 1 및 제 2 실시예들의 방법들은, (제 1 실시예에서 처럼) 계층들의 서브세트를 각각 포함하는 다수의 미디어 프리젠테이션들이 제공되게 하고 또한 제 2 실시예가 리프리젠테이션 내의 계층들의 서브세트에 적용되게 하기 위해서, 조합될 수 있다.

[0410] 제 4 실시예에 있어서, 제 1, 제 2 및/또는 제 3 실시예들의 방법들은 컨텐트의 다수의 독립적인 리프리젠테이션들이 제공되는 실시예들과 조합될 수 있음으로써, 예컨대, 독립적인 리프리젠테이션들 중 적어도 하나가 제 1, 제 2 및/또는 제 3 실시예들의 기술들이 적용되는 다수의 계층들을 포함한다.

[0411] 개선된 버퍼 관리자

[0412] 버퍼 모니터(126)(도 2 참조)의 조합에 있어서, 개선된 버퍼 관리자가 클라이언트측 버퍼를 최적화시키기 위해 사용될 수 있다. 블록-요청 스트리밍 시스템들은 미디어 플레이아웃이 신속하게 시작하여 원활하게 계속될 수 있으면서 동시에 사용자 또는 목적지 디바이스에 최대 미디어 품질을 제공하는 것을 보장하길 원한다. 이는, 가장 높은 미디어 품질을 갖지만 또한 이후에 프리젠테이션에 있어 중지를 강제하지 않으면서 적시에 신속하게 시작되고 수신될 수 있는 블록들을 클라이언트가 요청하는 것을 필요로 할 수 있다.

[0413] 개선된 버퍼 관리자를 사용하는 실시예들에 있어서, 그 관리자는 어떤 미디어 데이터의 블록들을 요청하고 언제 그러한 요청들을 수행할지를 결정한다. 개선된 버퍼 관리자에는, 예컨대, 제공될 컨텐트에 대한 메타데이터의 세트가 제공될 수 있는데, 이러한 메타데이터는 각각의 리프리젠테이션을 위한 컨텐트 및 메타데이터를 위해 이용가능한 리프리젠테이션들의 리스트를 포함한다. 리프리젠테이션을 위한 메타데이터는 리프리젠테이션의 데이터 레이트 및 다른 파라미터들, 이를 테면 비디오, 오디오 또는 다른 코덱들 및 코덱 파라미터들, 비디오 해상도, 디코딩 복잡도, 오디오 언어, 클라이언트에서의 리프리젠테이션 선택에 영향을 줄 수 있는 임의의 다른 파라미터들에 대한 정보를 포함할 수 있다.

[0414] 리프리젠테이션을 위한 메타데이터는 또한 리프리젠테이션이 세그먼트화되어진 블록들에 대한 식별자들을 포함할 수 있는데, 이러한 식별자들은 클라이언트가 블록을 요청하기 위해 요구되는 정보를 제공한다. 예컨대, 요청 프로토콜이 HTTP인 경우, 식별자는 URL에 의해 식별되는 파일 내에서의 바이트 범위 또는 시간 스팬을 식별하는 추가 정보와 더불어 어쩌면 HTTP URL일 수 있는데, 이러한 바이트 범위 또는 시간 스팬은 URL에 의해 식별되는 파일 내의 특정 블록을 식별한다.

[0415] 특정 구현에 있어서, 개선된 버퍼 관리자는 수신기가 새로운 블록들에 대한 요청을 수행하는 때를 결정하고, 그 요청들을 전송하는 것으로 스스로 처리할 수 있다. 신규한 양상에 있어서, 개선된 버퍼 관리자는 너무 큰 대역폭을 사용하는 것과 스트리밍 플레이아웃 동안에 미디어를 다 써버리지는 것 간의 균형을 맞추는 균형 비율의 값에 따라 새로운 블록들에 대한 요청들을 수행한다.

[0416] 블록 버퍼(125)로부터 버퍼 모니터(126)에 의해 수신되는 정보는 미디어 데이터가 언제 수신되는지, 얼마나 많이 수신되었는지, 미디어 데이터의 플레이아웃이 언제 시작되거나 중단되는지, 및 미디어 플레이아웃의 속도와 같은 각각의 이벤트에 대한 표시들을 포함할 수 있다. 이러한 정보에 기초하여, 버퍼 모니터(126)는 현재 버퍼 크기 $B_{current}$를 나타내는 변수를 계산할 수 있다. 이러한 예들에 있어서, $B_{current}$는 클라이언트 또는 다른 디바이스 버퍼 또는 버퍼들에 보유되는 미디어의 양을 나타내고 시간 단위들로 측정될 수 있으며, 그로 인해서 $B_{current}$는 어떠한 추가적인 블록들 또는 부분 블록들도 수신되지 않는 경우에 버퍼 또는 버퍼들에 저장된 블록들 또는 부분 블록들에 의해 표현되는 미디어 모두를 플레이아웃하는데 걸리는 시간의 양을 나타낸다. 따라서, $B_{current}$는 클라이언트에서 이용가능하지만 아직 플레이되지 않은 미디어 데이터의 정규 플레이아웃 속도로의 "플레이아웃 지속시간"을 나타낸다.

[0417] 시간이 경과함에 따라, $B_{current}$의 값은 미디어가 플레이아웃될 때 감소할 것이며, 블록에 대한 새로운 데이터가 수신될 때마다 증가할 수 있다. 본 설명을 위해서, 블록의 전체 데이터가 블록 요청자(124)에서 이용가능할 때 그 블록이 수신된다는 것이 가정되지만, 예컨대 부분 블록들의 수신을 고려하기 위해서 다른 측정들이 대신 사용될 수 있다는 것을 주시하자. 실제로는, 블록의 수신이 시간 기간에 걸쳐 이루어질 수 있다.

[0418] 도 13은 미디어가 플레이아웃되고 블록들이 수신될 때 시간에 걸친 $B_{current}$의 값의 변화를 도시한다. 도 13에 도시된 바와 같이, $B_{current}$의 값은 t_o미만의 시간 동안에 제로이고, 이는 어떤 데이터도 수신되지 않았음을 나

타낸다. t_o에서는, 제 1 블록이 수신되고, $B_{current}$의 값이 수신된 블록의 플레이아웃 지속시간과 동일하도록 증가한다. 이때에, 플레이아웃은 아직 시작되지 않았고 따라서 $B_{current}$의 값이 시간 t_1까지 일정하게 유지되며, 그 시간 t_1에서는, 제 2 블록이 도달하고 $B_{current}$가 이러한 제 2 블록의 사이즈만큼 증가한다. 이때에는, 플레이아웃이 시작되고 $B_{current}$의 값이 시간 t_2까지 선형적으로 감소하기 시작하며, 그 시간 t_2에서는. 제 3 블록이 도달한다.

[0419] $B_{current}$의 진행은 이러한 "톱니모양" 방식으로 계속되며, 블록이 수신될 때마다(시간 t_2, t_3, t_4, t_5 및 t_6에) 계단식으로 증가하고, 데이터가 그 사이에 플레이아웃될 때 순조롭게 감소한다. 이러한 예에서는 플레이아웃이 컨텐트에 대한 정규 플레이아웃 레이트로 진행하고 따라서 블록 수신 사이의 곡선의 기울기가 정확히 -1이며, 이는 1초의 미디어 데이터가 경과하는 각 1초의 실시간 동안 플레이된다는 것을 의미함을 주시하자. 초당 정해진 수의 프레임들, 예컨대 초당 24-프레임들로 프레임-기반 미디어가 플레이아웃됨으로써, -1의 기울기는 각각의 개별적인 데이터 프레임의 플레이아웃을 나타내는 작은 단계 함수들에 의해, 예컨대 각각의 프레임이 플레이아웃될 때 1초의 -1/24의 단계들에 의해 근사화될 것이다.

[0420] 도 14는 시간에 걸쳐 $B_{current}$의 점진적 변화에 대한 다른 예를 나타낸다. 그 예에서, 제 1 블록은 t_0에 도달하고 플레이아웃이 즉시 시작된다. 블록 도달 및 플레이아웃은 시간 t_3까지 계속되고, 그 시간 t_3에서는 $B_{current}$의 값이 제로에 이른다. 그러한 경우가 발생할 때는, 어떠한 추가적인 미디어 데이터도 플레이아웃을 위해 이용가능하지 않고, 이는 미디어 프리젠테이션의 중지를 강제한다. 시간 t_4에서는, 제 4 블록이 수신되고 플레이아웃이 다시 시작될 수 있다. 이러한 예는 따라서 제 4 블록의 수신이 원하는 것보다 늦은 경우를 나타내고, 이는 플레이아웃의 중지 및 그로 인한 나쁜 사용자 경험을 초래한다. 따라서, 개선된 버퍼 관리자 및 다른 특징의 목적은 이러한 이벤트의 확률을 감소시키면서 동시에 높은 미디어 품질을 유지하는 것이다.

[0421] 버퍼 모니터(126)는 또한 시간 기간의 길이에 대한 주어진 시간 기간에서 수신된 미디어의 비율인 다른 메트릭($B_{ratio}(t)$)을 계산할 수 있다. 보다 구체적으로, $B_{ratio}(t)$는 $T_{received}/(T_{now}-t)$와 동일하고, 여기서 $T_{received}$는 대략 현재 시간(T_{now})까지 현재 시간보다 이른 t로부터 시간 기간에서 수신된 미디어의 양(자신의 플레이아웃(playout) 시간에 의해 측정됨)이다.

[0422] $B_{ratio}(t)$는 $B_{current}$의 변화율(rate of change)을 측정하기 위하여 사용될 수 있다. $B_{ratio}(t)=0$은 시간 t 이래 어떠한 데이터도 수신되지 않은 경우이고; $B_{current}$는, 미디어가 플레잉 아웃하는 것을 가정하여, 상기 시간 이래 ($T_{now}-t$)까지 감소되었을 것이다. $B_{ratio}(t)=1$은 시간($T_{now}-t$) 동안 미디어가 플레잉 아웃될 때와 동일한 양으로 미디어가 수신된 경우이고; $B_{current}$는 시간 t에서처럼 시간 T_{now}에서 동일한 값을 가질 것이다. $B_{ratio}(t)>1$은 시간($T_{now}-t$) 동안 플레이 아웃하기에 필요한 것보다 많은 데이터가 수신된 경우이고; $B_{current}$는 시간 t로부터 시간 T_{now}로 증가될 것이다.

[0423] 버퍼 모니터(126)는 정수의 값들을 취할 수 있는 값 State를 추가로 계산한다. 버퍼 모니터(126)는 $t<T_{now}$ 동안 $B_{current}$의 현재 값 및 B_{ratio}의 값들이 제공되면, 출력으로서 새로운 State 값을 제공하는 함수, NewState($B_{current}$, B_{ratio})가 추가로 장착된다. $B_{current}$ and B_{ratio}가 이 함수로 하여금 State의 현재 값과 상이한 값을 리턴하게 할때마다, 새로운 값은 State에 할당되고 이런 새로운 State 값은 블록 선택기(1213)에 표시된다.

[0424] 함수 NewState는 쌍($B_{current}$, $B_{ratio}(T_{now}-T_x)$)의 모든 가능한 값들의 공간을 참조하여 평가될 수 있거나 ? 여기서 T_x는 고정된(구성된) 값일 수 있음 ?, 예를 들어 $B_{current}$의 값들로부터 T_x의 값들로 맵핑하는 구성 테이블에 의해 $B_{current}$로부터 유도될 수 있거나, State의 이전 값에 의존할 수 있다. 버퍼 모니터(126)는 이 공간의 하나 또는 그 초과의 구획부들(patitionings)이 공급되고, 여기서 각각의 구획부는 해체된 영역들의 세트들을 포함하고, 각각의 영역은 State 값이 주석이 달린다. 그 다음 함수 NewState의 평가는 구획부를 식별하는 동작 및 쌍($B_{current}$, $B_{ratio}(T_{now}-T_x)$)이 속하는 영역을 결정하는 동작을 포함한다. 그 다음 리턴 값은 상기 영역과 연관된 주석이다. 간단한 경우에서, 단지 하나의 구획부가 제공된다. 보다 복잡한 경우, 구획부는 NewState 함수의 이전 시간의 평가시 쌍($B_{current}$, $B_{ratio}(T_{now}-T_x)$) 또는 다른 팩터들에 의존할 수 있다.

[0425] 특정 실시예에서, 상기된 구획부는 $B_{current}$에 대한 다수의 임계값들 및 다수의 B_{ratio}에 대한 임계값들을 포함하는 구성 테이블에 기초될 수 있다. 특히, $B_{current}$에 대한 임계값들이 $B_{thresh}(0)=0$, $B_{thresh}(1)$, ...,$B_{thresh}(n_1)$, $B_{thresh}(n_1+1)=\infty$라고 하자, 여기서 n_1은 $B_{current}$에 대한 비-제로 임계값들의 수이다. B_{ratio}에 대한 임계값들이 $B_{r-thresh}(0)=0$, $B_{r-thresh}(1)$, ..., $B_{r-thresh}(n_2)$, $B_{r-thresh}(n_2+1)=\infty$라고 하자, 여기서 n_2는 B_{ratio}에 대한 임계값들의 수이다. 이들 임계값들은 셀들의 $(n_1+1) \times (n_2+1)$ 그리드를 포함하는 구획부를 정의하고, j번째 로우의 i번째 셀은 $B_{thresh}(i-1) <= B_{current} < B_{thresh}(i)$ 및 $B_{r-thresh}(j-1) <= B_{ratio} < B_{r-thresh}(j)$인 영역에 상응한다. 상기된 그리드의 각각의 셀에는 메모리에 저장된 명시 값들과 연관됨으로써 같이 상태 값 주석이 달리고, 그 다음 함수 NewState는 값들 $B_{current}$ 및 $B_{ratio}(T_{now} - T_x)$에 의해 표시된 셀과 연관된 상태 값을 리턴한다.

[0426] 추가 실시예에서, 히스테리시스 값은 각각의 임계값에 연관될 수 있다. 이런 강화된 방법에서, 함수 NewState의 평가는 다음과 같이 일시적으로 변형된 임계값들의 세트를 사용하여 구성된 임시 구획부에 기초될 수 있다. NewState의 최종 평가를 위해 선택된 셀에 상응하는 $B_{current}$ 범위보다 작은 각각의 $B_{current}$ 임계값에 대해, 임계값은 상기 임계값과 연관된 히스테리시스 값을 감산함으로써 감소된다. NewState의 최종 평가를 위해 선택된 셀에 상응하는 $B_{current}$ 범위보다 큰 각각의 $B_{current}$ 임계값에 대해, 임계값은 상기 임계값과 연관된 히스테리스 값을 가산함으로써 증가된다. NewState의 최종 평가를 위해 선택된 셀에 상응하는 B_{ratio} 범위보다 작은 각각의 B_{ratio} 임계값에 대해, 임계값은 상기 임계값과 연관된 히스테리시스 값을 감산함으로써 감소된다. NewState의 최종 평가를 위해 선택된 셀에 상응하는 B_{ratio}보다 큰 각각의 B_{ratio} 임계값에 대해, 임계값은 상기 임계값과 연관된 히스테리시스 값을 가산함으로써 증가된다. 변형된 임계값들은 NewState의 값을 평가하기 위하여 사용되고 그 다음 임계값들은 자신의 본래 값들로 리턴된다.

[0427] 공간의 구획부들을 정의하는 다른 방식들은 본 명세서를 읽는 당업자에게 명백할 것이다. 예를 들어, 구획부는 전체 공간 내의 하프-공간들을 정의하기 위해 그리고 상기 다수의 하프-공간의 인터섹션(intersection)으로서 각각의 별개의 세트를 정의하여, B_{ratio} 및 $B_{current}$의 선형 조합들에 기초한 부등식들, 예를 들어 실제 값 $\alpha 0$, $\alpha 1$, 및 $\alpha 2$에 대해 형태 $\alpha 1?B_{ratio} + \alpha 2?B_{current} \leq \alpha 0$ 형태의 선형 부등식 임계값들의 사용에 의해 정의될 수 있다.

[0428] 상기 설명은 기본 프로세스를 도시한다. 본 개시를 읽는 실시간 프로그래밍의 당업자에게 명확할 바와 같이, 효과적인 구현들은 가능하다. 예를 들어, 새로운 정보가 버퍼 모니터(126)에 제공되는 각각의 시간에, 예를 들어 블록들에 대한 추가 데이터가 수신되지 않으면 NewState가 새로운 값으로 변이할 미래 시간을 계산하는 것은 가능하다. 그 다음 타이머는 이 시간으로 설정되고 추가 입력들의 부재시 이 타이머의 만료는 새로운 State 값으로 하여금 블록 선택기(123)로 전송되게 할 것이다. 결과적으로, 계산들은 연속적으로보다, 새로운 정보가 버퍼 모니터(126)에 제공될 때 또는 타이머가 만료할 때만 수행될 필요가 있다.

[0429] State의 적당한 값들은 "Low", "Stable" 및 "Full"일 수 있다. 임계값들의 적당한 세트의 예 및 결과적인 셀 그리드는 도 15에 도시된다.

[0430] 도 15에서, $B_{current}$ 임계값들은 "+/-값"으로서 아래에 도시된 히스테리시스 값들을 사용하여 수평 축 상에 밀리초들 단위로 도시된다. B_{ratio} 임계값들은 "+/-값"으로서 아래에 도시된 히스테리시스 값들을 사용하여 수직 축 상에 천분율(즉, 1000에 의해 곱셈됨)로 도시된다. State 값들은 각각 "Low", "Stable" 및 "Full"에 대해 "L", "S" 및 "F"로서 그리드 셀들에 주석이 달린다.

[0431] 블록 선택기(123)는 새로운 블록을 요청할 기회가 있을 때마다 블록 요청기(124)로부터 주석들을 수신한다. 상기된 바와 같이, 블록 선택기(123)에는 예를 들어 각각의 블록의 미디어 데이터 레이트에 관한 정보를 포함하는 이용 가능한 복수의 블록들 및 이들 블록들에 대한 메타데이터에 대한 정보가 제공된다.

[0432] 블록의 미디어 데이터 레이트에 관한 정보는 명시 블록의 실제 미디어 데이터 레이트(즉, 초들의 플레이아웃 시간에 의해 나누어진 바이트들의 블록 크기)를 일시 중지들 없이 블록이 속하는 표현을 플레이 아웃하기 위하여, 유지된 기초에 의해, 블록이 속하거나 이용 가능한 대역폭의 측정이 요구되는 표현의 평균 미디어 데이터 레이트, 또는 상기의 조합을 포함할 수 있다.

[0433] 블록 선택기(123)는 버퍼 모니터(126)에 의해 지시된 최종 State 값에 기초하여 블록들을 선택한다. 이 상태

값이 "Stable"일 때, 블록 선택기(123)는 이전 선택된 블록과 동일한 표현으로부터 블록을 선택한다. 선택된 블록은 어떠한 미디어 데이터도 이전에 요청되지 않은 프리젠테이션의 시간 기간 동안 미디어 데이터를 포함하는 제 1 블록(플레이아웃 순서에서)이다.

[0434] State 값이 "Low"일 때, 블록 선택기(123)는 이전에 선택된 블록의 것보다 낮은 미디어 데이터 레이트를 가진 표현으로부터 블록을 선택한다. 다수의 팩터들은 이 경우 표현의 정확한 선택에 영향을 미칠 수 있다. 예를 들어, 블록 선택기(123)는 인입 데이터의 어그리게이트(aggregate) 레이트의 표시가 제공될 수 있고 상기 값보다 작은 미디어 데이터 레이트를 가진 표현을 선택할 수 있다.

[0435] State 값이 "Full"일 때, 블록 선택기(123)는 이전에 선택된 블록의 것보다 높은 미디어 데이터 레이트를 가진 표현으로부터 블록을 선택한다. 다수의 팩터들은 이 경우에 표현의 정확한 선택에 영향을 미칠 수 있다. 예를 들어, 블록 선택기(123)는 인입 데이터의 어그리게이트 레이트의 표시가 제공될 수 있고 상기 값보다 많지 않은 미디어 데이터 레이트를 가진 표현을 선택할 수 있다.

[0436] 다수의 부가적인 팩터들은 블록 선택기(123)의 동작에 추가로 영향을 미칠 수 있다. 특히, 버퍼 모니터(126)가 "Full" 상태를 계속 표시할지라도, 선택된 블록의 미디어 데이터 레이트가 증가된 빈도는 제한될 수 있다. 게다가, 블록 선택기(123)가 "Full" 상태 표시를 수신하지만 이용 가능한 보다 높은 미디어 데이터 레이트의 블록들이 없다는 것(예를 들어 최종 선택된 블록이 이미 가장 높은 이용 가능한 미디어 데이터 레이트를 위한 것이기 때문에)은 가능하다. 이 경우, 블록 선택기(123)는, 블록 버퍼(125) 내에 버퍼된 미디어 데이터의 전체 양이 상한 제한되도록, 선택된 시간만큼 다음 블록의 선택을 지연할 수 있다.

[0437] 부가적인 팩터들은 선택 프로세스 동안 고려된 블록들의 세트에 영향을 미칠 수 있다. 예를 들어, 이용 가능한 블록들은 인코딩 분해능이 블록 선택기(123)에 제공된 명시 범위 내에 속하는 표현들로부터의 이용 가능한 블록들로 제한될 수 있다.

[0438] 블록 선택기(123)는 또한 미디어 디코딩 동안 계산 자원들의 이용 가능성 같은 시스템의 다른 양상들을 모니터하는 다른 컴포넌트들로부터의 입력들을 수신할 수 있다. 만약 상기 자원들이 부족하게 되면, 블록 선택기(123)는 디코딩이 메타데이터 내에서 보다 낮은 계산 복잡성을 가지도록 표시되는(예를 들어, 보다 낮은 분해능 또는 프레임 레이트를 가진 표현들은 일반적으로 보다 낮은 디코딩 복잡성을 가짐) 블록들을 선택할 수 있다.

[0439] 상기된 실시예는 버퍼 모니터(126) 내의 NewState 함수의 평가시 값 B_{ratio}의 이용이 단지 $B_{current}$만을 고려하는 방법과 비교하여 프리젠테이션의 시작시 품질의 보다 빠른 증가를 허용한다는 점에서 실질적인 장점을 유발한다. B_{ratio}를 고려하지 않고, 다량의 버퍼된 데이터는 시스템이 보다 높은 미디어 데이터 레이트 그리고 이에 따라 보다 높은 품질을 가진 블록들을 선택할 수 있기 전에 누적될 수 있다. 그러나, B_{ratio} 값이 클 때, 이것은 이용 가능한 대역폭이 이전에 수신된 블록들의 미디어 데이터 레이트보다 훨씬 높고 심지어 비교적 작은 버퍼된 데이터(즉, $B_{current}$에 대한 낮은 값)을 가지는 것을 가리키고, 보다 높은 미디어 데이터 레이트 및 이에 따른 보다 높은 품질의 블록들을 요청하는 것이 안전하다. 똑같이, 만약 B_{ratio} 값이 낮을 때(예를 들어, <1), 이것은 이용 가능한 대역폭이 이전에 요청된 블록들의 미디어 데이터 레이트 아래로 떨어졌고 따라서 비록 $B_{current}$가 높다는 것을 가리키고, 예를 들어 시스템이 $B_{current}$=0이고 미디어의 플레이아웃이 멈추는 포인트에 도달하는 것을 회피하기 위해, 시스템이 보다 낮은 미디어 데이터 레이트로 스위칭하고 이에 따라 보다 낮은 품질로 스위칭할 것이다. 이런 개선된 동작은 네트워크 조건들 및 따라서 전달 속도들이 빠르고 다이나믹하게 가변할 수 있는, 예를 들어 사용자들이 모바일 디바이스들로 스트리밍하는 환경들에서 특히 중요할 수 있다.

[0440] 다른 장점은 ($B_{current}$, B_{ratio})의 값들의 공간의 구획부를 지정하기 위하여 구성 데이터의 사용에 의해 수여된다. 상기 구성 데이터는 프리젠테이션 메타데이터의 일부로서 또는 다른 다이나믹 수단에 의해 버퍼 모니터(126)에 제공될 수 있다. 실제 전개들에서, 사용자 네트워크 접속들의 동작은 사용자들 사이 및 단일 사용자에 대한 시간에 걸쳐 크게 가변 가능할 수 있기 때문에, 모든 사용자들을 위해 잘 작동할 구획부들을 예측하는 것은 어려울 수 있다. 상기 구성 정보를 사용자들에게 제공할 가능성은 우수한 구성 세팅들이 누적된 경험에 따라 시간에 걸쳐 개발될 수 있게 한다.

