(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0002500
(43) 공개일자 2025년01월07일

(51) 국제특허분류(Int. Cl.)
H04L 65/75 (2022.01) *H04L 65/612* (2022.01)
H04L 65/80 (2022.01)
(52) CPC특허분류
H04L 65/75 (2022.05)
H04L 65/612 (2022.05)
(21) 출원번호 10-2024-7037908
(22) 출원일자(국제) 2023년04월13일
심사청구일자 없음
(85) 번역문제출일자 2024년11월13일
(86) 국제출원번호 PCT/US2023/018469
(87) 국제공개번호 WO 2023/205025
국제공개일자 2023년10월26일
(30) 우선권주장
63/332,210 2022년04월18일 미국(US)
(뒷면에 계속)

(71) 출원인
돌비 레버러토리즈 라이센싱 코오포레이션
미합중국, 캘리포니아 94103, 샌프란시스코, 마켓 스트리트 1275
(72) 발명자
라이드 밀러, 제프리
미국 94103 캘리포니아 샌프란시스코 마켓 스트리트 1275 돌비 레버러토리즈, 인크. 내
산체즈, 프레디
미국 94103 캘리포니아 샌프란시스코 마켓 스트리트 1275 돌비 레버러토리즈, 인크. 내
(뒷면에 계속)
(74) 대리인
양영준, 김연송, 백만기