[0441] 변수 요청 사이즈 설정

[0442] 높은 빈도의 요청들은 각각의 요청이 단일 블록을 위한 것이면 그리고 각각의 블록이 짧은 미디어 세그먼트를

인코딩하면 요구될 수 있다. 만약 미디어 블록들이 짧으면, 비디오 플레이아웃은 블록으로부터 블록으로 빠르게 이동하고, 이는 표현을 변화시킴으로써 수신기가 자신의 선택된 데이터 레이트를 조절하거나 변화시키기 위한 보다 빈번한 기회들을 제공하고, 플레이아웃이 멈추지 않고 계속될 수 있는 가능성을 개선시킨다. 그러나, 높은 빈도의 요청들에 대한 하강 부분은, 이용 가능한 대역폭이 3G 및 4G 무선 WAN들 같은 무선 WAN 네트워크들에서 서버 네트워크에 대한 클라이언트 내에 한정되는 특정 네트워크들 상에서 유지할 수 없다는 것이고, 여기서 클라이언트로부터 네트워크로 데이터 링크의 용량은 무선 조건들의 변화들로 인해 짧거나 긴 기간들 동안 제한되거나 제한될 수 있다.

[0443] 높은 빈도의 요청들은 또한 서빙 인프라구조 상에서의 높은 로드를 암시하고, 이는 용량 요건들의 측면에서 연관된 비용들을 초래한다. 따라서, 아무런 단점들 없이 높은 빈도의 요청들의 장점들 중 일부를 가지는 것은 바람직할 것이다.

[0444] 블록 스트리밍 시스템의 일부 실시예들에서, 높은 요청 빈도의 유연성은 보다 적은 빈도 요청들과 결합된다. 이 실시예에서, 블록들은 또한 상기된 바와 같이 구성될 수 있고 또한 상기된 바와 같이, 다수의 블록들을 포함하는 세그먼트들로 어그리게이트된다. 프리젠테이션의 시작시, 각각의 요청이 단일 블록을 참조하거나 다수의 동시 요청들이 블록의 파트들을 요청하게 되는 상기된 프로세스들은 빠른 채널 재핑(zapping) 시간 및 그러므로 프리젠테이션의 시작시 우수한 사용자 경험을 보장하기 위해 적용된다. 그후, 하기 설명될 명시 조건이 부합될 때, 클라이언트는 단일 요청 내에 다수의 블록들을 포함하는 요청들을 발행할 수 있다. 이것은 블록들이 보다 큰 파일들 또는 세그먼트들로 어그리게이트되고 바이트 또는 시간 범위들을 사용하여 요청될 수 있기 때문에 가능하다. 연속적인 바이트 또는 시간 범위들은 다수의 블록들에 대한 단일 요청을 유발하는 단일의 보다 큰 바이트 또는 시간 범위로 어그리게이트될 수 있고, 심지어 불연속적인 블록들은 하나의 요청시 요청될 수 있다.

[0445] 단일 블록(또는 부분 블록)을 요청할지 또는 다수의 연속적인 블록들을 요청할지를 결정함으로써 구동될 수 있는 하나의 기본적인 구성은 요청된 블록들이 플레이아웃될지 아닐지에 대한 결정의 구성 베이스를 가진다. 예를 들어, 곧 다른 표현으로 변화될 필요가 있을 것이 가능하다면, 클라이언트가 단일 블록들, 즉 작은 양들의 미디어 데이터에 대한 요청들을 하는 것이 더 좋다. 이것에 대한 하나의 이유는 다른 표현으로의 스위치가 임박될 수 있을 때 만약 다수의 블록들에 대한 요청이 이루어지면, 요청의 최종 몇몇 블록들이 플레이 아웃되기 전에 스위치가 이루어질 수 있다는 것이다. 따라서, 이들 최종 몇몇 블록들의 다운로드는 스위치가 이루어지는 표현의 미디어 데이터의 전달을 지연시킬 수 있고, 이는 미디어 플레이아웃 멈춤들을 유발할 수 있다.

[0446] 그러나, 단일 블록들에 대한 요청들은 보다 높은 빈도의 요청들을 유발한다. 다른 한편, 곧 다른 표현으로 변화할 필요가 없을 것 같으면, 이들 블록들 모두가 플레이 아웃 될 것 같을 때, 다수의 블록들에 대해 요청들을 하는 것이 바람직할 수 있고, 그리고 이는 특히 표현의 임박한 변화가 없다는 것이 통상적이면, 실질적으로 요청 오버헤드를 낮출 수 있는 보다 낮은 빈도의 요청들을 유발한다.

[0447] 통상적인 블록 어그리게이션 시스템들에서, 각각의 요청시 요청된 양은 다이나믹하게 조절되지 않는다, 즉 통상적으로 각각의 요청은 전체 파일에 대한 것이거나, 또는 각각의 요청은 대략 표현의 동일한 양의 파일(때때로 시간적으로 측정됨, 때때로 바이트들로 측정됨)에 대한 것이다. 따라서, 만약 모든 요청들이 보다 작으면, 요청 오버헤드는 높은 반면, 만약 모든 요청들이 보다 크면, 이것은 미디어 멈춤 이벤트들의 기회들을 증가시키고, 및/또는 네트워크 조건들이 가변할 때 보다 낮은 품질 표현들이 표현들을 빠르게 변화시켜야 하는 것을 회피하기 위하여 선택된다면 보다 낮은 품질의 미디어 플레이아웃을 제공한다.

[0448] 충족될 때 추후 요청들이 다수의 블록들을 참조하게 할 수 있는 조건의 예는 버퍼 사이즈 $B_{current}$에 대한 임계값이다. 만약 $B_{current}$가 임계값 아래라면, 발행된 각각의 요청은 단일 블록을 참조한다. 만약 $B_{current}$가 임계값보다 크거나 같으면, 그 다음 발행된 각각의 요청은 다수의 블록들을 참조한다. 만약 다수의 블록들을 참조하는 요청이 발행되면, 그 다음 각각의 단일 요청시 요청된 블록들의 수는 몇몇 가능한 방식들 중 하나의 방식으로 결정될 수 있다. 예를 들어, 상기 수는 일정할 수 있고, 예를 들어 2이다. 대안적으로, 단일 요청시 요청된 블록들의 수는 버퍼 상태 및 특히 $B_{current}$에 의존할 수 있다. 예를 들어, 다수의 임계값들은 설정될 수 있고, 단일 요청시 요청된 블록들의 수는 $B_{current}$보다 작은 다수의 임계값들 중 가장 높은 임계값으로부터 유도된다.

[0449] 충족될 때 요청들이 다수의 블록들을 참조하게 할 수 있는 조건의 다른 예는 상기된 값 State 변수이다. 예

를 들어, State가 "Stable" 또는 "Full"일 때, 요청들은 다수의 블록들에 대해 발행될 수 있지만, State가 "Low"일 때 모든 요청들은 하나의 블록에 대해 발행될 수 있다.

[0450] 다른 실시예는 도 16에 도시된다. 이 실시예에서, 다음 요청이 발행될 때(단계(1300)에서 결정됨), 현재 State 값 및 Bcurrent는 다음 요청의 사이즈를 결정하기 위해 사용된다. 만약 현재 State 값이 "Low"이거나 현재 State 값이 "Full"이고 현재 표현이 이용 가능한 가장 높은 것이 아니면(단계(1310)에서 결정되고, 대답은 "Yes"임), 다음 요청은 예를 들어 단지 다음 블록(단계(1320)에서 결정된 블록 및 이루어진 요청)에 대해서만 곧 선택된다. 이 배후의 이유는 이들이 아주 빨리 표현들의 변화가 있을 것 같은 조건들이기 때문이다. 만약 전류 State 값이 "Stable"이거나 현재 State 값이 "Full"이고 현재 표현이 이용 가능한 가장 높은 것이면(단계(1310)에서 결정되고, 대답은 "No"임), 다음 요청에서 요청된 연속적인 블록들의 지속시간은 몇몇 고정된 $\alpha<1$(단계(1330)에서 결정된 블록들, 단계(1340)에서 이루어진 요청)에 대한, 예를 들어 $B_{current}=5$이면 $\alpha=0.4$에 대한 $B_{current}$의 α-프랙션에 비례하도록 선택되고, 그 다음 다음 요청은 블록들의 대략 4 초 동안일 수 있다. 이것에 대한 하나의 이유는 이들 조건들에서 새로운 표현으로의 스위치가 $B_{current}$에 비례하는 시간의 양에 대해 이루어질 것 갖지 않을 수 있기 때문이다.

[0451] 유연한 파이프라이닝

[0452] 블록-스트리밍 시스템들은 명시 근원 전송 프로토콜, 예를 들어 TCP/IP를 가진 파일 요청 프로토콜을 사용할 수 있다. TCP/IP 또는 다른 전송 프로토콜 접속의 시작시, 완전 이용 가능한 대역폭의 사용을 달성하기 위해 몇몇 무시할 수 없는 시간이 걸릴 수 있다. 이것은 새로운 접속이 시작되는 매 시간 "접속 개시 페널티"를 유발한다. 예를 들어, TCP/IP의 경우에, 접속 개시 페널티는 접속을 설정하기 위하여 최초 TCP 핸드쉐이크를 위해 걸리는 시간 및 이용 가능한 대역폭의 전체 이용을 달성하기 위하여 혼잡 제어 프로토콜을 위해 걸린 시간 둘 다로 인해 발생한다.

[0453] 이 경우, 접속 개시 페널티가 초래되는 빈도를 감소시키기 위하여, 단일 접속을 사용하여 다수의 요청들을 발생하는 것이 바람직할 수 있다. 그러나, 몇몇 파일 전송 프로토콜들, 예를 들어 HTTP는 전송 계층 접속을 전부 폐쇄하는 것이 아닌 요청을 취소하기 위한 메카니즘을 제공하지 않아서, 새로운 접속이 이전 접속 대신 설정될 때 접속 개시 페널티를 초래한다. 발행된 요청은 이용 가능한 대역폭이 변경되었고 다른 매체 데이터 레이트가 대신 요구되는 것이 결정되면, 즉 상이한 표현으로 스위치할 결정이 존재하면 취소될 필요가 있을 수 있다. 발행된 요청을 취소하기 위한 다른 이유는 미디어 프리젠테이션이 종료되었고 새로운 프리젠테이션이 시작되는 것(아마도 프리젠테이션시 상이한 포인트에서 동일한 컨텐트 아이템 또는 아마도 새로운 컨텐트 아이템)을 사용자가 요청했기 때문일 수 있다.

[0454] 알려진 바와 같이, 접속 개시 페널티는 접속을 개방으로 유지하고 추후 요청들에 대해 동일한 접속을 재사용함으로써 회피될 수 있고 또한 알려진 바와 같이 상기 접속은 만약 다수의 요청들이 동일한 접속 상에서 동시에 발생되면(HTTP의 환경에서 "파이프라이닝"으로서 공지된 기술) 완전히 이용되게 유지될 수 있다. 그러나, 동시에, 또는 보다 일반적으로 이전 요청들이 접속을 통하여 완료되기 전에 다수의 요청이 발행되는 방식으로 다수의 요청들을 발생하는 것의 단점은 접속이 이들 요청들에 대한 응답을 운반하는 것에 전념되고 따라서 요청들이 발행되어야 하는 변경들이 바람직하면, 더 이상 요구되지 않는 이전에 발행된 요청들을 취소하는 것이 필요하게 되는 경우 접속이 폐쇄될 수 있다는 것일 수 있다.

[0455] 발행된 요청이 취소될 필요가 있는 확률은 요청의 발행과 요청된 블록의 플레이아웃 시간 사이의 시간 인터벌이 높을 때 발행된 요청이 취소될 필요가 있는 확률이 또한 높다(이용 가능한 대역폭이 상기 인터벌 동안 변화할 것이기 때문에)는 점에서 상기 요청의 발행과 상기 요청된 블록의 플레이아웃 시간 사이의 시간 인터벌의 지속시간에 부분적으로 의존할 수 있다.

[0456] 알려진 바와 같이, 몇몇 파일 다운로드 프로토콜들은 단일 근원 전송 계층 접속이 유리하게 다수의 다운로드 요청들을 위해 사용될 수 있는 특성을 가진다. 예를 들어, HTTP는 이런 특성을 가지는데, 그 이유는 다수의 요청들을 위해 단일의 접속의 재사용이 첫 번째와 상이한 요청들에 대해 상기된 "접속 개시 페널티"를 회피하기 때문이다. 그러나, 이런 해결책의 단점은 접속이 각각의 발행된 요청시 요청된 데이터를 전송하는 것에 전념되고 그러므로 만약 요청 또는 요청들이 취소될 필요가 있다면, 접속이 폐쇄될 수 있어서, 대체 접속이 설정될 때 접속 개시 페널티를 초과하거나, 클라이언트가 더 이상 필요하지 않은 데이터를 수신하기 위해 기다릴 수 있어서, 추후 데이터의 수신시 지연을 초래하는 것이다.

[0457] 우리는 지금 이런 단점을 초래함이 없이 접속 재사용의 장점들을 유지하고 또한 부가적으로 접속이 재사용될

수 있는 주파수를 개선하는 실시예를 설명한다.

[0458] 여기에 설명된 블록-스트리밍 시스템들의 실시예들은 시작시 접속을 요청들의 명시 세트에 전념시킴이 없이 다수의 요청들에 대해 접속을 재사용하도록 구성된다. 필연적으로, 새로운 요청은 접속 중 이미 발행된 요청들이 아직 완료되지 않았지만 완료에 근접할 때, 기존 접속에 대해 발행된다. 기존 요청들이 완료될 때까지 기다리지 않는 하나의 이유는 만약 이전 요청들이 완료되면, 접속 속도가 저하될 수 있고, 즉 근원 TCP 세션이 유휴 상태로 진행할 수 있거나, TCP cwnd 변수가 실질적으로 감소될 수 있어서, 그 접속 중 발행된 새로운 요청의 최초 다운로드 속도를 실질적으로 감소시키기 때문이다. 부가적인 요청을 발행하기 전 완료에 근접할 때까지 기다리는 하나의 이유는 이전 요청들이 완료되기 전 새로운 요청이 오래 발행되면, 새로운 발행된 요청이 일부 상당한 시간 기간 동안 심지어 재시될 수 없고, 새로운 발행된 요청이 개시되기 전 이런 시간 기간 동안 예를 들어 표현들을 스위치할 결정으로 인해 새로운 요청을 형성할 결정이 더 이상 유효하지 않기 때문이다. 따라서, 이 기술을 구현하는 클라이언트들의 실시예는 접속의 다운로드 능력들을 감소시킴이 없이 가능한 한 늦게 접속 중에 새로운 요청을 발행할 것이다.

[0459] 방법은 이런 접속 중에 발행된 최근의 요청에 응답하여 접속중에 수신된 바이트들의 수를 모니터링하는 단계 및 이런 수로 테스트를 적용하는 단계를 포함한다. 이것은 수신기(또는 전송기, 만약 적용 가능하면)가 모니터 및 테스트하도록 구성됨으로써 행해질 수 있다.

[0460] 테스트가 통과되면, 추가 요청은 접속 중에 발행될 수 있다. 적당한 테스트의 하나의 예는 수신된 바이트들의 수가 요청된 데이터의 사이즈의 고정된 프랙션보다 큰지이다. 예를 들어, 이 프랙션은 80%일 수 있다. 적당한 테스트의 다른 예는 도 17에 도시된 바와 같이, 다음 계산에 기초한다. 계산시, R이 접속의 데이터 레이트의 추정치이고, T가 라운드 트립 시간("RTT")의 추정치이고 그리고 X가 예를 들어 0.5 내지 2 사이의 값으로 설정된 상수일 수 있는 분수 팩터라고 하자, 여기서 R 및 T의 추정치들은 정기적으로 업데이트된다(단계(1410)에서 업데이트됨). S가 최근 요청시 요청된 데이터의 사이즈이고, B가 수신된 요청된 데이터의 바이트들의 수라고 하자(단계(1420)에서 계산됨).

[0461] 적당한 테스트는 수신기(또는 전송기, 만약 적용 가능하면)가 부등식 (S-B)<X?R?T를 평가하기 위해 루틴을 실행하고(단계(1430)에서 테스트됨), 그리고 만약 "Yes"이면, 동작을 취하여야 할 것이다. 예를 들어, 접속 중 발행될 준비가 된 다른 요청이 있는지(단계(1440)에서 테스트됨), 그리고 만약 "Yes"이면 접속에 대한 그 요청을 발행하고(단계(1450)) 그리고 만약 "No"이면 프로세스가 단계(1410)로 되돌아가 계속 업데이팅 및 테스팅을 할지를 알기 위해 테스트가 이루어질 수 있다. 단계(1430)에서 테스트의 결과가 "No"이면, 프로세스는 업데이팅 및 테스팅을 계속하도록 단계(1410)로 되돌아 간다.

[0462] 단계(1430)에서의 부등식 테스트(예를 들어, 적당한 프로그램된 엘리먼트들에 의해 수행됨)는, 수신될 나머지 데이터의 양이 하나의 RTT 내에서 현재 추정된 수신 레이트에서 수신될 수 있는 데이터의 양보다 X 배와 동일할 때 각각의 추후 요청이 발생되게 한다. 단계(1410)에서 데이터 레이트 R을 추정할 다수의 방법들은 종래에 공지된다. 예를 들어, 데이터 레이트는 Dt/t로서 추정될 수 있고, 여기서 Dt는 이전 t 초들 내에서 수신된 비트들의 개수이고 여기서 t는 예를 들어 1초 또는 0.5초 또는 몇몇 다른 인터벌일 수 있다. 다른 방법은 인입 데이터 레이트의 지수 웨이티드 평균, 또는 1차 무한 임펄스 응답(IIR) 필터이다. 단계(1410)에서 RTT, T를 추정할 다수의 방법들은 종래에 공지되었다.

[0463] 단계(1430)에서의 테스트는 하기에 상세히 설명된 바와 같이, 인터페이스 상에서 모든 액티브 접속들의 어그리게이트(aggregate)에 적용될 수 있다.

[0464] 상기 방법은 후보 요청들의 리스트를 구성하는 단계, 각각의 후보 요청을 요청이 이루어질 적당한 서버들의 세트와 연관시키는 단계 및 우선순위의 순서로 후보 요청들의 리스트를 순서화하는 단계를 더 포함한다. 후보 요청들의 리스트 내의 일부 엔트리들은 동일한 우선순위를 가질 수 있다. 각각의 후보 요청과 연관된 적당한 서버들의 리스트 내 서버들은 호스트네임들에 의해 식별된다. 각각의 호스트네임은 잘 공지된 바와 같이 도메인 네임 시스템(Domain Name System)으로부터 획득될 수 있는 인터넷 프로토콜(Internet Protocol) 어드레스들의 세트에 상응한다. 그러므로 후보 요청들의 리스트에 대한 각각의 가능한 요청은 구체적으로 후보 요청과 연관된 서버들과 연관된 호스트네임들과 연관된 인터넷 프로토콜 어드레스들의 세트들의 연합인 인터넷 프로토콜 어드레스들의 세트와 연관된다. 단계(1430) 내에서 설명된 세트가 접속을 위해 충족되고, 그리고 새로운 요청이 그 접속에 대해 아직 발행되지 않을 때마다, 접속의 목적지의 인터넷 프로토콜 어드레스가 연관되는 후보 요청들의 리스트들 상에서 가장 높은 우선순위 요청은 선택되고, 그리고 이 요청은 접속에 대해 발행된다. 상기 요청은 또한 후보 요청들의 리스트로부터 제거된다.

[0465] 후보 요청들은 후보 요청들의 리스트로부터 제거(취소)될 수 있고, 새로운 요청들은 후보 리스트 상에서 이미 기존 요청들보다 높은 우선순위를 가진 후보 리스트에 부가될 수 있고, 그리고 후보 리스트 상의 기존 요청들은 변경된 자신의 우선순위를 가질 수 있다. 어느 요청들이 후보 요청들의 리스트 상에 있는지의 다이나믹 성질, 및 후보 리스트 상에서 자신의 우선순위의 다이나믹 성질은 단계(1430)에서 설명된 타입의 테스트가 만족되는 시기에 의존하여 어느 요청들이 다음에 발행될 수 있는지를 변경할 수 있다.

[0466] 예를 들어, 단계(1430)에서 설명된 테스트에 대한 대답이 언젠가 t에서 "Yes"이면, 발행된 다음 요청이 요청 A일 것인 반면, 단계(1430)에서 설명된 세트에 대한 대답이 언젠가 t'>t일 때까지 "Yes"가 아니면, 발행된 다음 요청이 대신 요청 B일 것이 가능할 수 있는데, 그 이유는 요청 A가 시간 t 및 t' 사이에서 후보 요청들의 리스트로부터 제거되었기 때문에, 또는 요청 B가 시간 t 및 t' 사이에서 요청 A보다 높은 우선순위를 가진 후보 요청들의 리스트에 부가되었기 때문에, 또는 요청 B가 시간 t에서 후보 리스트 상에 있지만 요청 A보다 낮은 우선순위를 가지며, t 및 t' 사이에서 요청 B의 우선순위가 요청 A의 우선순위보다 높게 이루어지기 때문이다.

[0467] 도 18은 요청들의 후보 리스트 상에서 요청들의 리스트의 예를 도시한다. 이 예에서, 3개의 접속들이 있고, 그리고 A, B, C, D, E 및 F로 라벨링된 후보 리스트 상에 6개의 요청들이 존재한다. 후보 리스트 상의 요청들의 각각은 표시된 바와 같이 접속들의 서브세트 상에서 발행될 수 있고, 예를 들어 요청 A는 접속 1 중에 발행될 수 있는 반면, 요청 F는 접속 2 또는 접속 3 상에서 발행될 수 있다. 각각의 요청의 우선순위는 도 18에서 또한 라벨링되고, 보다 낮은 우선순위 값은 요청이 보다 높은 우선순위인 것을 가리킨다. 따라서, 우선순위 0을 가진 요청들 A 및 B는 가장 높은 우선순위 요청들인 반면, 3의 우선순위 값을 가진 요청 F는 후보 리스트 상의 요청들 중 가장 낮은 우선순위이다.

[0468] 이런 시점 t에서, 접속 1이 단계(1430)에서 설명된 테스트를 통과하면, 요청 A 또는 요청 B가 접속 1 중에 발행된다. 대신 접속 3이 이런 시점 t에서 단계(1430)에서 설명된 세트를 통과하면, 요청 D는 접속 3 상에서 발행되는데, 그 이유는 요청 D가 접속 3 상에서 발행될 수 있는 가장 높은 우선순위를 가진 요청이기 때문이다.

[0469] 모든 접속들에 대해 시간 t로부터 약간 보다 늦은 시간 t'으로 단계(1430)에서 설명된 테스트에 대한 대답이 "No"이고, 시간 t 및 t' 사이에서 요청 A가 자신의 우선순위를 0으로부터 5로 변경하면, 요청 B는 후보 리스트로부터 제거되고, 그리고 우선순위 0을 가진 새로운 요청 G가 후보 리스트에 부가된다. 그 다음, 시간 t'에서, 새로운 후보 리스트는 도 19에 도시된 바와 같을 수 있다.

[0470] 시간 t' 접속 1이 단계(1430)에서 설명된 테스트를 통과하면, 우선순위 4를 가진 요청 C는 접속 1 중에 발행되는데, 그 이유는 상기 요청 C가 이 시점에서 접속 1 중에 발행될 수 있는 후보 리스트 상에서 가장 높은 우선순위 요청이기 때문이다.

[0471] 이런 동일한 상황에서 대신 요청 A가 시간 t에서 접속 1 중에 발행되는 것을 가정하자(이는 도 18에서 도시된 바와 같이 시간 t에서 접속 1에 대해 2개의 가장 높은 우선순위 선택들 중 하나였음). 시간 t로부터 약간 보다 늦은 시간 t'으로 단계(1430)에서 설명된 테스트에 대한 대답이 모든 접속들에 대해 "No"이기 대문에, 접속 1은 시간 t 이전에 발행된 요청들에 대해 적어도 시간 t까지 여전히 데이터를 전달하고, 따라서 요청 A는 적어도 시간 t'까지 개시되지 않았을 것이다. 시간 t'에서 요청 C를 발행하는 것은 시간 t에서 요청 A를 발행하는 것보다 우수한 결정인데, 그 이유는 요청 C는 요청 A가 시작되었을 시간 t' 이후 동일한 시간에 시작되고, 그리고 상기 시간까지 요청 C가 요청 A보다 높은 우선순위를 가지기 때문이다.

[0472] 다른 대안으로서, 단계(1430)에서 설명된 타입의 테스트가 액티브 접속들의 어그리게이트에 적용되면, 내부 프로토콜 어드레스가 후보 요청들의 리스트 상의 제 1 요청 또는 상기 제 1 요청과 동일한 우선순위를 가진 다른 요청과 연관된 목적지를 가진 접속은 선택될 수 있다.

[0473] 다수의 방법들은 후보 요청들의 리스트의 구성을 위해 가능하다. 예를 들어, 후보 리스트는 시간 시퀀스 순서로 프리젠테이션의 현재 표현의 데이터의 다음 n개의 부분들에 대한 요청들을 표현하는 n개의 요청들을 포함할 수 있고, 여기서 데이터의 가장 빠른 부분에 대한 요청은 가장 높은 우선순위를 가지며 데이터의 가장 늦은 부분에 대한 요청은 가장 낮은 우선순위를 가진다. 일부 경우들에서 n은 1일 수 있다. n의 값은 버퍼 크기 $B_{current}$에 의존하거나, 클라이언트 버퍼 점유 상태의 다른 측정 또는 State 변수에 의존할 수 있다. 예를 들어, 다수의 임계값들은 $B_{current}$에 대해 설정될 수 있고 각각의 임계값과 연관된 값 및 그 다음 n의 값은 $B_{current}$보다 작은 가장 높은 임계값과 연관된 값이도록 취해진다.

[0474] 상기된 실시예들은 접속들에 대한 요청들의 유연한 할당을 보장하여, 비록 가장 높은 우선순위 요청이 상기 접속에 적당하지 않더라도(접속의 목적지 IP 어드레스가 요청과 연관된 임의의 호스트네임들에 할당되는 것이 아니기 때문에) 우선권은 기존 접속을 재사용하는 것에 주어지는 것을 보장한다. $B_{current}$ 또는 State 또는 클라이언트 버퍼 점유의 다른 측정에 대한 n의 의존성은 클라이언트가 시퀀스 내에서 플레이 아웃될 데이터의 다음 부분과 연관된 요청의 발행 및 완료의 긴급 필요성 내에 있을 때 그런 "우선순위 순서 외" 요청들이 발행되지 않는 것을 보장한다.

[0475] 이들 방법들은 협력 HTTP 및 FEC와 유리하게 조합될 수 있다.

[0476] 일관된 서버 선택

[0477] 잘 알려진 바와 같이, 파일 다운로드 프로토콜을 사용하여 다운로드될 파일들은 일반적으로 호스트네임 및 파일네임을 포함하는 식별자에 의해 식별된다. 예를 들어 이것은 HTTP 프로토콜에 대한 경우이고 상기 경우 식별자는 URI(Uniform Resource Identifier)이다. 호스트네임은 인터넷 프로토콜 어드레스들에 의해 식별되는 다수의 호스트들에 상응할 수 있다. 예를 들어 이것은 다수의 물리적 머신들에 걸쳐 다수의 클라이언트들로부터의 요청들의 로드를 확산시키는 일반적인 방법이다. 특히 이런 해결책은 CDN들(Content Delivery Networks)에 의해 일반적으로 취해진다. 이 경우 임의의 물리적 호스트들에 대한 접속 중 발행된 요청은 성공할 것으로 예상된다. 다수의 방법들은 호스트네임과 연관된 인터넷 프로토콜 어드레스들 중에서 선택할 수 있다. 예를 들어, 이들 어드레스들은 도메인 네임 시스템(Domain Name System)을 통하여 클라이언트에 통상적으로 제공되고 우선순위 순서로 제공된다. 그 다음 클라이언트는 가장 높은 우선순위(제 1) 인터넷 프로토콜 어드레스를 선택할 수 있다. 그러나, 일반적으로 이런 선택이 어떻게 이루어지는가에 대해 클라이언트들 사이의 협력이 존재하지 않고, 그 결과 상이한 클라이언트들은 상이한 서버들로부터 동일한 파일을 요청할 수 있다. 이것은 동일한 파일이 가까운 다수의 서버들의 캐시 내에 저장되게 할 수 있고, 이는 캐시 인프라구조의 효율성을 낮춘다.

[0478] 이것은 동일한 블록을 요청하는 2명의 클라이언트들이 동일한 서버로부터 이 블록을 요청할 확률을 유리하게 증가시키는 시스템에 의해 핸들링될 수 있다. 여기에 설명된 새로운 방법은 요청될 파일의 식별자에 의해 결정되는 방식으로 및 인터넷 프로토콜 어드레스들 및 파일 식별자들의 동일하거나 유사한 선택들을 제시받은 상이한 클라이언트들이 동일한 선택을 수행하는 방식으로 이용 가능한 인터넷 프로토콜 어드레스들 중에서 선택하는 것을 포함한다.

[0479] 상기 방법의 제 1 실시예는 도 20을 참조하여 설명된다. 클라이언트는 우선 단계(1710)에 도시된 바와 같이 인터넷 프로토콜 어드레스들 IP_1, IP_2, ..., IP_n의 세트를 얻는다. 만약 단계(1720)에서 결정된 바와 같이, 요청들이 발행될 파일이 있다면, 단계들(1730-1770)에서 결정된 바와 같이, 발행할 인터넷 프로토콜 어드레스가 파일을 요청하는 것을 결정한다. 요청될 파일에 대한 식별자 및 인터넷 프로토콜 어드레스들의 세트가 주어지면 상기 방법은 파일 식별자에 의해 결정된 방식으로 인터넷 프로토콜 어드레스들의 순서를 결정하는 것을 포함한다. 예를 들어, 각각의 인터넷 프로토콜 어드레스에 대해 단계(1730)에 도시된 바와 같이, 인터넷 프로토콜 어드레스 및 파일 식별자의 연쇄(concatenation)를 포함하는 바이트 문자열은 구성된다. 해시 함수는 단계(1740)에서 도시된 바와 같이, 이런 바이트 문자열에 적용되고, 결과적인 해시 값들은 단계(1750)에서 도시된 바와 같이 고정된 순서, 예를 들어 증가하는 숫자 순서에 따라 배열되어, 인터넷 프로토콜 어드레스들에 대한 순서결정을 유도한다. 동일한 해시 함수는 모든 클라이언트들에 의해 사용될 수 있어서, 동일한 결과가 모든 클라이언트들에 의해 주어진 입력에 대한 해시 함수에 의해 형성되는 것을 보장한다. 해시 함수는 클라이언트들의 세트 내의 모든 클라이언트들로 구성될 수 있거나, 클라이언트 세트의 모든 클라이언트들은 클라이언트들이 인터넷 프로토콜 어드레스들의 리스트를 얻을 때 해시 함수의 부분 또는 전체 설명을 얻을 수 있거나, 클라이언트의 세트 내의 모든 클라이언트들은 클라이언트들이 파일 식별자를 얻을 때 해시 함수의 부분 또는 전체 설명을 얻을 수 있거나, 해시 함수는 다른 수단에 의해 결정될 수 있다. 이런 순서에서 최우선인 인터넷 프로토콜 어드레스는 선택되고 이 어드레스는 단계들(1760 및 1770)에서 도시된 바와 같이, 접속을 설정하기 위해 사용되고 파일의 모든 또는 부분들에 대한 요청들을 발행한다.

[0480] 상기 방법은 새로운 접속이 파일을 요청하기 위해 설정될 때 적용될 수 있다. 또한 다수의 설정된 접속들이 이용 가능하고 이들 중 하나가 새로운 요청을 발행하도록 선택될 수 있을 때 적용될 수 있다.

[0481] 게다가, 설정된 접속이 이용 가능하고 요청이 동일한 우선순위를 가진 후보 요청들의 세트 중에서 선택될 수 있을 때, 후보 요청들에 대한 순서는 예를 들어 상기 설명된 해시 값들과 동일한 방법에 의해 유도되고 이 순서에서 첫 번째 나타나는 후보 요청은 선택된다. 상기 방법들은 다시 접속 및 요청의 각각의 조합에 대한 해

시를 계산하고, 고정된 순서에 따라 이들 해시 값들의 순서를 결정하고 그리고 요청들 및 접속들의 조합들의 세트 상에서 유도된 순서에서 첫 번째 발생하는 조합을 선택함으로써, 접속들의 세트 중에서 접속 및 후보 요청과, 동일한 우선순위의 요청들 둘 다를 선택하도록 조합될 수 있다.

[0482] 이 방법은 다음 이유에 때문에 장점을 가진다: 도 1(BSI(101)) 또는 도 2(BSI들(101))에 도시된 바와 같은 블록 서빙 인프라구조에 의해 취해진 통상적인 해결책, 및 특히 CDN들에 의해 일반적으로 취해진 해결책은 클라이언트 요청들을 수신하는 다수의 캐싱 프록시 서버들을 제공하는 것이다. 캐싱 프록시 서버는 주어진 요청에서 요청된 파일이 제공될 수 없고 이 경우 그런 서버들은 통상적으로 상기 요청을 다른 서버에 포워드하고, 통상적으로 요청된 파일을 포함하는 상기 서버로부터 응답을 수신하고, 그리고 상기 응답을 클라이언트에게 포워드한다. 캐싱 프록시 서버는 요청된 파일을 또한 저장(캐시)하여, 파일에 대한 추후 요청들에 즉각적으로 응답할 수 있다. 상기된 일반적인 해결책은 캐싱 프록시 서버가 수신된 요청들의 세트에 의해 주어진 캐싱 프록시 서버상에 저장된 파일들의 세트가 주로 결정되는 특성을 가진다.

[0483] 상기된 방법은 다음 장점을 가진다. 클라이언트들의 세트 내의 모든 클라이언트들이 인터넷 프로토콜 어드레스의 동일한 리스트를 가지면 이들 클라이언트들은 동일한 파일에 대해 발행된 모든 요청들에 대해 동일한 인터넷 프토토콜 어드레스를 사용할 것이다. 인터넷 프로토콜 어드레스들의 2개의 상이한 리스트들이 존재하고 각각의 클라이언트가 이들 2개의 리스트들 중 하나를 가지면 클라이언트들은 동일한 파일에 대해 발행된 모든 요청들에 대해 최대 2개의 상이한 인터넷 프로토콜 어드레스들을 사용할 것이다. 일반적으로, 클라이언트들에 제공된 인터넷 프로토콜 어드레스들의 리스트들이 유사하다면, 클라이언트들은 동일한 파일에 대해 발행된 모든 요청들에 대해 제공된 인터넷 프로토콜 어드레스들의 작은 세트를 사용할 것이다. 가까운 클라이언트들이 인터넷 프로토콜 어드레스들의 유사한 리스트들을 제공받는 경향이 있기 때문에, 인접한 클라이언트들이 이들 클라이언트들에 이용 가능한 캐싱 프록시 서버들의 작은 부분만으로부터 파일에 대한 요청들을 발행하는 것은 가능할 것 같다. 따라서, 파일을 캐시하는 캐싱 프록시 서버들의 작은 부분만이 존재할 것이고, 이는 유리하게 파일을 캐시하기 위하여 사용된 캐싱 자원들의 양을 최소화한다.

[0484] 인터넷 프로토콜 어드레스들의 주어진 세트에 대해, 인터넷 프로토콜 어드레스들 중 주어진 하나가 단계(1750)에 의해 형성된 분류된 리스트에서 첫 번째인 파일들의 부분이 대략적으로 리스트 내의 모든 인터넷 프로토콜 어드레스들에 대해 동일한 것을 보장하기 위하여, 바람직하게 해시 함수는 상이한 입력들의 매우 작은 부분이 동일한 출력에 맵핑되고, 상이한 입력들이 본질적으로 랜덤 출력들에 맵핑되는 특성을 가진다. 다른 한편 주어진 입력에 대해 해시 함수의 출력이 모든 클라이언트들에 대해 동일하다는 점에서 해시 함수가 결정론적인 것이 중요하다.

[0485] 상기된 방법의 다른 장점은 아래에 있다. 클라이언트들의 세트 내의 모든 클라이언트들이 인터넷 프로토콜 어드레스들과 동일한 리스트를 제공받는 것을 가정하자. 바로 이전에 설명된 해시 함수의 특성들로 인해, 이들 클라이언트들로부터 상이한 파일들에 대한 요청들이 인터넷 프로토콜 어드레스들의 세트에 걸쳐 균등하게 확산될 것이 가능하고, 이는 차례로 요청들이 캐싱 프록시 서버들에 걸쳐 균등하게 확산할 것을 의미한다. 따라서, 이들 파일들을 저장하기 위해 사용된 캐싱 자원들은 캐싱 프록시 서버들에 걸쳐 균등하게 확산되고, 파일들에 대한 요청들은 캐싱 프록시 서버들에 걸쳐 균등하게 확산된다. 따라서, 상기 방법은 캐싱 인프라구조에 걸쳐 저장 밸런싱 및 로드 밸런싱 둘 다를 제공한다.

[0486] 상기된 해결책에 대한 다수의 변형들은 당업자에게 알려지고 많은 경우들에서 이들 변형들은 주어진 프록시 상에 저장된 파일들의 세트가, 캐싱 프록시 서버가 수신한 요청들의 세트에 의해 적어도 부분적으로 결정되는 특성을 유지한다. 주어진 호스트네임이 다수의 물리적 캐싱 프록시 서버들을 결정하는 일반적인 경우에서, 모든 이들 서버들이 빈번하게 요청된 임의의 주어진 파일의 카피를 균등하게 저장하는 것은 일반적일 것이다. 그런 복제는 바람직하지 않은데, 그 이유는 캐싱 프록시 서버들 상의 저장 자원들이 한정되고 결과적으로 파일들이 때때로 캐시로부터 제거(퍼지(purge))될 수 있기 때문이다. 여기에 설명된 새로운 방법은 이런 복제가 감소되는 방식으로 주어진 파일에 대한 요청들이 캐싱 프록시 서버들에 지시되는 것을 보장하여, 캐시로부터 파일들을 제거할 필요를 감소시키고 이에 따라 임의의 주어진 파일이 프록시 캐시 내에 존재(프록시 캐시로부터 퍼지되지 않음)할 가능성을 증가시킨다.

[0487] 파일이 프록시 캐시 내에 존재할 때, 클라이언트에 전송된 응답은 더 빠르고, 이것은 미디어 플레이아웃의 일시정지 및 그러므로 바람직하지 않은 사용자 경험을 유발할 수 있는 요청된 파일이 늦게 도달할 확률을 감소시키는 장점을 가진다. 부가적으로, 파일이 프록시 캐시 내에 존재하지 않을 때, 요청은 다른 서버에 전송될 수 있어서, 서빙 인프라구조 및 서버들 사이의 네트워크 접속들 둘 다에 대한 부가적인 로드를 유발한다. 많은 경우들에서 요청이 전송된 서버는 먼 위치에 있을 수 있고 이 서버로부터 다시 캐싱 프록시 서버로 파일의

전송은 전송 비용들을 부과할 수 있다. 그러므로 여기에 설명된 새로운 방법은 이들 전송 비용들의 감소를 유발한다.

[0488] 개연적 전체 파일 요청

[0489] HTTP 프로토콜이 범위 요청들(Range requests)과 함께 사용되는 경우의 특정 관심은 서빙 인프라구조 내의 확장성(scalability)을 제공하기 위하여 일반적으로 사용된 캐시 서버들의 동작이다. HTTP 캐시 서버들이 HTTP 범위 헤더(HTTP Range header)를 지원하는 것이 일반적이지만, 상이한 HTTP 캐시 서버들의 정확한 동작은 구현에 의해 가변한다. 대부분의 캐시 서버 구현들은 파일이 캐시 내에서 이용 가능한 경우 캐시로부터 범위 요청들을 서빙한다. HTTP 캐시 서버들의 공통의 구현은 캐시 서버가 파일의 카피를 가지지 않으면(캐시 서버 또는 본래의 서버) 범위 헤더를 포함하는 다운스트림 HTTP 요청들을 업스트림 노드에 항상 포워드한다. 몇몇 구현들에서 범위 요청에 대한 업스트림 응답은 전체 파일이고, 이런 전체 파일은 캐시되고 다운스트림 범위 요청에 대한 응답은 이 파일로부터 추출되고 전송된다. 그러나, 적어도 하나의 구현에서, 범위 요청에 대한 업스트림 응답은 범위 요청 자체 내의 바로 데이터 바이트들이고 이들 데이터 바이트들은 캐시되는 것이 아니고 대신 다운스트림 범위 요청에 대한 응답으로서 전송된다. 결과적으로, 클라이언트들에 의한 범위 헤더들의 사용은 파일 자체가 결코 캐시들 내로 운반되지 않고 네트워크의 원하는 확장성 특성들이 손실될 것이라는 결과를 가질 수 있다.

[0490] 상기에서, 캐싱 프록시 서버들의 동작이 설명되었고 또한 다수의 블록들의 어그리케이션들인 파일로부터 블록들을 요청하는 방법은 설명되었다. 예를 들어 이것은 HTTP 범위 요청 헤더의 사용에 의해 달성될 수 있다. 그런 요청들은 다음에서 "부분 요청들"이라 지칭된다. 추가 실시예는 블록 서빙 인프라구조(101)가 HTTP 범위 헤더에게 완전한 지원을 제공하지 않는 경우 장점을 갖는 것이 지금 설명된다. 일반적으로, 블록 서빙 인프라구조 내의 서버들, 예를 들어 컨텐트 전달 네트워크(Content Delivery Network)는 부분 요청들을 지원하지만 로컬 스토로지(캐시) 내의 부분 요청들에 대한 응답을 저장할 수 없다. 그런 서버는, 전체 파일이 로컬 스토로지 내에 저장되지 않으면 ? 상기 경우 응답은 다른 서버에 요청을 포워드없이 전송될 수 있음 ?, 다른 서버에 요청을 포워딩함으로써 부분 요청을 이행할 수 있다.

[0491] 상기 설명된 블록 어그리게이션의 새로운 개선을 이용하는 블록-요청 스트리밍 시스템은 블록 서빙 인프라구조가 이런 동작을 나타내면 바람직하지 않게 수행될 수 있는데, 그 이유는 부분 요청들인 모든 요청들이 다른 서버로 포워딩될 것이고 요청들은 캐싱 프록시 서버들에 의해 서빙되지 않을 것이고, 제 1 장소 내의 캐싱 프록시 서버들을 제공하는 목적을 좌절시키기 때문이다. 상기된 바와 같이 블록-요청 스트리밍 프로세스 동안, 클라이언트는 몇몇 포인트에서 파일의 시작에 있는 블록을 요청할 수 있다.

[0492] 여기에 설명된 새로운 방법에 따라, 명시 조건이 충적될 때마다, 그런 요청들은 파일 내의 제 1 블록(Block)에 대한 요청들로부터 전체 파일에 대한 요청들로 변환될 수 있다. 전체 파일에 대한 요청이 캐싱 프록시 서버에 의해 수신될 때, 프록시 서버는 통상적으로 응답을 저장한다. 그러므로 이들 요청들의 사용은 파일이 로컬 캐싱 프록시 서버들의 캐시로 운반되게 하여, 전체 파일이든 부분 파일이든 추후 요청들은 캐싱 프록시 서버에 의해 직접적으로 서빙될 수 있다. 상기 조건은, 주어진 파일과 연관된 요청들의 세트, 예를 들어 당해의 컨텐트 아이템을 보여주는 클라이언트들의 세트에 의해 생성된 요청들의 세트 중, 조건이 이들 요청들의 적어도 제공된 프랙션에 대해 충족되도록 할 수 있다.

[0493] 적당한 조건의 예는 랜덤하게 선택된 수가 제공된 임계값보다 높은 것이다. 이 임계값은 전체 파일 요청으로의 단일 블록 요청의 변환이 요청들의 제공된 프랙션에 대해 평균, 예를 들어 10번 중 1번(상기 경우 랜덤 번호는 인터벌 [0,1]로부터 선택될 수 있고 임계값은 0.9일 수 있음) 발생하도록 설정될 수 있다. 적당한 조건의 다른 예는 블록과 연관된 몇몇 정보 및 클라이언트와 연관된 몇몇 정보에 걸쳐 계산된 해시 함수가 값들의 제공된 세트 중 하나를 취하는 것이다. 이 방법은 빈번하게 요청된 파일에 대해, 파일이 로컬 프록시 서버의 캐시로 운반될 장점을 가지지만, 블록-요청 스트리밍 시스템의 동작은 각각의 요청이 단일 Block에 대한 것인 표준 동작으로부터 크게 변경되지 않는다. 많은 경우들에서, 단일 Block 요청으로부터 전체 파일 요청으로 요청의 변환이 발생하는 경우, 클라이언트 절차들은 파일 내의 다른 Block들을 요청하기 위하여 계속 진행될 것이다. 이것이 그 경우이면, 그런 요청들은 당해 Block들이 임의의 경우 전체 파일 요청의 결과로서 수신될 것이기 때문에 억제될 수 있다.

[0494] URL 구성 및 세그먼트 리스트 생성 및 탐색

[0495] 세그먼트 리스트 생성은 주문의 경우들에 대한 미디어 프리젠테이션의 시작에 관련하여 또는 벽-시계 시간으로 표현된 몇몇 시작 시간(starttime)에서 시작하는 명시 표현을 위해 명시 클라이언트-로컬 시간 NOW에서 클라이언트가 MPD로부터 세그먼트 리스트를 어떻게 생성할 수 있는가의 문제를 다룬다. 세그먼트 리스트는 선택적 개시 표현 메타데이터뿐 아니라 미디어 세그먼트들의 리스트에 대한 로케이터, 예를 들어 URL을 포함할 수 있다. 각각의 미디어 세그먼트는 시작시간, 지속시간 및 로케이터가 할당되었을 수 있다. 시작시간은 반드시 샘플 정밀 시간이 아닌, 통상적으로 세그먼트 내에 포함된 미디어의 미디어 시간의 근사값을 표현한다. 시작시간은 적당한 시간에 다운로드 요청을 발생하기 위하여 HTTP 스트리밍 클라이언트에 의해 사용된다. 각각의 시작 시간을 포함하는 세그먼트 리스트의 생성은 상이한 방식으로 행해질 수 있다. URL들은 플레이 리스트 또는 URL 구성 규칙이 세그먼트 리스트의 컴팩트 표현을 위해 바람직하게 사용될 수 있을 때 제공될 수 있다.

[0496] URL 구성에 기초한 세그먼트 리스트는 예를 들어 FileDynamicInfo 또는 동등한 시간 같은 명시 속성 또는 엘리먼트에 의해 MPD가 시그널링하면 수행될 수 있다. URL 구성으로부터 세그먼트 리스트를 생성하기 위한 일반적인 방식은 "URL 구성 개요" 섹션에서 아래에 제공된다. 플레이리스-기반 구성은 예를 들어 상이한 신호에 의해 시그널링될 수 있다. 세그먼트 리스트의 탐색 및 정확한 미디어 시간에 대한 획득은 또한 바람직하게 이런 환경에서 구현된다.

[0497] URL 구성기 개요

[0498] 이전에 설명된 바와 같이, 본 발명의 일 실시예에서 클라이언트 디바이스들이 표현의 Block들에 대한 파일 식별자들을 구성하게 하는 URL 구성 규칙들을 포함하는 메타데이터 파일이 제공될 수 있다. 우리는 지금 URL 구성 규칙들에 대한 변화들을 포함하는 메타데이터 파일의 변화들, 이용 가능한 인코딩들의 수에 대한 변화들, 비트레이트, 종횡비, 해상도, 오디오 또는 비디오 코덱 또는 코덱 파라미터들 또는 다른 파라미터들 같은 이용 가능한 인코딩들과 연관된 메타데이터에 대한 변화들을 제공하는 블록 요청 스트리밍 시스템에 대한 추가 새로운 개선을 설명한다.

[0499] 이런 새로운 개선에서, 전체 표현 내의 시간 인터벌을 가리키는 메타데이터 파일의 각각의 엘리먼트와 연관된 부가적인 데이터가 제공될 수 있다. 이런 시간 인터벌 내에서 엘리먼트는 유효 그렇지 않으면 엘리먼트가 무시될 수 있는 시간 인터벌로 고려될 수 있다. 게다가, 메타데이터의 신택스(syntax)는 단지 1번 도는 최대 1번 나타나도록 이전에 허용된 엘리먼트가 다수번 나타날 수 있도록 개선될 수 있다. 부가적인 제한은 그런 엘리먼트들에 대해 명시 시간 인터벌들이 해체되어야 하는 것을 제공하는 이런 경우에 적용될 수 있다. 임의의 주어진 시간 순간에, 주어진 시간을 포함하는 시간 인터벌을 가진 엘리먼트들만이 본래의 메타데이터 신택스와 일치하는 메타데이터 파일을 유발하는 것을 고려하자. 우리는 그런 시간 인터벌들을 유효 인터벌들이라 지칭한다. 그러므로 이 방법은 상기된 종류의 단일 메타데이터 파일 변화들 내에서 시그널링을 제공한다. 바람직하게, 그런 방법은 프리젠테이션 내의 명시 포인트들에서 설명된 종료의 변화들을 지원하는 메타데이터 프리젠테이션을 제공하기 위해 사용될 수 있다.

[0500] URL 구성기

[0501] 여기에 설명된 바와 같이, 블록-요청 스트리밍 시스템들의 공통 특징은 클라이언트에게 이용 가능한 미디어 인코딩들을 식별하는 "메타데이터"를 제공하고 클라이언트에 의해 필요한 정보를 이들 인코딩들로부터 블록들을 요청하기 위해 제공할 필요이다. 예를 들어 HTTP의 경우 이 정보는 미디어 블록들을 포함하는 파일들에 대한 URL들을 포함할 수 있다. 주어진 인코딩에 대한 블록들에 대해 URL들을 리스트하는 플레이리스트 파일은 제공될 수 있다. 다수의 플레이리스트 파일들은 상이한 인코딩들에 상응하는 플레이리스트들을 리스트하는 마스터 플레이리스트-오브-플레이리스트들과 함께 각각의 인코딩을 위해 하나가 제공된다. 이런 시스템의 단점은 클라이언트가 스트림을 시작할 때 메타데이터가 매우 크게될 수 있고 그러므로 요청되도록 약간의 시간이 걸린다는 것이다. 이런 시스템의 추가 단점은, 미디어 데이터 블록들에 상응하는 파일들이 라이브 스포츠 이벤트 또는 뉴스 프로그램 같은 실시간(라이브)으로 캡쳐된 미디어 스트림으로부터 "온-더-플라이" 생성될 때, 라이브 컨텐트의 경우 명백하다. 이 경우 플레이리스트 파일들은 새로운 블록이 이용 가능한 각각의 시간(예를 들어 매 몇 초들)에 업데이트될 수 있다. 클라이언트 디바이스들은 만약 새로운 블록들이 이용 가

능하고 그들의 URL들을 얻는지를 결정하기 위하여 플레이리스트 파일을 반복적으로 인출할 수 있다. 이것은 서빙 인프라구조 상에 상당한 로드를 배치시킬 수 있고 특히 메타데이터 파일들이 일반적으로 몇 초 정도인 블록과 동일한 업데이트 인터벌보다 길게 캐시될 수 없다는 것을 의미한다.

[0502] 블록-요청 스트리밍 시스템의 하나의 중요한 양상은 파일 다운로드 프로토콜과 함께 Block들을 요청하기 위하여 사용되어야 하는 파일 식별자들, 예를 들어 URL들을 클라이언트들에게 알리기 위해 사용된 방법이다. 상기 방법에서 예를 들어, 프리젠테이션의 각각의 표현을 위해 미디어 데이터의 Block들을 포함하는 파일들의 URL들을 리스트하는 플레이리스트 파일이 제공된다. 이 방법의 단점은 플레이아웃이 시작되기 전에 플레이리스트 파일 자체의 적어도 일부가 다운로드될 필요가 있어서, 채널 재핑 시간을 증가시키고 그러므로 나쁜 사용자 경험을 유발한다는 것이다. 몇몇 또는 많은 표현들을 이용한 긴 미디어 프리젠테이션에 대해, 파일 URL들의 리스트는 클 수 있고 따라서 플레이리스트 파일은 더 클 수 있어서 채널 재핑 시간을 추가로 증가시킨다.

[0503] 이 방법의 다른 단점은 라이브 컨텐트의 경우에 발생한다. 이 경우, URL들의 완전한 리스트는 미리 이용 가능하지 않고 플레이리스트 파일은 새로운 블록들이 이용 가능할 때 주기적으로 업데이트되고 클라이언트들은 업데이트된 버전을 수신하기 위해, 플레이리스트 파일을 주기적으료 요청한다. 이 파일이 빈번하게 업데이트되기 때문에, 캐싱 프록시 서버들 내에 오래 저장될 수 없다. 이것은 이 파일에 대한 매우 많은요청들이 다른 서버들에 그리고 궁극적으로 파일을 생성하는 서버에 포워드될 것을 의미한다. 대중적 미디어 프리젠테이션의 경우, 이것은 이 서버 및 네트워크 상에 높은 로드를 유발하고, 차례로 느린 응답 시간을 유발하고 그리므로 높은 채널 재핑 시간 및 나쁜 사용자 경험을 유발한다. 최악의 경우 서버는 오버로드되고 이것은 몇몇 사용자들이 프리젠테이션을 시청할 수 없게 한다.

[0504] 블록-요청 스트리밍 시스템의 설계에서 사용될 수 있는 파일 식별자들의 형태에 대해 제한들을 두는 것을 회피하는 것은 바람직하다. 이것은 다수의 고려들이 특정 형태의 식별자들의 사용을 유도하기 때문이다. 예를 들어, 블록 서빙 인프라구조(Block Serving Infrastructure)가 컨텐트 전달 네트워크(Content Delivery Network)인 경우 시스템 설계 시간에서 예측될 수 없는 파일 식별자의 특정 형태들을 유도하는 네트워크 또는 다른 요건들에 걸쳐 스토로지 또는 서빙 로드를 분산시킬 욕구에 관련된 파일 네이밍 또는 스토로지 변환들이 존재할 수 있다.

[0505] 적당한 파일 식별 변환들을 선택하기 위하여 유연성을 유지하면서 상기 언급된 단점들을 완화하는 추가 실시예가 지금 설명된다. 이 방법에서 메타데이터는 파일 식별자 구성 규칙을 포함하는 미디어 프리젠테이션의 각각의 표현을 위해 제공될 수 있다. 파일 식별자 구성 규칙은 예를 들어 텍스트 문자열을 포함할 수 있다. 프리젠테이션의 주어진 블록에 대한 파일 식별자를 결정하기 위하여, 파일 식별자 구성의 해석 방법은 제공될 수 있고, 이 방법은 입력 파라미터들과 함께 파일 식별 구성 규칙의 평가 및 입력 파라미터들의 결정을 포함한다. 이 입력 파라미터들은 예를 들어 식별될 파일의 인덱스를 포함할 수 있고, 여기서 제 1 파일은 인덱스 제로를 가지며, 제 2 파일은 인덱스 하나를 가지며, 제 3 파일은 인덱스 둘을 가지며 등등이 있다. 예를 들어, 매 파일이 동일한 시간 지속시간(또는 대략적으로 동일한 시간 지속시간)에 걸치는 경우에서, 프리젠테이션 내의 임의의 주어진 시간과 연관된 파일의 인덱스는 쉽게 결정될 수 있다. 대안적으로, 각각의 파일에 의해 걸쳐진 프리젠테이션 내의 시간은 프리젠테이션 또는 버전 메타데이터 내에 제공될 수 있다.

[0506] 일 실시예에서, 파일 식별자 구성 규칙은 입력 파라미터들에 상응하는 명시 특별한 식별자들을 포함할 수 있는 텍스트 문자열을 포함할 수 있다. 파일 식별자 구성 규칙의 평가 방법은 텍스트 문자열 내의 특별한 식별자들의 위치들을 결정하고 상응하는 입력 파라미터의 값의 문자 표현으로 상기 특별한 식별자를 대체하는 것을 포함한다.

[0507] 다른 실시예에서, 파일 식별자 구성 규칙은 익스프레션(expression) 언어에 부합하는 텍스트 문자열을 포함할 수 있다. 익스프레이션 언어는 언어 내의 익스프레이션들이 순응할 수 있는 신택스 및 신택스에 부합하는 문자열을 평가하기 위한 규칙들의 세트를 포함한다.

[0508] 특별한 예가 도 21을 참조하여 이하에 지금 설명될 것이다. 증분된 배커스-나우(Backus-Naur) 형에서 정의된 적당한 익스프레이션 언어에 대한 신택스 정의의 예는 도 21에 도시된 바와 같다. 도 21의 <익스프레이션> 프러덕션(production)에 부합하는 문자열을 평가하기 위한 규칙들의 예는 <익스프레션> 프러덕션(<익스프레션>)에 부합하는 문자열을 다음과 같이 <문자> 프러덕션에 부합하는 문자열로 반복적으로 변환하는 것을 포함한다.

[0509] <문자> 프러덕션에 부합하는 <익스프레션>은 변화되지 않는다.

[0510] <변수> 프러덕션에 부합하는 <익스프레션>은 <변수> 프러덕션의 <토큰> 문자열에 의해 식별된 변수의 값으로 대체된다.

[0511] <함수> 프러덕션에 부합하는 <익스프레션>은 이들 규칙들에 따라 자신의 아규먼트들 각각을 평가하고 그리고 하기에 설명된 바와 같이 <함수>의 <토큰> 엘리먼트에 따라 이들 아규먼트들에 변환을 적용함으로써 평가된다.

[0512] <익스프레션> 프러덕션의 최종 대안에 부합하는 <익스프레션>은 2개의 <익스프레션> 엘리먼트들을 평가하고 그리고 하기에 설명된 바와 같이 <익스프레션>의 최종 대안의 <오퍼레이터> 엘리먼트에 따라 이들 아규먼트들에 연산을 적용함으로써 평가된다.

[0513] 상기된 방법에서 다수의 변수들의 정의될 수 있는 환경에서 평가가 발생하는 것이 가정된다. 변수는 (네임, 값) 쌍이고 여기서 "네임"은 <토큰> 프러덕션에 부합하는 문자열이고 "값"은 <문자> 프러덕션에 부합하는 문자열이다. 몇몇 변수들은 평가가 시작되기 전에 평가 외측에서 정의될 수 있다. 다른 변수들은 평가 프로세서 자체 내에서 정의될 수 있다. 모든 변수들은 단지 하나의 변수가 각각의 가능한 "네임"으로 존재한다는 점에서 "글로벌"이다.

[0514] 함수의 예는 "printf" 함수이다. 이 함수는 하나 또는 그 초과의 아규먼트들을 허용한다. 제 1 아규먼트는 <문자열> 프러덕션(이후 "문자열")에 부합할 수 있다. printf 함수는 자신의 제 1 아규먼트의 변환된 버전을 평가한다. 적용된 변환은 C 표준 라이브러리의 "printf" 함수와 동일하고, 부가적인 아규먼트들은 C 표준 라이브러리 printf 함수에 의해 예상된 부가적인 아규먼트들을 공급하는 <함수> 프러덕션 내에 포함된다.

[0515] 함수의 다른 예는 "해시" 함수이다. 이 함수는 2개의 아규먼트들을 허용하고, 제 1 아규먼트는 문자열일 수 있고 제 2 아규먼트는 <번호> 프러덕션(이후 "번호")에 부합할 수 있다. "해시" 함수는 해시 알고리즘을 제 1 아규먼트에 적용하고 제 2 아규먼트보다 작은 음수가 아닌 정수인 결과들을 리턴한다. 적당한 해시 함수의 예는 도 22에 도시된 C 함수에서 제공되고, 상기 C 함수의 아규먼트들은 입력 문자열(에워싸는 인용 마크들을 제외) 및 수치 입력 값이다. 해시 함수들의 다른 예들은 당업자에게 잘 알려졌다.

[0516] 함수의 다른 예는 1, 2 또는 3개의 문자열 아규먼트들을 취하는 "Subst" 함수이다. 하나의 아규먼트가 공급되는 경우, "Subst" 함수의 결과는 제 1 아규먼트이다. 2개의 아규먼트들이 공급되는 경우, "Subst" 함수의 결과는 제 1 아규먼트 내의 제 2 아규먼트(에워싸는 인용 마크들 제외)의 임의의 발생들을 소거하고 그렇게 변형된 제 1 아규먼트를 리터닝함으로써 계산된다. 3개의 아규먼트들이 공급되는 경우, "Subst" 함수의 결과는 제 3 아규먼트(에워싸는 인용 마크들 제외)를 가진 제 1 아규먼트 내의 제 2 아규먼트(에워싸는 인용 마크들 제외)의 임의의 발생을 대체하고, 그렇게 변형된 제 1 아규먼트를 리터닝함으로써 계산된다.

[0517] 오퍼레이터들의 몇몇 예들은 "오퍼레이터" 프러덕션들 '+', '-', '/', '*', '%' 각각에 의해 식별된 가산, 감산, 나눗셈, 곱셈 및 율 오퍼레이터들이다. 이들 오퍼레이터들은 <오퍼레이터> 프러덕션의 어느 한 측 <익스프레이션> 프러덕션들이 수들을 평가하는 것을 요구한다. 오퍼레이터의 평가는 적당한 산술 연산(각각 가산, 감산, 나눗셈, 곱셈 및 율)을 이들 수들에 일반적으로 적용하고 <수> 프러덕션에 부합하는 형태의 결과를 리터닝하는 것을 포함한다.

[0518] 오퍼레이터의 다른 예는 <오퍼레이터> 프러덕션 '='에 의해 식별되는 할당 오퍼레이터이다. 이 오퍼레이터는 좌측 아규먼트가 <토큰> 프러덕션에 부합하는 컨텐츠의 문자열을 평가하는 것을 요구한다. 문자열의 컨텐트는 에워싸는 인용 마크들 내의 문자 문자열인 것으로 정의된다. 등식 오퍼레이터는 우측 아규먼트를 평가한 결과와 동일한 값이 할당될 좌측 아규먼트의 컨텐트와 동일한 <토큰>인 네임을 가진 변수를 야기한다. 이 값은 또한 오퍼레이터 익스프레이션을 평가한 결과이다.

[0519] 오퍼레이터의 다른 예는 <오퍼레이터> 프러덕션 ';'에 의해 식별된 시퀀스 오퍼레이터이다. 이 오퍼레이터의 평가 결과는 우측 아규먼트이다. 모든 오퍼레이터들 처럼, 양쪽 아규먼트들이 평가되고 좌측 아규먼트가 먼저 평가되는 것이 주의된다.

[0520] 본 발명의 일 실시예에서 파일의 식별자는 요구된 파일을 식별하는 입력 변수들의 명시 세트로 상기 규칙에 따라 파일 식별자 구성 규칙을 평가함으로써 얻어질 수 있다. 입력 변수의 예는 프리젠테이션 내의 파일의 수칙 인덱스와 동일한 네임 "인덱스" 및 값을 가진 변수이다. 입력 변수의 다른 예는 프리젠테이션의 요구된 버전의 평균 비트레이트와 동일한 값 및 네임 "비트레이트"를 가진 변수이다.

[0521] 도 23은 파일 식별자 구성 규칙들의 몇몇 예들들을 예시하고, 여기서 입력 변수들은 "id"이고, 원하는 프리젠

테이션의 표현에 대한 식별자 및 "seq"를 제공하고, 파일에 대한 시퀀스 번호를 제공한다.

[0522] 본 개시를 읽은 당업자에게 명확할 바와 같이, 상기 방법의 다수의 변수들은 가능하다. 예를 들어, 상기된 전부가 아닌 함수들 및 오퍼레이터들은 제공될 수 있거나 부가적인 함수들 또는 오퍼레이터들은 제공될 수 있다.

[0523] URL 구성 규칙들 및 타이밍

[0524] 이 섹션은 미디어 프리젠테이션 및 표현 내의 각각의 세그먼트에 대한 시작 시간뿐 아니라 파일 또는 세그먼트 URI를 할당하기 위한 기본 URI 구성 규칙들을 제공한다.

[0525] 이런 절에 대해 클라이언트에서 미디어 표현 설명의 이용 가능성은 가정된다.

[0526] HTTP 스트리밍 클라이언트가 미디어 프리젠테이션 내에 다운로드된 미디어를 플레잉 아웃하는 것을 가정한다. HTTP 클라이언트의 실제 표현 시간은 프리젠테이션 시간이 프리젠테이션의 시작에 관련되는지에 대해 정의될 수 있다. 초기화시, 프리젠테이션 시간 t=0는 가정될 수 있다.

[0527] 임의의 포인트에서, HTP 클라이언트는 실제 프리젠테이션 시간(t) 이전 최대 MaximumClientPreBufferTime의 플레이-시간(tp)(또한 프리젠테이션의 시작에 관련하여)을 사용하여 임의의 데이터를 다운로드할 수 있고 사용자 상호작용, 예를 들어 탐색, 고속-포워드, 등으로 인해 요구된 임의의 데이터를 다운로드할 수 있다. 몇몇 실시예들에서 MaximumClientPreBufferTime은 클라이언트들이 제한들 없이 현재 플레이-시간(tP) 이전 데이터를 다운로드할 수 있다는 점에서 심지어 명시되지 않을 수 있다.

[0528] HTTP 클라이언트는 불필요한 데이터를 다운로딩하는 것을 회피할 수 있고, 예를 들어 플레이-아웃될 것으로 예상되지 않는 표현들로부터의 임의의 세그먼트들은 통상적으로 다운로드될 수 없다.

[0529] 스트리밍 서비스들을 제공하는 것의 기본 프로세스는 예를 들어 HTTP GET 요청들 또는 HTTP 부분 GET 요청들을 사용함으로써 전체 파일들/세그먼트들 또는 파일들/세그먼트들의 서브세트를 다운로드할 적당한 요청들의 생성에 의해 데이터의 다운로딩일 수 있다. 이 설명은 명시 플레이-시간(tP) 동안 데이터에 액세스하는 방법을 처리하지만 일반적으로 클라이언트는 불충분한 요청들을 회피하기 위하여 플레이-시간의 보다 큰 시간 범위 동안 데이터를 다운로드할 수 있다. HTTP 클라이언트는 스트리밍 서비스를 제공할 때 HTTP 요청들의 수/빈도를 최소화할 수 있다.

[0530] 플레이-시간(tP)에서 또는 명시 표현의 플레이-시간(tP)에 적어도 근접할 때 미디어 데이터에 액세싱하기 위하여, 클라이언트는 이런 플레이-시간을 포함하는 파일에 대한 URL을 결정하고 그리고 게다가 이런 플레이-시간에 액세스하기 위해 파일 내의 바이트 범위를 결정한다.

[0531] Media Presentation Description은 표현 id(r)을 예를 들어 RepresentationID 속성의 사용에 의해 각각의 표현에 할당할 수 있다. 다른 말로, 섭치 시간에 의해 기록되거나 클라이언트에 의해 판독될 때 MPD의 컨텐트는, 할당이 있도록 해석될 것이다. id(r)을 가진 명시 표현에 대해 명시 플레이-시간(tP) 동안 데이터를 다운로드하기 위해, 클라이언트는 파일에 대해 적당한 URI를 구성할 수 있다.

[0532] Media Presentation Description은 각각의 파일 또는 각각의 표현(r)의 세그먼트를 다음 속성들에 할당할 수 있다.

[0533] (a) 표현(r) 내의 파일의 시퀀스 번호(i), i=1,2,..., Nr, (b) 표현 id(r)을 가진 파일 및 ts(r,i)로서 정의된 프리젠테이션 시간에 관련하여 파일 인덱스(i)의 상대적 시작 시간, (c) FileURI(r,i)로서 표시된 파일 인덱스(i) 및 표현 id(r)을 가진 파일/세그먼트에 대한 URI.

[0534] 일 실시예에서 파일 및 파일 URI들의 시작 시간은 명확하게 표현을 위해 제공될 수 있다. 다른 실시예에서, URI들의 리스트는 명확하게 제공될 수 있고 여기서 각각의 파일 URI는 리스트의 위치에 따라 인덱스(i)가 고유하게 할당되게 하고 세그먼트의 시작 시간은 1 내지 i-1의 세그먼트들에 대한 모든 세그먼트 지속시간들의 합으로서 유도된다. 각각의 세그먼트의 지속시간은 상기 논의된 임의의 규칙들에 따라 제공될 수 있다. 예를 들어, 기본 수리 연산들의 당업자는 단일 엘리먼트 또는 속성으로부터 시작 시간 및 표현 내의 파일 URI의 위치/인덱스를 쉽게 유도하기 위한 방법론을 유도하기 위한 다른 방법들을 사용할 수 있다.

[0535] 만약 다이내믹 URI 구성 규칙이 MPD로 제공된다면, 각 파일의 시작 시간 및 각 파일 URI는 구성 규칙, 요청된 파일의 인덱스, 및 잠재적으로 미디어 프리젠테이션 설명으로 제공되는 일부 추가적인 파라미터들을 사용함으

로써 다이내믹하게 구성될 수 있다. 정보는 예컨대 FileURIPattern 및 FileInfoDynamic와 같은 MPD 속성들 및 엘리먼트로 제공될 수 있다. FileURIPattern은 파일 인덱스 시퀀스 번호 i 및 리프리젠테이션 ID r에 기초하여 URI들을 어떻게 구성할지에 대한 정보를 제공한다. FileURIFormat는 아래와 같이 구성되고:

[0536] FileURIFormat=sprintf("%s%s%s%s%s.%s", BaseURI, BaseFileName, RepresentationIDFormat, SeparatorFormat, FileSequenceIDFormat, FileExtension)

[0537] FileURI(r,i)는 다음과 같이 구성된다:

[0538] FileURI(r,i)=sprintf(FileURIFormat, r, i)

[0539] 각각의 파일/세그먼트에 대한 상대적인 시작 시간 ts(r,i)은 이러한 리프리젠테이션에서 세그먼트들의 지속시간을 설명하는 MPD에 보유된 임의의 속성, 예컨대 FileInfoDynamic 속성에 의해 유도될 수 있다. MPD는 또한 위에서 규정된 것과 동일한 방식으로 적어도 기간 내의 모든 리프리젠테이션에 대해 또는 미디어 프리젠테이션에서 모든 리프리젠테이션들에 대해 글로벌한 FileInfoDynamic 속성들의 시퀀스를 부유할 수 있다. 만약 리프리젠테이션 r의 특정 플레이-시간 tP에 대한 미디어 데이터가 요청된다면, 상응하는 인덱스 i(r,tP)가 $i(r,t_p)$로서 유도될 수 있고, 그럼으로써 이러한 인덱스의 플레이-시간은 ts(r,i(r,tP)) 및 ts(r,i(r,tP)+1)의 시작 시간의 인터벌 내에 있다. 세그먼트 액세스는 위의 경우들에 의해 추가로 제약될 수 있는데, 예컨대 세그먼트가 액세스가능하지 않다.

[0540] 정확한 플레이-시간 tP에 액세스하기 위해서, 일단 인덱스 및 획득되는 상응하는 세그먼트의 URI는 실제 세그먼트 포맷에 의존한다. 본 예에서는, 미디어 세그먼트들이 일반성의 손실이 없이 0에서 시작하는 로컬 시간 라인을 갖는다고 가정된다. 플레이-시간 tP에 데이터를 액세스하여 제공하기 위해서, 클라이언트는 URI FileURI(r,i)(i=(r,t_p))를 통해 액세스될 수 있는 파일/세그먼트로부터 로컬 시간에 상응하는 데이터를 다운로딩할 수 있다.

[0541] 일반적으로, 클라이언트들은 전체 파일을 다운로딩할 수 있고, 이어서 플레이-시간 tP에 액세스할 수 있다. 그러나, 3GP 파일은 바이트 범위들에 로컬 타이밍을 맵핑하기 위한 구성을 제공하기 때문에, 3GP 파일의 모든 정보가 다운로딩될 필요는 없다. 그러므로, 충분한 랜덤 액세스 정보가 이용가능하게 되는 한, 단지 플레이-시간 tP에 대한 특정 바이트 범위들만이 미디어를 플레이하는데 충분할 수 있다. 또한, 바이트 범위의 구조 및 맵핑에 대한 충분한 정보 및 미디어 세그먼트의 로컬 타이밍이 예컨대 세그먼트 인덱스를 사용하여 세그먼트의 초기 부분에서 제공될 수 있다. 세그먼트의 초기 바이트들, 예컨대 1200 바이트들로 액세스함으로써, 클라이언트는 플레이 시간 tP를 위해 필요한 바이트 범위를 직접 액세스하기 위해서 충분한 정보를 가질 수 있다.

[0542] 추가의 예에 있어서는, 아래에서와 같이 "tidx" 박스로 어쩌면 명시된 세그먼트 인덱스가 요구되는 프래그먼트 또는 프래그먼트들의 바이트 오프셋들을 식별하기 위해 사용될 수 있다. 부분 GET 요청들이 요구되는 프래그먼트 또는 프래그먼트들에 대해 형성될 수 있다. 예컨대, 클라이언트가 파일에 대한 표준 요청을 발행하고 제 1 "tidx" 박스가 수신되었을 때는 이를 취소할 수 있는 다른 대안들이 존재한다.

[0543] 탐색

[0544] 클라이언트는 리프리젠테이션에 있어서 특정 리프리젠테이션 시간 tP을 탐색하려 시도할 수 있다. MPD에 기초하여, 클라이언트는 리프리젠테이션에서 각 세그먼트의 미디어 세그먼트 URL 및 미디어 세그먼트 시작 시간에 액세스한다. 클라이언트는 프리젠테이션 시간 tp에 대한 미디어 샘플들을 최대 세그먼트 인덱스 i로서 보유할 가능성이 가장 큰 세그먼트의 세그먼트 인덱스 segment_index를 획득할 수 있고, 그 경우에, 시작 시간 tS(r,i)은 프리젠테이션 시간 tp보다 작거나 그와 동일한데, 즉, $\text{segment\_index} = \max\{ i \mid tS(r,i) <= tp \}$ 이다. 세그먼트 URL은 FileURL(r,i)로서 획득된다.

[0545] MPD의 타이밍 정보는 랜덤 액세스 포인트들의 배치에 관련된 이슈들, 미디어 트랙들의 정렬 및 미디어 타이밍 드리프트로 인해 근사화될 수 있다는 것을 주시하자. 그 결과, 위의 절차에 의해 식별된 세그먼트는 tp보다 약간 이후의 시간에 시작할 수 있고, 프리젠테이션 시간 tp을 위한 미디어 데이터가 이전 미디어 세그먼트에 존재할 수 있다. 탐색의 경우에는, 탐색 시간이 리트리브된(retrieved) 파일의 제 1 샘플 시간과 동일하도록

업데이팅될 수 있거나, 선행 파일이 대신에 리트리브될 수 있다. 그러나, 대안적인 리프리젠테이션들/버전들 사이에서의 교환이 존재하는 경우들을 포함해서 연속적인 플레이아웃 동안에는, tp와 리트리브된 세그먼트의 시작 사이의 시간에 대한 미디어 데이터가 그럼에도 불구하고 이용가능하지 않다는 것을 주시하자.

[0546] 프리젠테이션 시간 tp을 정확히 탐색하기 위해서, HTTP 스트리밍 클라이언트는 랜덤 액세스 포인트(RAP)에 액세스할 필요가 있다. 3GPP Adaptive HTTP Streaming의 경우에 미디어 세그먼트에서의 랜덤 액세스 포인트를 결정하기 위해, 클라이언트는, 예컨대, 존재하는 경우 'tidx' 또는 'sidx' 박스 내의 정보를 사용함으로써 미디어 프리젠테이션에서 상응하는 프리젠테이션 시간 및 랜덤 액세스 포인트의 위치를 결정할 수 있다. 세그먼트가 3GPP 뮤비 프래그먼트인 경우들에서는, 클라이언트가 예컨대 'moof' 및 'mdat' 박스들 내의 정보를 사용함으로써, RAP들의 위치를 결정하고, 뮤비 프래그먼트의 정보로부터 필요한 프리젠테이션 시간 및 MPD로부터 유도되는 세그먼트 시작 시간을 획득하는 것이 또한 가능하다. 만약 요청된 프리젠테이션 시간 tp 이전의 프리젠테이션 시간을 갖는 어떤 RAP도 이용가능하지 않다면, 클라이언트는 이전 세그먼트를 액세스할 수 있거나 제 1 랜덤 액세스 포인트를 탐색 결과로서 단지 사용할 수 있다. 미디어 세그먼트가 RAP를 통해 시작할 때, 이러한 절차들은 간단하다.

[0547] 반드시 미디어 세그먼트의 모든 정보가 프리젠테이션 시간 tp을 액세스하기 위해 다운로딩될 필요가 없다는 것을 주시하자. 클라이언트는, 예컨대, 바이트 범위 요청들을 사용하여 미디어 세그먼트의 처음부터 'tidx' 또는 'sidx' 박스를 초기에 요청할 수 있다. 'tidx' 또는 'sidx' 박스들의 사용을 통해, 세그먼트 타이밍은 세그먼트의 바이트 범위들에 맵핑될 수 있다. 부분 HTTP 요청들을 계속해서 사용함으로써, 단지 미디어 세그먼트의 관련 부분들만이 개선된 사용자 경험 및 낮은 개시 지연들을 위해 액세스될 필요가 있다.

[0548] 세그먼트 리스트 생성

[0549] 여기서 설명된 바와 같이, 시그널링된 대략적인 세그먼트 지속시간 dur을 갖는 리프리젠테이션을 위한 세그먼트들의 리스트를 생성하기 위해 MPD에 의해서 제공되는 정보를 사용하는 간단한 HTTP 스트리밍 클라이언트를 어떻게 구현할지가 자명해야 한다. 일부 실시예들에 있어서, 클라이언트는 리프리젠테이션 연속 인덱스들 i=1, 2, 3,...을 미디어 세그먼트들에 할당될 수 있는데, 즉, 제 1 미디어 세그먼트에는 인덱스 i=1이 할당되고, 제 2 미디어 세그먼트에는 인덱스 i=2가 할당되며, 계속 이러한 방식을 따른다. 이어서, 예컨대 아래와 같이, 세그먼트 인덱스들 i을 갖는 미디어 세그먼트들의 리스트에는 startTime[i]가 할당되고, URL[i]가 생성된다. 먼저, 인덱스 i는 1로 설정된다. 제 1 미디어 세그먼트의 시작 시간은 0으로서 획득되는데, startTime[1]=0이다. 미디어 세그먼트 i의 URL(URL[i])는 FileURI(r,i)로서 획득된다. 인덱스 i를 갖는 모든 설명된 미디어 세그먼트들에 대해 프로세스가 계속되고, 미디어 세그먼트 i의 startTime[i]이 (i-1)*dur로서 획득되며, URL[i]가 FileURI(r,i)로서 획득된다.

[0550] 동시적인 HTTP/TCP 요청들

[0551] 블록-요청 스트리밍 시스템에서의 관심사는 플레이아웃을 위한 시간 내에 완전히 수신될 수 있는 가장 높은 품질의 블록들을 항상 요청하고자 하는 요구이다. 그러나, 데이터 도달 레이트가 미리 공지될 수 없고, 따라서 요청된 블록이 플레이아웃될 시간에 도달하지 못한다. 이는 미디어 플레이아웃을 중지시킬 필요성을 유발하며, 이는 나쁜 사용자 경험을 초래한다. 이러한 문제는, 심지어 데이터 도달 레이트가 블록의 수신 동안에 떨어지는 경우에도 적시에 수신될 가능성이 가장 많은 더 낮은 품질(그리고 따라서 더 낮은 사이즈)의 블록들을 요청함으로써 요청할 블록들의 선택에 대한 보수적인 해결책을 취하는 클라이언트 알고리즘에 의해서 완화된다. 그러나, 이러한 보수적인 해결책은 사용자 또는 목적지 디바이스에 더 낮은 품질의 플레이아웃을 어쩌면 전달하는 단점을 가지며, 이 또한 나쁜 사용자 경험이다. 그 문제는 아래에 설명되는 바와 같이 다수의 HTTP 접속들이 상이한 블록들을 다운로드하기 위해 동일한 시간에 사용될 때 증대되는데, 그 이유는 이용가능한 네트워크 자원들이 접속들에 걸쳐 공유되고 따라서 상이한 플레이아웃 시간들을 갖는 블록들을 위해 동시에 사용되고 있기 때문이다.

[0552] 클라이언트가 다수의 블록들에 대한 요청들을 동시에 발행하는 것이 유리할 수 있고, 이 문맥에서 "동시"는 요청들에 대한 응답들이 겹치는 시간 인터벌들에서 발생하는 것을 의미하며, 그것은 요청들이 정확히 동시에 또는 심지어 대략적으로 동시에 이루어지는 경우는 반드시 아니다. HTTP 프로토콜의 경우에, 이러한 해결책은 (널리 공지된 바와 같이) TCP 프로토콜의 작용으로 인한 이용가능한 대역폭의 활용을 향상시킬 수 있다.

이는, 새로운 컨텐트가 제일 먼저 요청될 때 블록들에 대한 데이터가 요청되게 하는 상응하는 HTTP/TCP 접속들이 느리게 시작할 수 있고 따라서 이 시점에 수 개의 HTTP/TCP 접속들을 사용하는 것은 제 1 블록들의 데이터 전달 시간을 극적으로 증가시킬 수 있기 때문에, 컨텐트 재핑(zapping) 시간을 향상시키는데 특히 중요할 수 있다. 그러나, 상이한 HTTP/TCP 접속들을 통해 상이한 블록들 또는 프래그먼트들을 요청하는 것은 또한 열화된 성능을 유도할 수 있는데, 그 이유는 제일 먼저 플레이아웃될 블록들에 대한 요청들이 후속 블록들에 대한 요청들과 경쟁하고, 경쟁하는 HTTP/TCP 다운로드들이 그들의 전달 속도에 있어 크게 변하며, 따라서 요청의 완료 시간이 매우 가변적일 수 있기 때문이고, HTTP/TCP 다운로드들이 어떤 HTTP/TCP 다운로드들이 완전히 빠르고 어떤 HTTP/TCP 다운로드들이 더 느릴 것인지를 제어하는 것은 일반적으로 불가능하며, 따라서 적어도 임의의 시간에 제 1의 몇몇 블록들의 HTTP/TCP 다운로드들이 완료를 위해 마지막이 되기 쉽고, 따라서 크고 가변적인 채널 재핑 시간들을 초래한다.

[0553] 세그먼트의 각 블록 또는 프래그먼트가 별도의 HTTP/TCP 접속을 통해 다운로드된다는 것, 병렬 접속들의 수는 n이고 각 블록의 플레이아웃 지속시간은 t초라는 것, 세그먼트와 연관된 컨텐트의 스트리밍 레이트가 S라는 것을 가정하자. 클라이언트가 제일 먼저 컨텐트를 스트리밍하기 시작할 때, 제 1 n 블록들에 대한 요청들이 발행되는데, 이는 미디어 데이터의 n*t초를 나타낸다.

[0554] 당업자들에게 알려진 바와 같이, TCP 접속들의 데이터 레이트에 있어서는 큰 변화가 존재한다. 그러나, 이러한 논의를 간략히 하기 위해서는, 모든 접속들이 병렬로 진행되고 있어서 제 1 블록이 요청된 다른 n-1 블록과 대략 동일한 시간에 완전히 수신될 것이라는 점을 이상적으로 가정하자. 논의를 더욱 간략히 하기 위해, n개의 다운로드 접속들에 의해 활용되는 통합 대역폭이 다운로드의 전체 지속시간에 대한 값 B으로 고정된다는 것 및 스트리밍 레이트 S가 전체 리프리젠테이션에 걸쳐 일정하다는 것을 가정하자. 또한, 전체 블록이 클라이언트에서 이용가능할 때 블록의 플레이아웃이 수행될 수 있도록(즉, 예컨대 기초 비디오 인코딩의 구조로 인해서 또는 각각의 프래그먼트 또는 블록을 개별적으로 암호화하기 위해 암호화가 이용되고 있기 때문에, 전체 블록이 수신된 이후에나 블록의 플레이아웃이 단지 시작할 수 있음) 미디어 데이터 구조가 이루어지고, 따라서 전체 프래그먼트 또는 블록이 그것이 암호화되기 이전에 수신될 필요가 있다는 것을 가정하자. 따라서, 아래의 논의를 간략히 하기 위해, 우리는 블록 중 임의의 블록이 플레이아웃될 수 있기 이전에 전체 블록이 수신될 필요가 있다는 것을 가정한다. 이어서, 제 1 블록이 도달하고 플레이아웃될 수 있기 이전의 필요한 시간은 대략 n*t*S/B이다.

[0555] 컨텐트 재핑 시간을 최소화하는 것이 바람직하기 때문에, 따라서 n*t*S/B를 최소화하는 것이 바람직하다. t의 값은 기초 비디오 인코딩 구조 및 수집 방법들이 어떻게 활용되는지와 같은 팩터들에 의해 결정될 수 있고, 따라서 t는 적절히 작을 수 있지만, 매우 작은 t의 값들은 과도하게 복잡한 세그먼트 맵을 유도하며 어쩌면 사용될 경우 효율적인 비디오 인코딩 및 암호해독과 비호환적일 수 있다. n의 값은 또한 B의 값에 영향을 줄 수 있는데, 즉, B는 매우 많은 수(n)의 접속들에 대해 클 수 있고, 따라서 접속들의 수 n을 감소시키는 것은 활용되는 이용가능한 대역폭의 양 B을 잠재적으로 감소시키는 부정적인 측면의 영향을 가지며, 따라서 컨텐트 재핑 시간을 감소시키고자 하는 목적을 달성하는데 효과적이지 않을 수 있다. S의 값은 어떤 리프리젠테이션이 다운로드 및 플레이아웃을 위해 선택되는지에 의존하며, 이상적으로 S는 정해진 네트워크 조건들에 대한 미디어의 플레이아웃 품질을 최대화시키기 위해서 가능한 B에 가깝게 되어야 한다. 따라서, 이러한 논의를 간략히 하기 위해서, S가 대략적으로 B와 같다고 가정하자. 이어서, 채널 재핑 시간은 n*t에 비례한다. 따라서, 상이한 프래그먼트들을 다운로딩하기 위해 더 많은 접속들을 활용하는 것은, 만약 그 접속들에 의해 활용되는 통합 대역폭이 접속들의 수에 준-선형적으로 비례한다면, 채널 재핑 시간을 떨어뜨릴 수 있다는 것이 통상적으로 사실이다.

[0556] 일예로서, t=1초이라고 가정하면, n=1인 경우에 B의 값은 500 Kbps이고, n=2인 경우에 B의 값은 700 Kbps이고, n=3인 경우에 B의 값은 800 Kbps이다. S=700 Kbps를 갖는 리프리젠테이션이 선택된다고 가정하자. 이어서, n=1인 경우에 제 1 블록에 대한 다운로드 시간은 1*700/500=1.4초이고, n=2인 경우에 제 1 블록에 대한 다운로드 시간은 2*700/700=2초이며, n=3인 경우에 제 1 블록에 대한 다운로드 시간은 3*700/800=2.625초이다. 게다가, 접속들의 수가 증가함에 따라, 접속의 개별적인 다운로드 속도들에서의 가변성은 증가하기 쉽다(비록, 심지어 하나의 접속에 있어서 어느 정도의 상당한 가변성이 존재하기 쉽겠지만). 따라서, 본 예에서, 채널 재핑 시간 및 채널 재핑 시간의 가변성은 접속들의 수가 증가함에 따라 증가한다. 직관적으로, 전달되고 있는 블록들은 상이한 우선순위들을 갖는데, 즉, 제 1 블록이 가장 이른 전달 데드라인을 갖고, 제 2 블록이 두 번째로 이른 전달 데드라인을 가지며, 계속해서 이러한 방식을 갖고, 그에 반해 블록들이 전달되고 있는 다운로드 접속들은 그 전달 동안에 네트워크 자원들에 대해 경쟁하며, 따라서 가장 이른 데드라인들을 갖는 블록들은 더 많은 경쟁 블록들이 요청됨에 따라 더욱 지연되게 된다. 다른 한편으로는, 심지어 이러한

경우에, 궁극적으로 하나보다 많은 수의 다운로드 접속을 사용하는 것은 실질적으로 더 높은 스트리밍 레이트의 지원을 허용하는데, 예컨대 3개의 접속들의 경우에는 최대 800 Kbps까지의 스트리밍 레이트가 본 예에서 지원될 수 있는데 반해, 한 개의 접속의 경우에는 단지 500 Kbps의 스트림만이 지원될 수 있다.

[0557] 실제로는, 위에서 설명된 주시된 바와 같이, 접속의 데이터 레이트는 시간에 걸쳐 동일한 접속 내에서 그리고 접속들 간에 매우 가변적일 수 있고, 그 결과, n개의 요청된 블록들이 일반적으로 동일한 시간에 완료하지 않으며, 사실상, 한 블록이 다른 블록의 시간의 절반 내에 완료할 수 있는 것이 일반적인 경우일 수 있다. 이러한 효과는 일부 경우들에 있어서는 제 1 블록이 다른 블록들보다 훨씬 빨리 완료할 수 있고 다른 경우들에 있어서는 제 1 블록이 다른 블록들보다 훨씬 늦게 완료할 수 있기 때문에 비-예측가능한 작용을 초래하고, 그 결과, 플레이아웃의 개시가 일부 경우들에 있어서는 비교적 빠르게 발생하고 다른 경우들에 있어서는 느리게 발생할 수 있다. 이러한 비-예측가능한 작용은 사용자에게 실망스러울 수 있고, 따라서 나쁜 사용자 경험으로 간주될 수 있다.

[0558] 따라서, 요구되는 것은, 다수의 TCP 접속들이 채널 재핑 시간 및 채널 재핑 시간의 가변성을 향상시키기 위해 활용될 수 있으면서 동시에 가능한 양호한 품질 스트리밍 레이트를 지원할 수 있는 방법들이다. 또한 요구되는 것은, 블록의 플레이아웃 시간에 도달할 때 조정될 각 블록에 할당된 이용가능한 대역폭의 공유를 허용함으로써 필요할 경우 이용가능한 대역폭의 더 큰 공유가 가장 가까운 플레이아웃 시간을 갖는 블록에 할당될 수 있게 하는 방법들이다.

[0559] 협력적인 HTTP/TCP 요청

[0560] 우리는 협력적인 형태로 동시적인 HTTP/TCP 요청들을 사용하기 위한 방법들을 이제 설명한다. 수신기는 예컨대 다수의 HTTP 바이트-범위 요청들을 사용하여 다수의 동시 협력적인 HTTP/TCP 요청들을 이용할 수 있는데, 여기서 각각의 이러한 요청은 소스 세그먼트에서 프래그먼트의 일부에 대한 것이거나, 소스 세그먼트의 프래그먼트 모두에 대한 것이거나, 보수 세그먼트의 보수 프래그먼트 모두에 대한 것이다.

[0561] FEC 보수 데이터의 사용과 더불어 협력적인 HTTP/TCP 요청의 장점은 일관적으로 빠른 채널 재핑 시간들을 제공하는데 특별히 중요할 수 있다. 예컨대, 채널 재핑 시간에는, TCP 접속들이 막 시작되었거나 또는 어느 정도의 시간 기간 동안 유휴 상태였기 쉽고, 그 경우에는 수집 윈도우 cwnd가 접속들을 위해 그것의 최소 값에 있으며, 따라서 이러한 TCP 접속들의 전달은 램프업하기 위해서 수 개의 라운드-트립 시간들(RTT들)이 걸릴 것이고, 이러한 램프-업 시간 동안에는 상이한 TCP 접속들을 통한 전달 속도들에 있어 높은 가변성이 존재할 것이다.

[0562] 협력적인 HTTP/TCP 요청 방법인 비-FEC 방법의 개요가 이제 설명되는데, 그 방법에서는 단지 소스 블록들의 미디어 데이터만이 다수의 동시적인 HTTP/TCP 접속들을 사용하여 요청되는데, 즉, 어떠한 FEC 보수 데이터도 요청되지 않는다. 비-FEC 방법을 통해, 동일 프래그먼트의 부분들이 예컨대 그 프래그먼트의 부분들에 대한 HTTP 바이트 범위 요청들을 사용하여 상이한 접속들을 통해 요청되고, 따라서 예컨대 각각의 HTTP 바이트 범위 요청은 프래그먼트의 세그먼트 맵에 표시된 바이트 범위의 부분에 대한 것이다. 개별적인 HTTP/TCP 요청이 수 개의 RTT들(round-trip times)에 걸쳐 이용가능한 대역폭을 충분히 활용하기 위해 그것의 전달 속도를 램프업하고, 그로 인해서 전달 속도가 이용가능한 대역폭 미만인 경우에는 비교적 긴 시간 기간이 존재하고, 따라서 만약 단일 HTTP/TCP 접속이 예컨대 플레이아웃될 컨텐트의 제 1 프래그먼트를 다운로드하기 위해 사용된다면, 채널 재핑 시간이 클 수 있다는 것이 사실일 수 있다. 비-FEC 방법을 사용함으로써, 상이한 HTTP/TCP 접속들을 통해 동일 프래그먼트의 상이한 부분들을 다운로딩하는 것은 채널 재핑 시간을 상당히 감소시킬 수 있다.

[0563] 협력적인 HTTP/TCP 요청 방법인 FEC 방법의 개요가 이제 설명되는데, 그 방법에서는 소스 세그먼트의 미디어 데이터 및 그 미디어 데이터로부터 생성되는 FEC 보수 데이터가 다수의 동시적인 HTTP/TCP 접속들을 사용하여 요청된다. FEC 방법을 통해, 동일 프래그먼트의 부분들 및 그 프래그먼트로부터 생성된 FEC 보수 데이터가 그 프래그먼트의 부분들에 대한 HTTP 바이트 범위 요청들을 사용하여 상이한 접속들을 통해 요청되고, 따라서 예컨대 각각의 HTTP 바이트 범위 요청은 그 프래그먼트에 대한 세그먼트 맵에 표시된 바이트 범위의 부분에 대한 것이다. 개별적인 HTTP/TCP 요청이 수 개의 RTT들(round-trip times)에 걸쳐 이용가능한 대역폭을 충분히 활용하기 위해 그것의 전달 속도를 램프업하고, 그로 인해서 전달 속도가 이용가능한 대역폭 미만인 경우에는 비교적 긴 시간 기간이 존재하고, 따라서 만약 단일 HTTP/TCP 접속이 예컨대 플레이아웃될 컨텐트의 제 1 프래그먼트를 다운로드하기 위해 사용된다면, 채널 재핑 시간이 클 수 있다는 것이 사실일 수 있다. FEC 방

법을 사용하는 것은 비-FEC 방법과 동일한 장점들을 갖고, 또한 프래그먼트가 복원될 수 있기 이전에 상기 요청된 데이터 모두가 도달할 필요는 없고 따라서 채널 재핑 시간 및 그 채널 재핑 시간의 가변성을 더욱 감소시킨다는 추가적인 장점을 갖는다. 상이한 TCP 접속들에 걸쳐 요청들을 수행하고, 그 접속들 중 적어도 하나를 통해 FEC 보수 데이터를 또한 요청하여 오버-요청함으로써, 예컨대 미디어 플레이백이 시작될 수 있게 하는 제 1 요청된 프래그먼트를 복원하기 위해 충분한 양의 데이터를 전달하는데 걸리는 시간의 양이 협력적인 TCP 접속들 및 FEC 보수 데이터가 사용되지 않은 경우보다 크게 감소되고 훨씬 더 일관적이게 이루어질 수 있다.

[0564] 도 24의 (a) 내지 (e)는 에뮬레이티드 EVDO(evolution data optimized) 네트워크의 동일 HTTP 웹 서버로부터 동일 클라이언트로의 동일 링크 상에서 운행되는 5개의 TCP 접속들의 전달 레이트 변동들에 대한 예를 나타낸다. 도 24의 (a) 내지 (e)에서, X-축은 초로 시간을 나타내고, Y-축은 각각의 접속에 대해서 1초의 인터벌들에 걸쳐 측정되는 5개의 TCP 접속들 각각을 통해 클라이언트에서 비트들이 수신되는 레이트를 나타낸다. 이러한 특정 에뮬레이션에서는, 이러한 링크를 통해서 모두 실행되는 12개의 TCP 접속들이 존재하였고, 따라서 네트워크는 도시된 시간 동안에 상대적으로 부하가 발생하였는데, 이는 하나보다 많은 클라이언트가 이동 네트워크의 동일 셀 내에서 스트리밍할 때 통상적일 수 있다. 비록 전달 레이트들이 시간에 걸쳐 다소 상호관련되지만, 많은 시점들에서 5개의 접속들의 전달 레이트에 있어 큰 차이가 존재함을 주시하자.

[0565] 도 25는 크기가 250,000인(대략적으로 31.25 킬로바이트들) 프래그먼트에 대한 가능한 요청 구조를 나타내고, 여기서 프래그먼트의 상이한 부분들에 대해 병렬로 수행되는 4개의 HTTP 바이트 범위 요청들이 존재하는데, 즉, 제 1 HTTP 접속은 첫 번째 50,000 비트들을 요청하고, 제 2 HTTP 접속은 그 다음 50,000 비트들을 요청하고, 제 3 HTTP 접속은 그 다음 50,000 비트들을 요청하며, 제 4 HTTP 접속은 그 다음 50,000 비트들을 요청한다. 만약 FEC가 사용되지 않는다면, 즉, 비-FEC가 수행된다면, 이들은 본 예에서 프래그먼트에 대한 단지 4개의 요청들이다. 만약 FEC가 사용된다면, 즉, FEC 방법이 사용된다면, 본 예에서는 프래그먼트로부터 생성되는 보수 세그먼트의 FEC 보수 데이터의 추가적인 50,000 비트들을 요청하는 하나의 추가적인 HTTP 접속이 존재한다.

[0566] 도 26은 도 24의 (a) 내지 (e)에 도시된 5개의 TCP 접속들의 초들의 제 1 연결의 확대도인데, 도 26에서는 X-축은 100 밀리초의 인터벌들인 시간을 나타내고, Y-축은 100 밀리초의 인터벌들에 걸쳐 측정되는 5개의 TCP 접속들 각각을 통해 클라이언트에서 비트들이 수신되는 레이트를 나타낸다. 하나의 라인은 제 1 4개의 HTTP 접속들(FEC 데이터가 요청되는 HTTP 접속을 포함함)로부터 프래그먼트, 즉, 비-FEC 방법을 사용하는 도달하는 프래그먼트에 대해 클라이언트에서 수신되어진 비트들의 통합 양을 나타낸다. 다른 라인은 모두 5개의 HTTP 접속들(FEC 데이터가 요청되는 HTTP 접속을 포함함)로부터 프래그먼트, 즉, FEC 방법을 사용하는 도달하는 프래그먼트에 대해 클라이언트에서 수신되어진 비트들의 통합 양을 나타낸다. FEC 방법의 경우에는, 프래그먼트가 250,000개의 요청된 비트들 중 임의의 200,000개의 비트들의 수신으로부터 FEC 디코딩될 수 있고, 이는 예컨대 리드-솔로몬 FEC 코드가 사용되는 경우에 구현될 수 있고, 본질적으로는 예컨대 Luby Ⅳ에 설명된 RaptorQ 코드가 사용되는 경우에 구현될 수 있다. 본 예에서 FEC 방법의 경우에는, 충분한 데이터가 1초 이후에 FEC 디코딩을 사용하여 프래그먼트를 복원하기 위해 수신되어, 1초의 채널 재핑 시간을 허용한다(후속 프래그먼트에 대한 데이터가 제 1 프래그먼트가 충분히 플레이아웃되기 이전에 요청되어 수신될 수 있다는 가정함). 본 예에서 비-FEC 방법의 경우에는, 4개의 요청들에 대한 모든 데이터가 프래그먼트가 복원될 수 있기 이전에 수신되어야 하고, 이는 1.7초 이후에 발생하여 1.7초의 채널 재핑 시간을 유도한다. 따라서, 도 26에 도시된 예에서, 비-FEC 방법은 FEC 방법보다 채널 재핑 시간에 있어 70% 더 나쁘다. 본 예에서 FEC 방법에 의해 제시된 장점들에 대한 이유들 중 하나는, FEC 방법의 경우에 요청된 데이터의 임의의 80%의 수신이 프래그먼트의 복원을 허용하는데 반해 비-FEC 방법의 경우에 요청된 데이터의 100%의 수신이 필요하기 때문이다. 따라서, 비-FEC 방법은 전달을 종료하기 위해서 가장 느린 TCP 접속을 기다려야 하고, 또한 TCP 전달 레이트에 있어서의 자연스런 변화들로 인해서 평균 TCP 접속에 비해 그 가장 느린 TCP 접속의 전달 속도의 큰 변화가 있기 쉽다. 본 예에서 FEC 방법을 통해, 하나의 느린 TCP 접속은 프래그먼트가 복원가능할 때를 결정하지 못한다. 대신에, FEC 방법의 경우에는 충분한 데이터의 전달이 더 나쁜 경우의 TCP 전달 레이트보다 평균 TCP 전달 레이트의 훨씬 큰 함수이다.

[0567] 위에서 설명된 비-FEC 방법 및 FEC 방법의 많은 변형들이 존재한다. 예컨대, 협력적인 HTTP/TCP 요청들은 채널 잽(zap)이 발생한 이후에 첫 번째 수 개의 프래그먼트들만을 위해 사용될 수 있고, 그런 이후에는 단일 HTTP/TCP 요청만이 추가적인 프래그먼트들, 다수의 프래그먼트들, 또는 전체 세그먼트들을 다운로딩하기 위해 사용된다. 다른 예로서, 사용되는 협력적인 HTTP/TCP 접속들의 수는 요청되고 있는 프래그먼트들의 긴급성, 즉, 이러한 프래그먼트의 플레이아웃 시간이 얼마나 급박한지, 및 현재 네트워크 조건들의 긴급성 양쪽 모두

의 함수일 수 있다.

[0568] 일부 변환들에 있어서, 다수의 HTTP 접속들은 보수 세그먼트들로부터의 보수 데이터를 요청하기 위해 사용될 수 있다. 다른 변형들에 있어서는, 예컨대 클라이언트에서 데이터 수신의 레이트 및 미디어 버퍼의 현재 크기에 따라 상이한 양의 데이터가 상이한 HTTP 접속들을 통해 요청될 수 있다. 다른 변형들에 있어서는, 소스 리프리젠테이션들이 서로 독립적이지만 대신에 계층화된 미디어 코딩을 나타내고, 여기서는 예컨대 개선된 소스 리프리젠테이션이 기본 소스 리프리젠테이션에 기초할 수 있다. 이러한 경우에는, 기본 소스 리프리젠테이션에 상응하는 보수 리프리젠테이션, 및 기본 개선 소스 리프리젠테이션들의 조합에 상응하는 다른 보수 리프리젠테이션이 존재할 수 있다.

[0569] 위에서 설명된 방법들에 의해 구현할 수 있는 장점들에 추가적인 전체 엘리먼트들이 추가된다. 예컨대, 사용되는 HTTP 접속들의 수는 미디어 버퍼에 있는 미디어의 현재 양 및/또는 미디어 버퍼로의 수신 레이트에 따라 변할 수 있다. FEC, 즉, 위에서 설명된 FEC 방법 및 그 방법의 변형들을 사용하는 협력적인 HTTP 요청들이 미디어 버퍼가 비교적 비어 있을 때 적극적으로 사용될 수 있는데, 예컨대 더 많은 협력적인 HTTP 요청들이 제 1 프래그먼트의 상이한 부분들에 대해 병렬로 수행되어, 상응하는 보수 프래그먼트로부터의 보수 데이터의 비교적 큰 프랙션 및 소스 프래그먼트의 모두를 요청하고, 이어서 동시적인 HTTP 요청들의 감소된 수로 전환하며, 요청마다의 미디어 데이터의 더 큰 부분을 요청하고, 보수 데이터의 더 작은 프랙션을 요청하는데, 예컨대 1, 2 또는 3개의 동시적인 HTTP 요청들로 전환하고, 요청마다 최대 프래그먼트 또는 다수의 연속적인 프래그먼트들에 대한 요청을 수행하며, 미디어 버퍼가 증대됨에 따라 어떠한 보수 데이터도 요청하도록 전환하지 않는다.

[0570] 다른 예로서, FEC 보수 데이터의 양은 미디어 버퍼 사이즈의 함수로서 변할 수 있는데, 즉, 미디어 버퍼가 작을 때는 더 많은 FEC 보수 데이터가 요청될 수 있고, 미디어 버퍼가 증대됨에 따라 요청되는 FEC 보수 데이터의 양이 감소할 수 있으며, 임의의 수간에 미디어 버퍼가 충분히 클 때는 어떠한 FEC 보수 데이터도 요청될 수 없고 단지 소스 리프리젠테이션의 소스 세그먼트들로부터의 데이터만이 요청될 수 있다. 이러한 개선된 기술들의 이점들은, 그것들이 더 빠르고 더 일관적인 채널 재핑 시간들 및 잠재적인 미디어 셔터들 또는 스톨들에 대해 더 큰 탄력성을 허용할 수 있으면서 동시에 요청 메시지 트래픽 및 FEC 보수 데이터 양쪽 모두를 감소시킴으로써 소스 세그먼트들로 단지 미디어를 전달하여 소모될 양을 넘어 사용되는 추가적인 데이터의 양을 최소화하는 동시에 정해진 네트워크 조건들에 대해 가능한 가장 높은 미디어 레이트들의 지원을 가능하게 한다는 것이다.

[0571] 동시적인 HTTP 접속들을 사용할 때 추가적인 개선들

[0572] HTTP/TCP 요청은, 만약 적절한 조건이 충족되고 다른 HTTP/TCP 요청이 포기된 요청을 통해 요청된 데이터를 대체할 수 있는 데이터를 다운로드하기 위해 수행될 수 있다면, 포기될 수 있는데, 제 2 HTTP/TCP 요청은 본래 요청에서와 정확히 동일한 데이터, 예컨대, 소스 데이터; 예컨대 제 1 요청을 통해 요청되지 않은 동일한 소스 데이터 및 보수 데이터 중 일부와 같은 겹치는 데이터; 또는 예컨대 제 1 요청을 통해 요청되지 않은 보수 데이터와 같은 완전히 해체된 데이터를 요청할 수 있다. 적절한 조건의 예는, 제공된 시간 내에 블록 서버 인프라구조(BSI)로부터의 응답의 부재로 인해서 실패하거나 또는 BSI로의 전송 접속의 설정 또는 서버 또는 서버로부터 명시적인 실패 메시지의 수신으로 인해서 실패하거나 또는 다른 실패 조건으로 인해서 요청이 실패하는 것이다.

[0573] 적절한 조건의 다른 예는, 거기에 포함된 미디어 데이터의 플레이아웃 시간 이전에 또는 그 시간에 의존적인 다른 시간에 응답을 수신하기 위해서 필요한 접속 속도의 추정치 또는 예상된 접속 속도와 접속 속도의 측정치(문제의 요청에 응하는 데이터 도달 레이트)의 비교에 따라, 데이터의 수신이 일반적으로 느리게 진행하는 것이다.

[0574] 이러한 해결책은 BSI가 종종 실패 또는 나쁜 성능을 나타내는 경우에 장점을 갖는다. 이 경우에, 위의 해결책은 클라이언트가 BSI 내에서의 실패 또는 나쁜 성능에도 불구하고 미디어 데이터의 신뢰적인 플레이아웃을 계속할 수 있는 확률을 증가시킨다. 일부 경우들에서는, BSI가 때에 따라서 이러한 실패 또는 나쁜 성능을 나타내는 방식으로 그 BSI를 설계하는 것이 유리할 수 있는데, 예컨대 이러한 설계는 이러한 실패들 또는 나쁜 성능을 나타내지 않거나 덜 자주 이러한 것들을 나타내는 대안적인 설계보다 더 낮은 비용을 가질 수 있다는 것을 주시하자. 이 경우에, 여기서 설명된 방법은 사용자 경험의 결과적인 저하가 없이 BSI에 대해 이러한 더 낮은 비용 설계의 활용을 허용하는 추가적인 장점을 갖는다.

[0575] 다른 실시예에서, 정해진 블록에 상응하는 데이터에 대해 발행되는 요청들의 수는 블록에 대한 적절한 조건이 충족되는지 여부에 의존적일 수 있다. 만약 그 조건이 충족되지 않는다면, 블록에 대한 모든 현재 불완전한 데이터 요청들의 성공적인 완료가 높은 확률로 블록의 복원을 허용할 경우에 블록에 대한 추가 요청들을 클라이언트가 수행하는 것이 제약될 수 있다. 만약 그 조건이 충족된다면, 블록에 대한 매우 많은 수의 요청들이 발행될 수 있는데, 즉, 위의 제약이 적용되지 않는다. 적절한 조건의 예로, 블록의 스케줄링된 플레이아웃 시간까지의 시간 또는 그 시간에 의존하는 다른 시간이 제공된 임계치 아래로 떨어진다. 이러한 방법은 블록의 수신이 더욱 긴급할 때는 그 블록을 포함하는 미디어 데이터의 플레이아웃 시간이 클로즈되기 때문에 그 블록에 대한 데이터에 대한 추가적인 요청이 발생되는 장점을 갖는다. HTTP/TCP와 같은 공통 전송 프로토콜의 경우에, 이러한 추가적인 요청들은 문제의 블록의 수신에 기여하는 데이터에 전용화된 이용가능한 대역폭의 공유를 증가시키는 효과를 갖는다. 이는 완료할 블록을 복원하기 위해 충분한 데이터의 수신에 필요한 시간을 감소시키고, 그로 인해서 블록이 그 블록을 포함하는 미디어 데이터의 스케줄링된 플레이아웃 시간 이전에 복원될 수 없을 확률을 감소시킨다. 위에서 설명된 바와 같이, 만약 블록을 포함하는 미디어 데이터의 스케줄링된 플레이아웃 시간 이전에 그 블록이 복원될 수 없다면, 플레이아웃은 중지되어 나쁜 사용자 경험을 유발할 수 있고 그로 인해 여기서 설명된 방법은 이러한 나쁜 사용자 경험의 확률을 유리하게 감소시킨다.

[0576] 본 명세서에 전체에 걸쳐, 블록의 스케줄링된 플레이아웃 시간에 대한 참조들은 그 블록을 포함하는 인코딩된 미디어 데이터가 중지 없이 프리젠테이션의 플레이아웃을 달성하기 위해 클라이언트에서 제일 먼저 이용가능할 수 있는 시간을 지칭함을 이해해야 한다. 미디어 프리젠테이션 시스템들의 분야의 당업자들에게 자명할 바와 같이, 이러한 시간은 사실상 플레이아웃을 위해 사용되는 물리적인 트랜스듀서들(스크린, 스피커 등)에서 블록을 포함하는 미디어를 나타낼 실제 시간의 약간 이전인데, 그 이유는 수 개의 변환 함수들이 블록의 실제 플레이아웃을 실행하기 위해 그 블록을 포함하는 미디어 데이터에 적용될 필요가 있을 수 있고 또한 이러한 함수들은 완료하기 위해 특정 양의 시간을 필요로 할 수 있기 때문이다. 예컨대, 미디어 데이터는 일반적으로 압축된 형태로 전달되고, 압축해제 변환이 적용될 수 있다.

[0577] 협력적인 HTTP/FEC 방법들을 지원하는 파일 구조들을 생성하기 위한 방법들

[0578] 협력적인 HTTP/FEC 방법을 이용하는 클라이언트에 의해 유리하게 사용될 수 있는 파일 구조를 생성하는 실시예가 이제 설명된다. 본 실시예에서는, 각각의 소스 세그먼트에 대해, 아래와 같이 생성되는 상응하는 보수 세그먼트가 존재한다. 파라미터 R은 얼마나 많은 FEC 보수 데이터가 소스 세그먼트들의 소스 데이터를 위해 생성되는지를 대체로 나타낸다. 예를 들어, R=0.33은, 소스 세그먼트가 1,000 킬로바이트들의 데이터를 포함하는 경우 상응하는 보수 세그먼트는 대략 330 킬로바이트들의 보수 데이터를 포함함을, 나타낸다. 파라미터 S는 FEC 인코딩 및 디코딩을 위해 사용되는 심볼 크기를 바이트들로 나타낸다. 예컨대, S=64는 소스 데이터 및 보수 데이터가 FEC 인코딩 및 디코딩의 목적들을 위해 64 바이트들 크기의 심볼들을 각각 포함한다.

[0579] 보수 세그먼트가 아래와 같이 소스 세그먼트에 대해 생성될 수 있다. 소스 세그먼트의 각각의 프래그먼트는 FEC 인코딩 목적을 위해 소스 블록으로서 간주되고, 따라서 각각의 프래그먼트는 보수 심볼들이 생성되는 소스 블록의 소스 심볼들의 시퀀스로서 처리된다. 제 1 i 프래그먼트들을 위해 전부 생성되는 보수 심볼들의 수는 TNRS(i)=ceiling(R*B(i)/S)로서 계산되며, 여기서 ceiling(x)는 적어도 x인 값을 갖는 가장 작은 정수를 출력하는 함수이다. 따라서, 프래그먼트 i를 위해 생성된 보수 심볼들의 수는 NRS(i)=TNRS(i)-TNRS(i-1)이다.

[0580] 보수 세그먼트는 프래그먼트들에 대한 보수 심볼들의 연쇄를 포함하는데, 여기서 보수 세그먼트 내의 보수 심볼들의 순서는 그들이 생성되는 프래그먼트들의 순서를 따르며, 프래그먼트 내에서 보수 심볼들은 그들의 인코딩 심볼 식별자(ESI)의 순서를 따른다. 소스 세그먼트 구조에 상응하는 보수 세그먼트 구조는 보수 세그먼트 생성기(2700)를 포함하는 도 27에 도시되어 있다.

[0581] 위에서 설명된 프래그먼트들에 대한 보수 심볼들의 수를 정의함으로써, 모든 이전 프래그먼트들에 대한 보수 심볼들의 총 수 및 그에 따른 보수 세그먼트 내의 바이트 인덱스는 단지 R, S, B(i-1) 및 B(i)에 의존하며, 소스 세그먼트 내의 프래그먼트들의 이전 또는 후속 구조 중 어느 것에도 의존하지 않는다는 것을 주시하자. 이는 클라이언트로 하여금 보수 블록이 생성되는 소스 세그먼트의 상응하는 프래그먼트의 구조에 대한 국부적인 정보만을 사용하여 보수 세그먼트 내의 보수 블록의 시작 위치를 신속히 계산하고 또한 그 보수 블록 내의 보수 심볼들의 수를 신속히 계산하도록 허용하기 때문에 유리하다. 따라서, 만약 클라이언트가 소스 세그먼트의 중간부터 프래그먼트의 다운로딩 및 플레이아웃을 시작하기로 결정한다면, 클라이언트는 또한 상응하는

보수 세그먼트 내로부터 상응하는 보수 블록을 신속하게 생성하고 액세스할 수 있다.

[0582] 프래그먼트 i에 상응하는 소스 블록 내의 소스 심볼들의 수는 NSS(i)=ceiling((B(i)-B(i-1))/S)로서 계산된다. 만약 B(i)-B(i-1)이 S의 배수가 아니라면 마지막 소스 심볼이 FEC 인코딩 및 디코딩의 목적들을 위해서 제로 바이트들로 패딩아웃되는데, 즉, 마지막 소스 심볼이 제로 바이트들로 패딩아웃되고, 그럼으로써 그것은 FEC 인코딩 및 디코딩의 목적들을 위해 크기에 있어 S 바이트들이지만 이러한 제로 패딩 바이트들은 소스 세그먼트의 일부로서 저장되지 않는다. 이러한 실시예에 있어서, 소스 심볼에 대한 ESI들은 0, 1,..., NSS(i-1)이고, 보수 심볼에 대한 ESI들은 NSS(i),..., NSS(i)+NRS(i)-1이다.

[0583] 본 실시예에서 보수 세그먼트에 대한 URL은 예컨대 접미사 ".repair"를 상응하는 소스 세그먼트의 URL에 간단히 추가함으로써 그 소스 세그먼트에 대한 URL로부터 생성될 수 있다.

[0584] 보수 세그먼트에 대한 보수 인덱싱 정보 및 FEC 정보는 여기서 설명된 바와 같이, 상응하는 소스 세그먼트에 대한 인덱싱 정보에 의해서 그리고 R 및 S의 값들로부터 묵시적으로 정의된다. 보수 세그먼트를 포함하는 프래그먼트 구조 및 시간 오프셋들이 상응하는 소스 세그먼트의 시간 오프셋들 및 구조에 의해 결정된다. 프래그먼트 i에 상응하는 보수 세그먼트내 보수 심볼들의 마지막에 대한 바이트 오프셋은 RB(i)=S*ceiling(R*B(i)/S)로서 계산될 수 있다. 이어서, 프래그먼트 i에 상응하는 보수 세그먼트내 바이트들의 수가 RB(i)-RB(i-1)이고, 따라서 프래그먼트 i에 상응하는 보수 심볼들의 수는 NRS(i)=(RB(i)-RB(i-1))/S로서 계산될 수 있다. 프래그먼트 i에 상응하는 소스 심볼들의 수는 NSS(i)=ceiling((B(i)-B(i-1))/S)로서 계산될 수 있다. 따라서, 본 실시예에서, 복수 세그먼트내 보수 블록에 대한 보수 인덱싱 정보 및 상응하는 FEC 정보가 상응하는 소스 세그먼트의 상응하는 프래그먼트에 대한 인덱싱 정보, R 및 S로부터 묵시적으로 유도될 수 있다.

[0585] 일예로서, 도 28에 도시된 예가 바이트 오프셋 B(1)=6,410에서 시작하고 바이트 오프셋 B(2)=6,770에서 종료하는 프래그먼트 2를 나타낸다고 간주하자. 본 예에서, 심볼 사이즈는 S=64 바이트들이고, 점선 수직 라인들은 S의 배수들에 상응하는 소스 세그먼트 내의 바이트 오프셋들을 나타낸다. 소스 세그먼트 사이즈의 프랙션으로서의 전체 보수 세그먼트 사이즈는 본 예에서 R=0.5로 설정된다. 프래그먼트 2에 대한 소스 블록내 소스 블록들의 수는 $\mathrm{NSS}(2) = \mathrm{ceiling}((6{,}770-6{,}410)/64) = \mathrm{ceil}(5.625) = 6$ 로서 계산되고, 이러한 6개의 소스 심볼들은 ESI들 0,..., 5를 각각 갖는데, 여기서 제 1 소스 심볼은 소스 세그먼트내 바이트 인덱스 6,410에서 시작하는 프래그먼트 2의 첫번째 64 바이트들이고, 제 2 소스 심볼은 소스 세그먼트내 바이트 인덱스 6,474에서 시작하는 프래그먼트 2의 그 다음 64 바이트들이며, 기타 등등이다. 프래그먼트 2에 상응하는 보수 블록의 마지막 바이트 오프셋은 $\mathrm{RB}(2) = 64*\mathrm{ceiling}(0.5*6{,}770/64) = 64*\mathrm{ceiling}(52.89\ldots) = 64*53 = 3{,}392$ 로서 계산되고, 프래그먼트 2에 상응하는 보수 블록의 시작 바이트 오프셋은 $\mathrm{RB}(1) = 64*\mathrm{ceiling}(0.5*6{,}410/64) = 64*\mathrm{ceiling}(50.07\ldots) = 64*51 = 3{,}264$ 로서 계산되며, 따라서 본 예에서는 보수 세그먼트내 바이트 오프셋 3,264에서 시작하고 바이트 오프셋 3,392에서 종료하는, ESI들 6 및 7을 갖는 프래그먼트 2에 상응하는 보수 블록내 2개의 보수 심볼들이 존재한다.

[0586] 도 28에 도시된 예에서는, 비록 R=0.5이고 프래그먼트 2에 상응하는 6개의 소스 심볼들이 존재할지라도, 보수 심볼들의 수를 계산하기 위해 소스 심볼들의 수를 간단히 사용하였지만 대신에 여기서 설명된 방법들에 따라 2인 것으로 시험되는 경우를 예상할 수 있기 때문에 보수 심볼들의 수는 3이 아니다는 것을 주시하자. 보수 심볼들의 수를 결정하기 위해서 프래그먼트의 소스 심볼들의 수를 간단히 사용하는 것과 반대로, 위에서 설명된 실시예들은 단지 상응하는 소스 세그먼트의 상응하는 소스 블록과 연관된 인덱스 정보로부터 보수 세그먼트 내에서 보수 블록의 위치를 계산하는 것을 가능하게 한다. 게다가, 소스 블록에서 소스 심볼들의 수 K가 증대함에 따라, 상응하는 보수 블록의 보수 심볼들의 수 KR는 K*R에 의해서 근접하게 근사화되는데, 그 이유는 일반적으로 KR이 기껏해야 ceil(K*R)이고, KR이 적어도 floor((K-1)*R)이기 때문이며, 여기서 floor(x)는 기껏해야 x인 가장 큰 정수이다.

[0587] 당업자가 인지할 바와 같이, 협력적인 HTTP/FEC 방법들을 이용하는 클라이언트에 의해 유리하게 사용될 수 있는 파일 구조를 생성하기 위한 위의 실시예들의 많은 변형들이 존재할 수 있다. 대안적인 실시예의 예로서, 리프리젠테이션을 위한 본래 세그먼트는 N>1개의 병렬 세그먼트들로 분할될 수 있고(i=1,..., N인 경우), 본래 세그먼트의 규정된 프랙션 F_i는 i번째 병렬 세그먼트에 포함되고, 여기서 F_i의 i=1,..., N의 합은 1과 같다. 본 실시예에 있어서, 보수 세그먼트 맵이 위에서 설명된 실시예에서의 소스 세그먼트 맵으로부터 유도

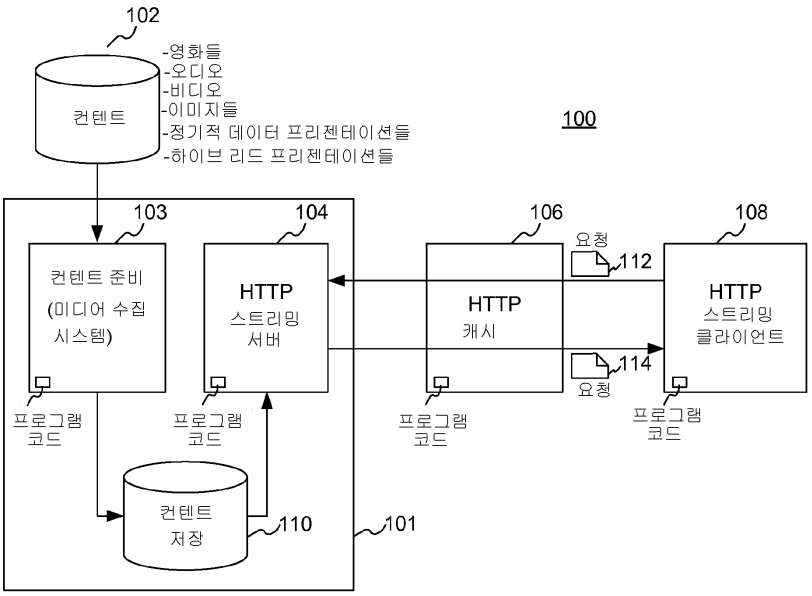
되는 방법과 유사하게, 병렬 세그먼트들 모두에 대한 세그먼트 맵을 유도하기 위해 사용되는 하나의 마스터 세그먼트 맵이 존재할 수 있다. 예컨대, 마스터 세그먼트 맵은 소스 미디어 데이터 모두가 병렬 세그먼트로 분할되지 않았고 대신에 하나의 본래 세그먼트에 포함된 경우에 프래그먼트 구조를 나타낼 수 있고, 이어서 i 번째 병렬 세그먼트에 대한 세그먼트 맵이, 만약 본래 세그먼트의 프래그먼트들의 제 1 프리픽스내 미디어 데이터의 양이 L 바이트들이라면, 제 1 i 병렬 세그먼트 간의 통합에서 이러한 프리픽스의 바이트들의 총 수가 $ceil(L*G_i)$임을 계산함으로써 마스터 세그먼트 맵으로부터 유도될 수 있고, 여기서 G_i는 F_j의 j=1,..., i에 걸친 합이다. 대안적인 실시예의 다른 예로서, 세그먼트들은 각각의 프래그먼트에 대한 보수 데이터가 바로 후속되는 그 각각의 프래그먼트에 대한 본래 소스 미디어 데이터의 조합으로 구성될 수 있어서, 소스 미디어 데이터 및 그 소스 미디어 데이터로부터의 FEC 코드를 사용하여 생성된 보수 데이터의 혼합을 포함하는 세그먼트를 유도한다. 대안적인 실시예의 다른 예로서, 소스 미디어 데이터 및 보수 데이터의 혼합을 포함하는 세그먼트가 소스 미디어 데이터 및 보수 데이터의 혼합을 포함하는 다수의 병렬 세그먼트들로 분할될 수 있다.

[0588] 추가적인 실시예들은 본 발명을 읽은 이후에 당업자에게 구상될 수 있다. 다른 실시예들에서는, 위에서 설명된 본 발명의 조합들 또는 서브-조합들이 유리하게 이루어질 수 있다. 예시적인 컴포넌트들의 배열들이 예시의 목적들을 위해서 도시되어 있고, 조합들, 추가들, 재배열들 등이 본 발명의 대안적인 실시예들에서 고찰된다는 점이 이해되어야 한다. 따라서, 비록 본 발명은 예시적인 실시예들에 대해 설명되었지만, 당업자는 다수의 변경들이 가능함을 인지할 것이다.

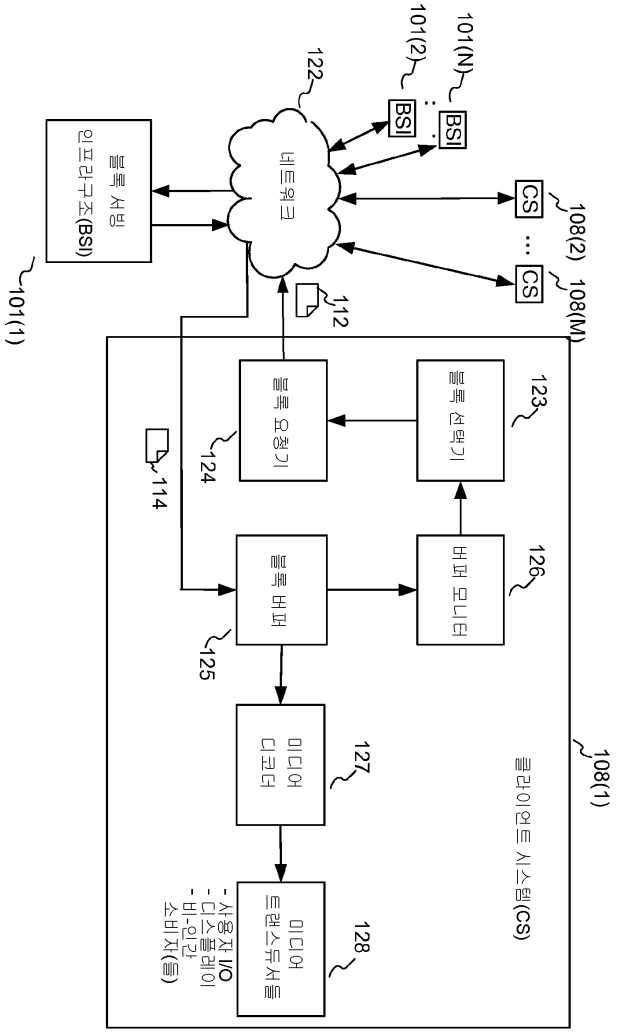
[0589] 예컨대, 여기서 설명된 프로세스들은 하드웨어 컴포넌트들, 소프트웨어 컴포넌트들, 및/또는 이들의 임의의 조합을 사용하여 구현될 수 있다. 일부 경우들에 있어서, 소프트웨어는 미디어에 제공되거나 또는 미디어로부터 분리되는 하드웨어 상에서의 실행을 위해 유형의 비일시적인 매체들 상에 제공될 수 있다. 명세서 및 도면들은 따라서 제약적인 의미보다는 예시적인 것으로 간주되어야 한다. 그러나, 다양한 변경들 및 변화들이 청구항들에서 설명된 바와 같은 본 발명의 사상 및 범위로부터 벗어나지 않고 본 발명에 대해 이루어질 수 있다는 점 및 본 발명이 모든 변경들 및 등가물들을 아래의 청구항들의 범위 내에 포함시키도록 의도된다는 점이 명백할 것이다.

도면

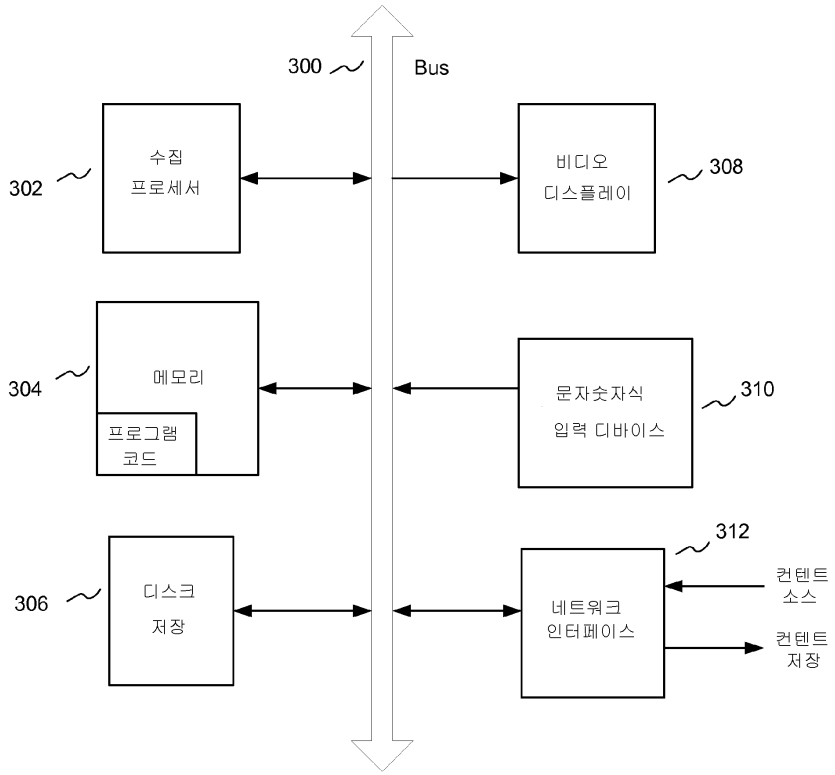
도면1



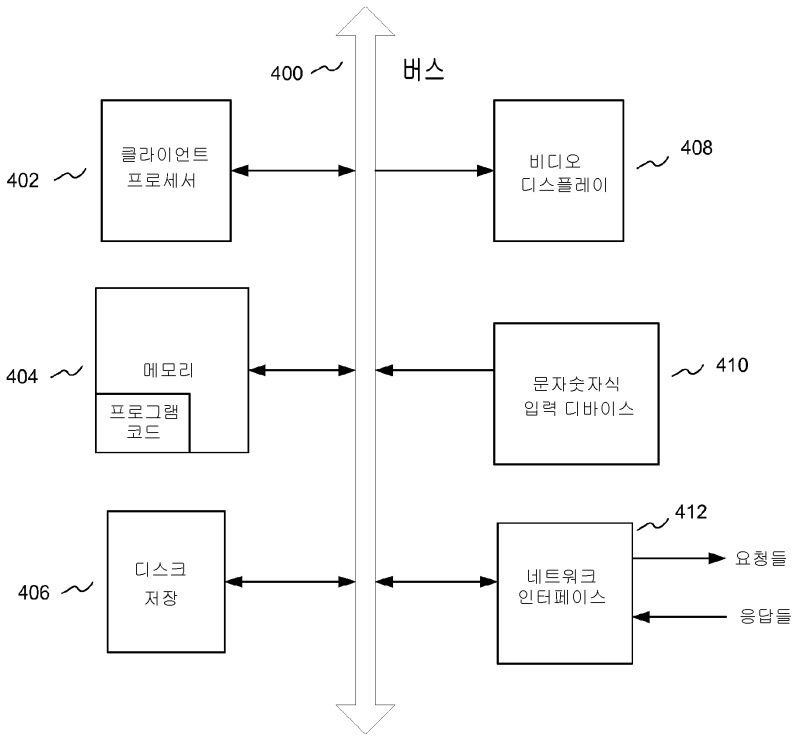
도면2


101(1)
블록 서빙 인프라구조(BSI)
122
네트워크
101(2)
101(N)
BSI
BSI
CS
CS
108(2)
108(M)
112
114
블록 요청기
124
블록 선택기
123
블록 버퍼
125
버퍼 모니터
126
미디어 디코더
127
미디어 트랜스듀서들
128
- 사용자 I/O
- 디스플레이
- 비-인간 소비자(들)
클라이언트 시스템(CS)
108(1)

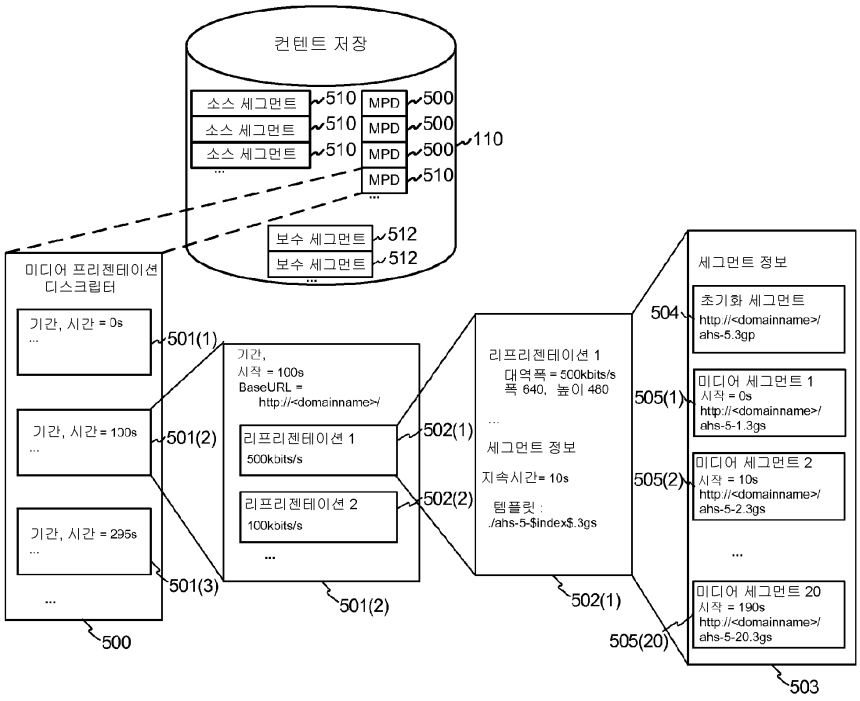
도면3


300
Bus
302
수집
프로세서
비디오
디스플레이
308
304
메모리
프로그램
코드
문자숫자식
입력 디바이스
310
306
디스크
저장
312
네트워크
인터페이스
컨텐트
소스
컨텐트
저장

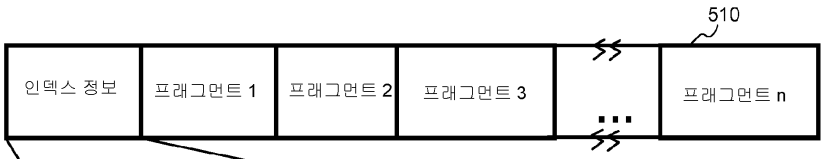
도면4


400
버스
402
클라이언트
프로세서
비디오
디스플레이
408
404
메모리
프로그램
코드
문자숫자식
입력 디바이스
410
406
디스크
저장
412
네트워크
인터페이스
요청들
응답들

도면5

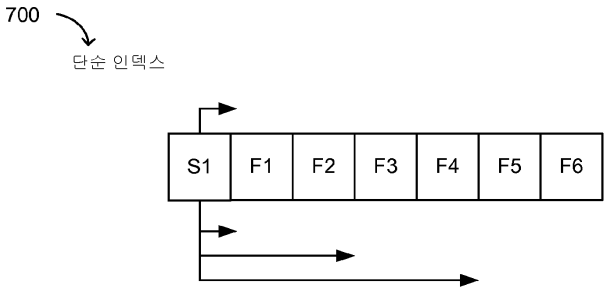


도면6

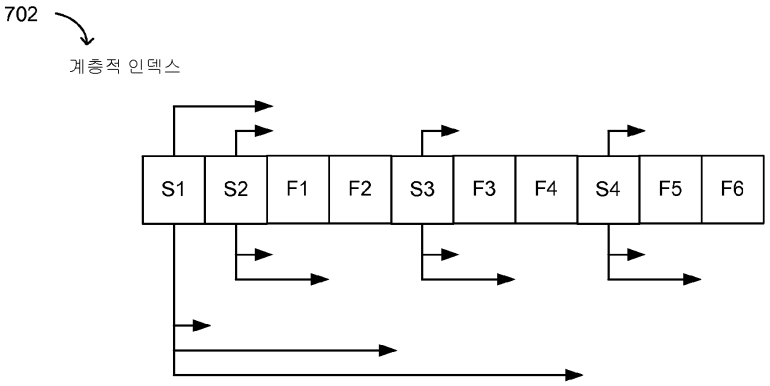


시간 오프셋 (ms)	바이트 오프셋	RAP 표시자
20,000	0	1
20,485	50,245	0
21,100	101,354	0
21,623	157,034	1
21,965	198,045	0
22,540	256,654	0
23,045	301,245	1
24,000	358,436	0
24,605	400,983	1
25,132	465,745	0
25,845	504,103	0

도면7a

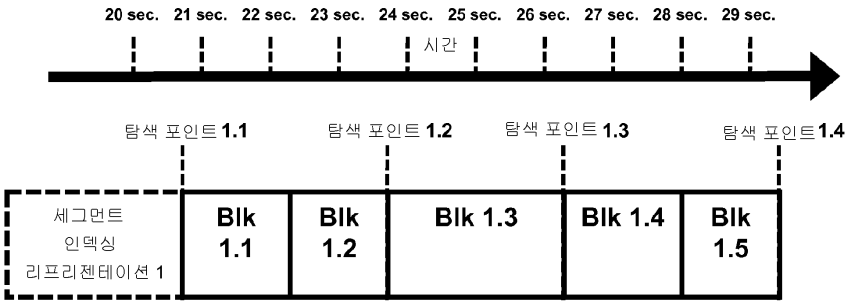


도면7b

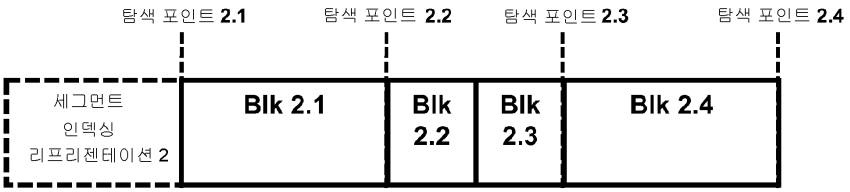


도면8a

정렬된 탐색 포인트들을 갖는 리프리젠테이션들을 위한 소스 세그먼트 구조



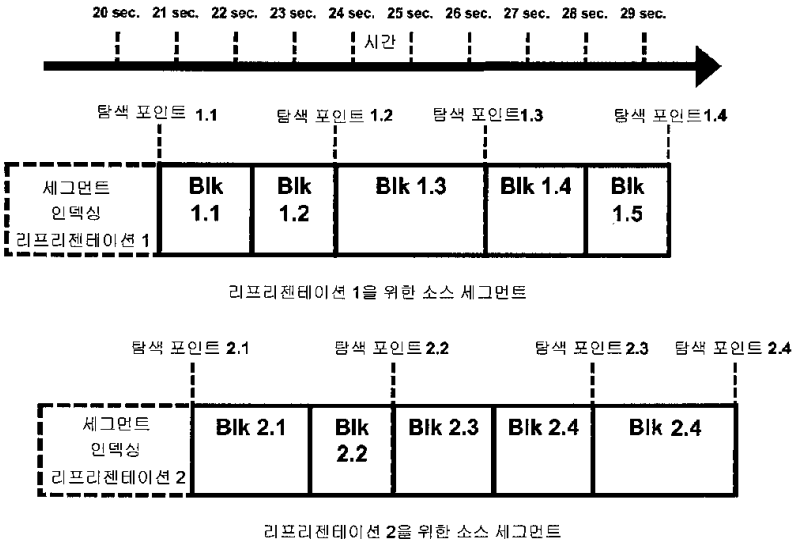
리프리젠테이션 1을 위한 소스 세그먼트



리프리젠테이션 2을 위한 소스 세그먼트

도면8b

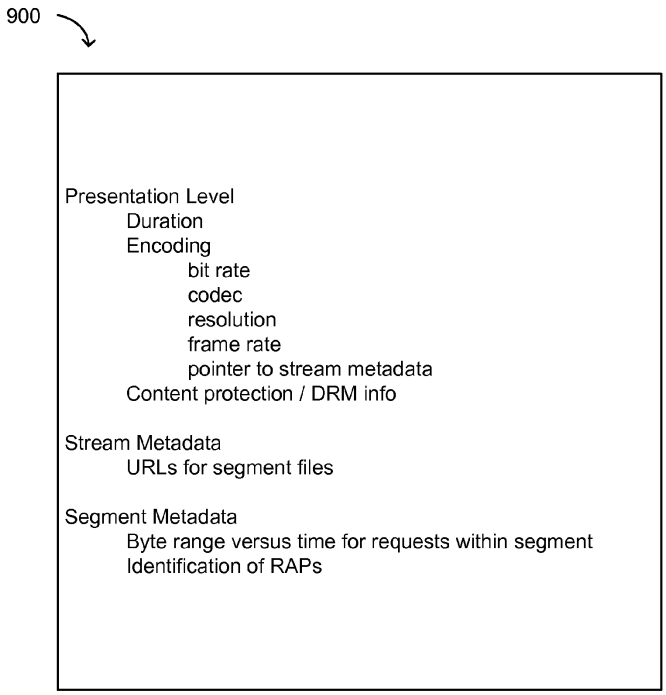
정렬된 탐색 포인트들을 갖는 리프리젠테이션들을 위한 소스 세그먼트 구조



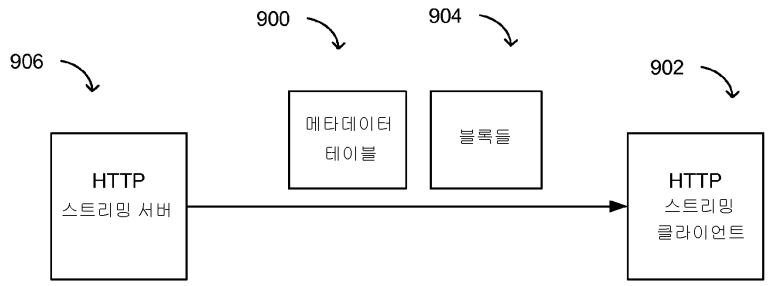
리프리젠테이션 1을 위한 소스 세그먼트

리프리젠테이션 2을 위한 소스 세그먼트

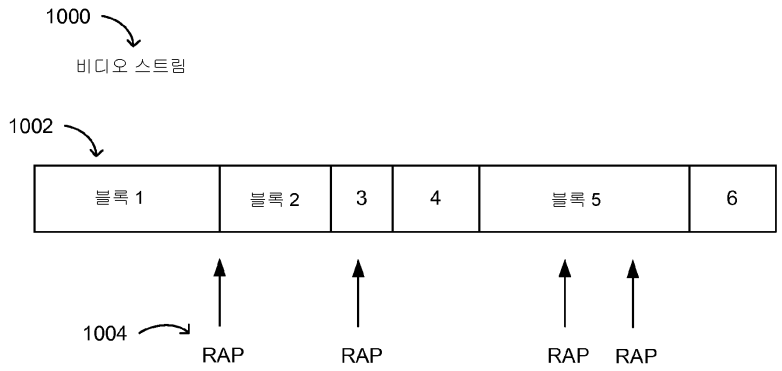
도면9a



도면9b

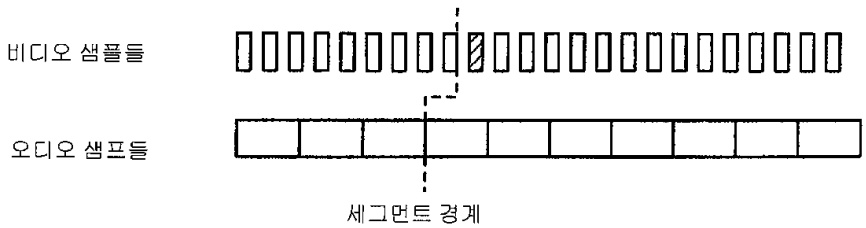


도면10

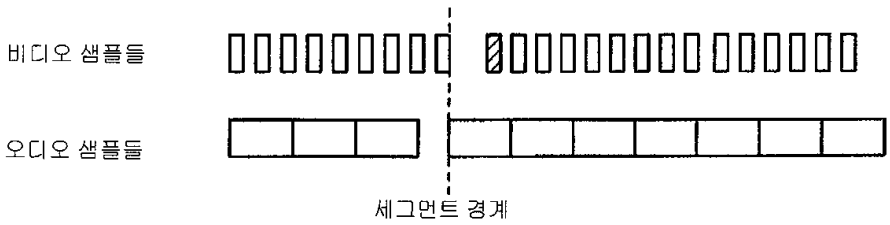


도면11

세그먼트들에 걸친 연속적인 타이밍



세그먼트들에 걸친 불연속적인 타이밍



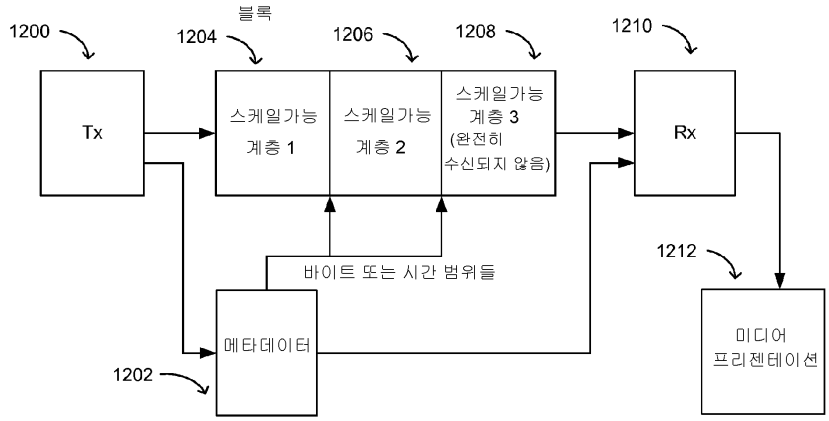
가정들 :

- 독립적인 이유들로 인해 경계 포인트로서 선택되는 빗금 친 프레임
- 오디오는 비디오보다 나중에 시작되지 않아야 한다

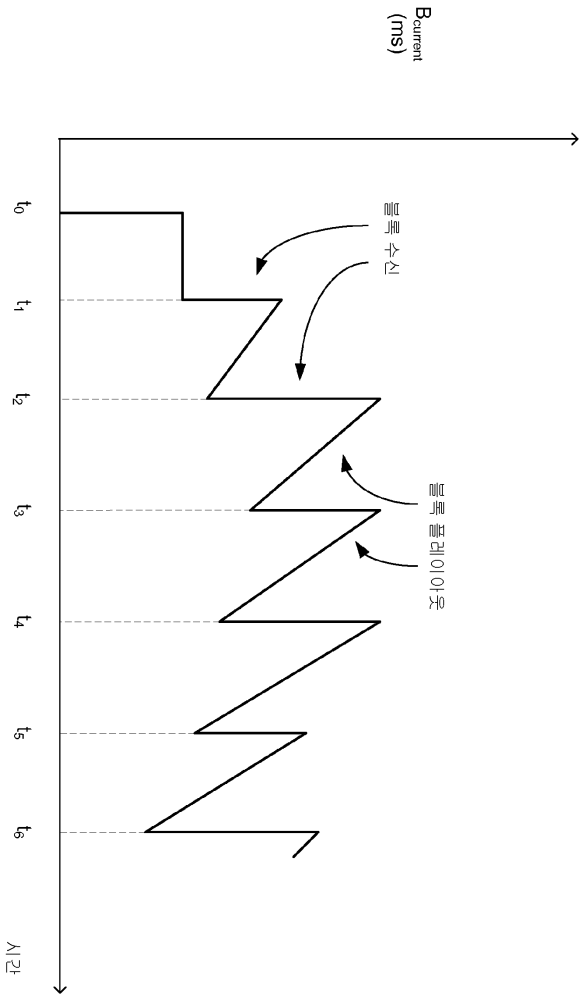
주시 :

- 연속적인 경우에, 제 2 세그먼트는 제 1 세그먼트와 동일한 컨텐트임
- 불연속적인 경우에, 제 2 세그먼트는 제 1 세그먼트와 상이한 컨텐트임

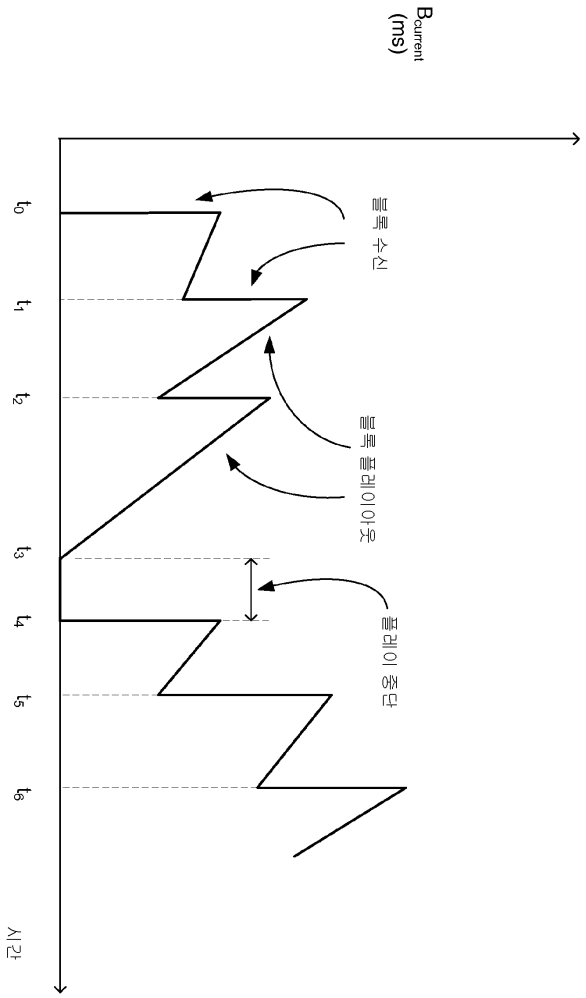
도면12


블록
1200
1204
1206
1208
1210
Tx
스케일가능
계층 1
스케일가능
계층 2
스케일가능
계층 3
(완전히
수신되지 않음)
Rx
1212
바이트 또는 시간 범위들
메타데이터
1202
미디어
프리젠테이션

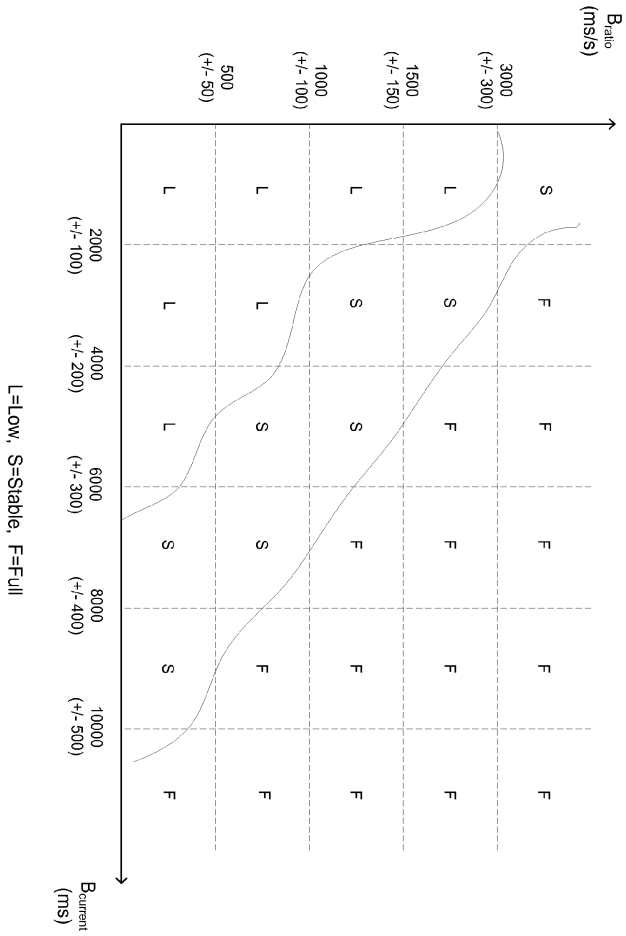
도면13


Bcurrent
(ms)
t0
t1
t2
t3
t4
t5
t6
시간
블록 수신
블록 플레이아웃

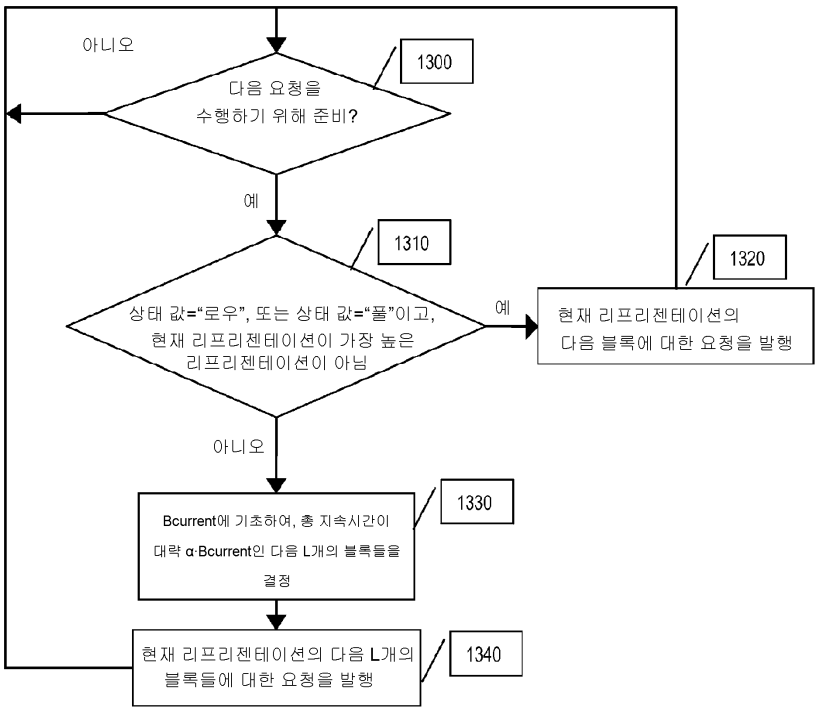
도면14



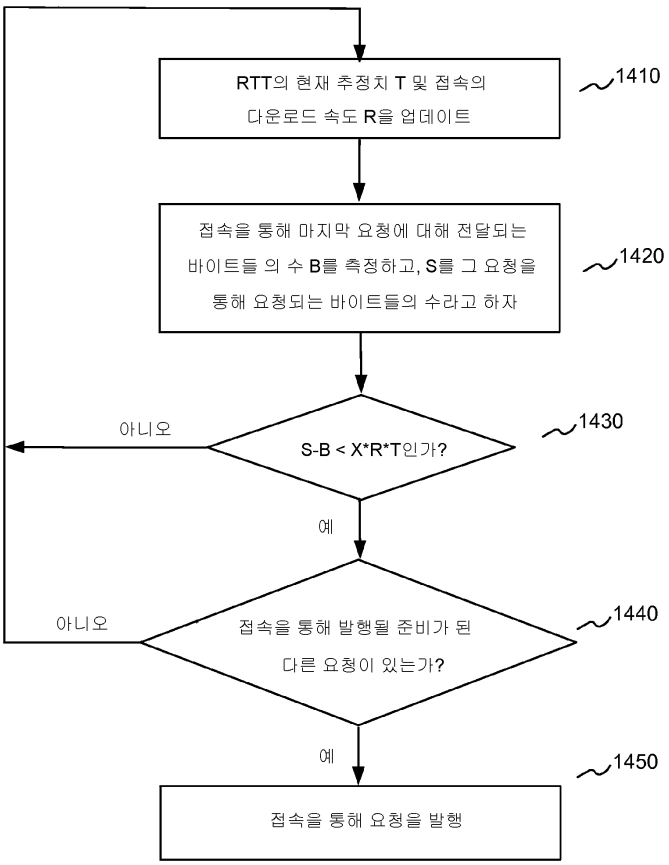
도면15


B ratio (ms/s)
500 (+/- 50)
1000 (+/- 100)
1500 (+/- 150)
3000 (+/- 300)
B current (ms)
2000 (+/- 100)
4000 (+/- 200)
6000 (+/- 300)
8000 (+/- 400)
10000 (+/- 500)
L=Low, S=Stable, F=Full

도면16


아니오
다음 요청을 수행하기 위해 준비?
1300
예
1310
상태 값="로우", 또는 상태 값="풀"이고, 현재 리프리젠테이션이 가장 높은 리프리젠테이션이 아님
예
1320
현재 리프리젠테이션의 다음 블록에 대한 요청을 발행
아니오
Bcurrent에 기초하여, 총 지속시간이 대략 α·Bcurrent인 다음 L개의 블록들을 결정
1330
현재 리프리젠테이션의 다음 L개의 블록들에 대한 요청을 발행
1340

도면17



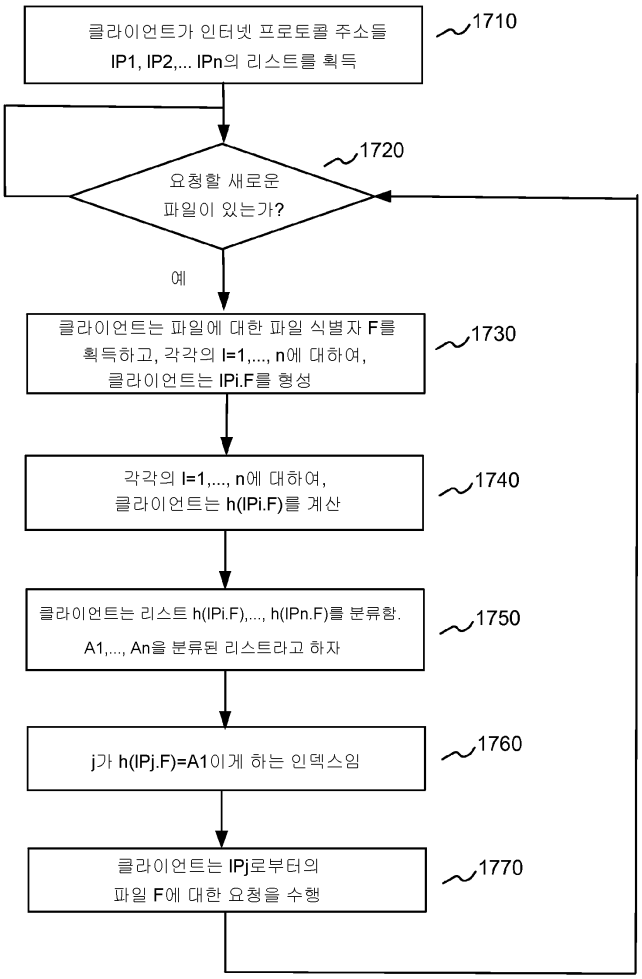
도면18

후보 요청 리스트	우선순위	접속 1에 대해 Ok?	접속 2에 대해 Ok?	접속 3에 대해 Ok?
요청 A	0	X		
요청 B	0	X	X	
요청 C	1	X		
요청 D	1			X
요청 E	2			X
요청 F	3		X	X

도면19

후보 요청 리스트	우선순위	접속 1에 대해 Ok?	접속 2에 대해 Ok?	접속 3에 대해 Ok?
요청 A	5	X		
요청 G	0			X
요청 C	4	X		
요청 D	1			X
요청 E	2			X
요청 F	3		X	X

도면20



도면21

```
<expression> ::= <literal> |
                 <variable> |
                 <function> |
                 '(' <expression> ')' |
                 <expression> <operator> <expression>

 <literal> ::= <string> | <number>

 <variable> ::= <token>

 <function> ::= <token>'(' [ <expression> *( ',' <expression> ) ] ')'

 <operator> ::= 1*<opchar>

 <token> ::= <tokenchar> *(<tokenchar> | <digit> )

 <string> ::= '"' *<char> '"'

 <number> ::= [ '-' ] 1*<digit> [ '.' 1*<digit> ]

 <digit> ::= '0-9'

 <char> ::= <any ASCII char except '"'> | '/' '"'

 <tokenchar> ::= 'A-Z' | 'a-z' |'_'

 <opchar> ::= '-' | '+' | '=' | '/' | ';' | '%' | '*' |
```

도면22

```
unsigned long Hash( const char *p, unsigned long max )
{
    unsigned long hash = 0;
        while( *p != 0 )
        hash = ( hash << 5 ) ^ ( ( hash & 0xf8000000 ) >> 27 ) ^ *p++;
    return hash % max;
}
```

도면23

예 1

타이틀:	간단한 고정된 도메인 매핑
파일 식별자 구성 규칙:	printf("http://<domainname>/%s_%3d.dfx", id, seq)
파일 ID:	SPEED_PAO_S2_E1_5gsyd75
시퀀스 #:	22
결과:	http://<domainname>/SPEED_PAO_S2_E1_5gsyd75_022.dfx

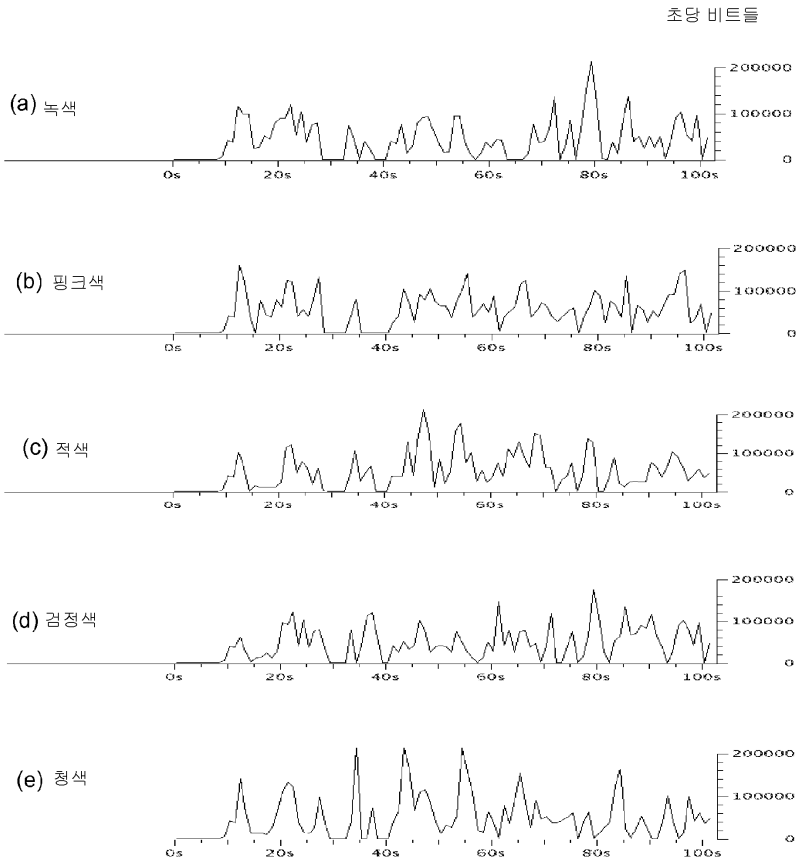
예 2

타이틀:	파일 ID에 기초한 1000개의 도메인들에 걸친 해시
파일 식별자 구성 규칙:	printf("http://%3d.<domainname>/%s_%3d.dfx", hash(id, 1000), id, seq)
파일 ID:	SPEED_PAO_S2_E1_5gsyd75
시퀀스 #:	22
결과:	http://564.<domainname>/SPEED_PAO_S2_E1_5gsyd75_022.dfx

예 3

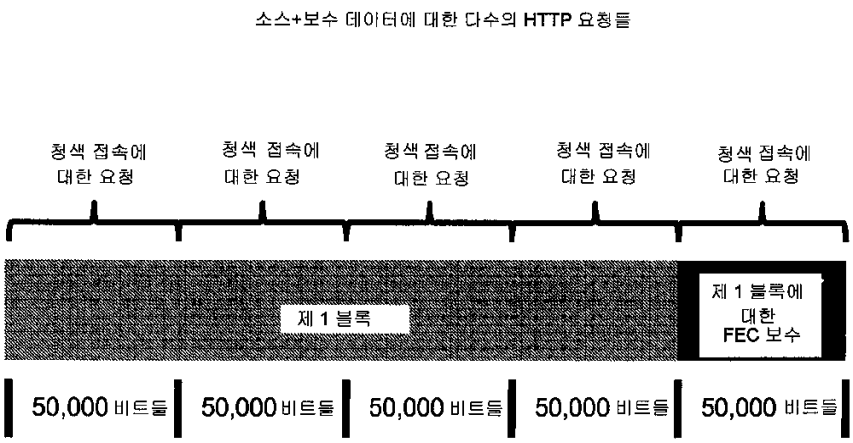
타이틀:	10개의 도메인들 및 1000개의 디렉토리들에 걸친 해시
파일 식별자 구성 규칙:	printf("http://%2d.<domainname>/%3d/%s_%3d.dfx", hash(id, 10), hash(id, 1000), id, seq)
파일 ID:	SPEED_PAO_S2_E1_5gsyd75
시퀀스 #:	22
결과:	http://4.<domainname>/564/SPEED_PAO_S2_E1_5gsyd75_022.dfx

도면24



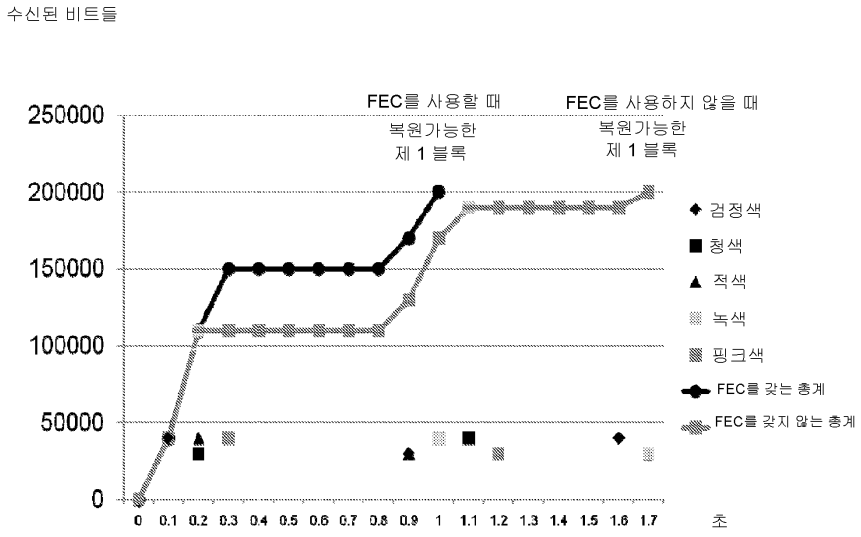
5개의 TCP 접속들의 통상적인 대역폭 변동

도면25

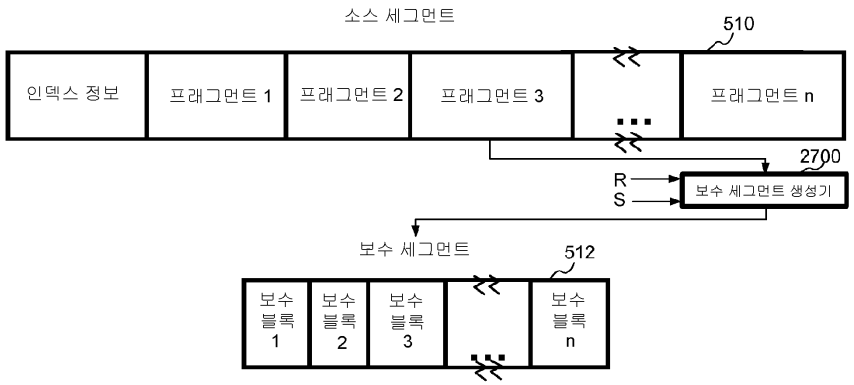


도면26

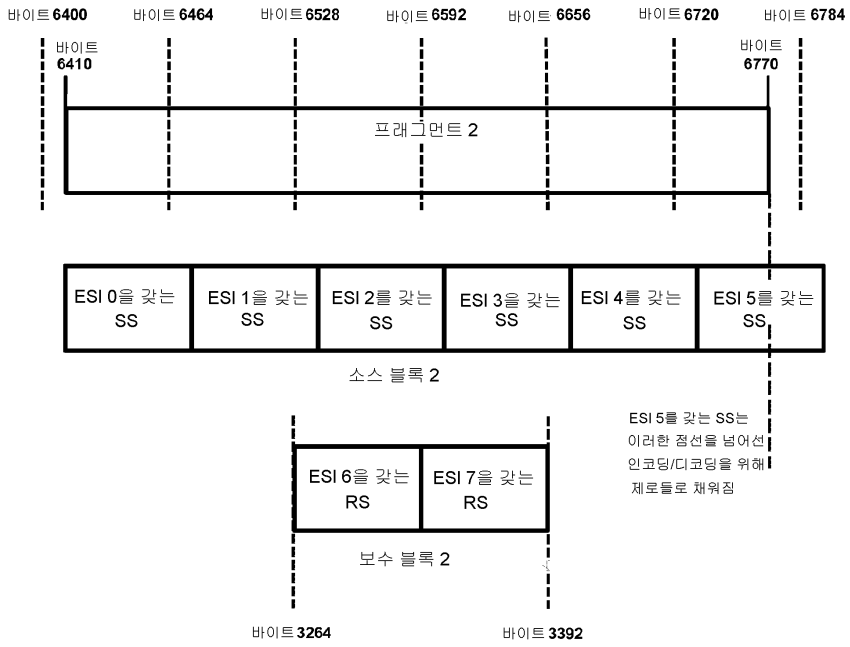
FEC를 갖는 그리고 갖지 않는 채널 재핑 시간의 예



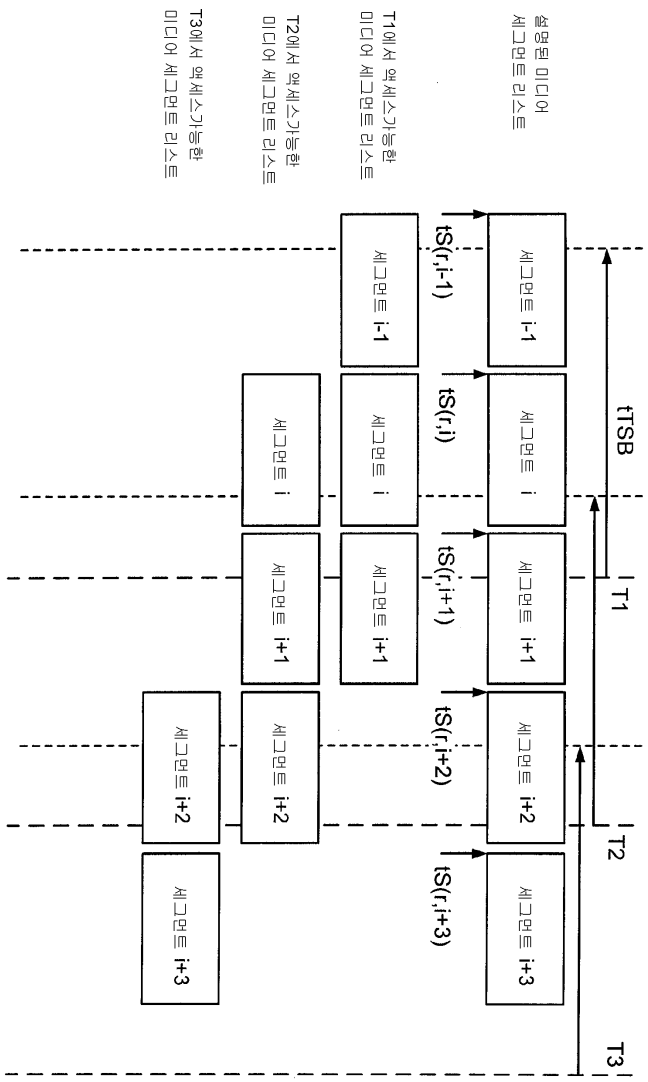
도면27



도면28


바이트 6400
바이트 6464
바이트 6528
바이트 6592
바이트 6656
바이트 6720
바이트 6784
바이트 6410
바이트 6770
프래그먼트 2
ESI 0을 갖는 SS
ESI 1을 갖는 SS
ESI 2를 갖는 SS
ESI 3을 갖는 SS
ESI 4를 갖는 SS
ESI 5를 갖는 SS
소스 블록 2
ESI 5를 갖는 SS는 이러한 점선을 넘어선 인코딩/디코딩을 위해 제로들로 채워짐
ESI 6을 갖는 RS
ESI 7을 갖는 RS
보수 블록 2
바이트 3264
바이트 3392

도면29


셀 영역 미디어 세그먼트 리스트
T1에서 액세스가능한 미디어 세그먼트 리스트
T2에서 액세스가능한 미디어 세그먼트 리스트
T3에서 액세스가능한 미디어 세그먼트 리스트
세그먼트 i-1
세그먼트 i
세그먼트 i+1
세그먼트 i+2
세그먼트 i+3
tS(r,i-1)
tS(r,i)
tS(r,i+1)
tS(r,i+2)
tS(r,i+3)
tTSB
T1
T2
T3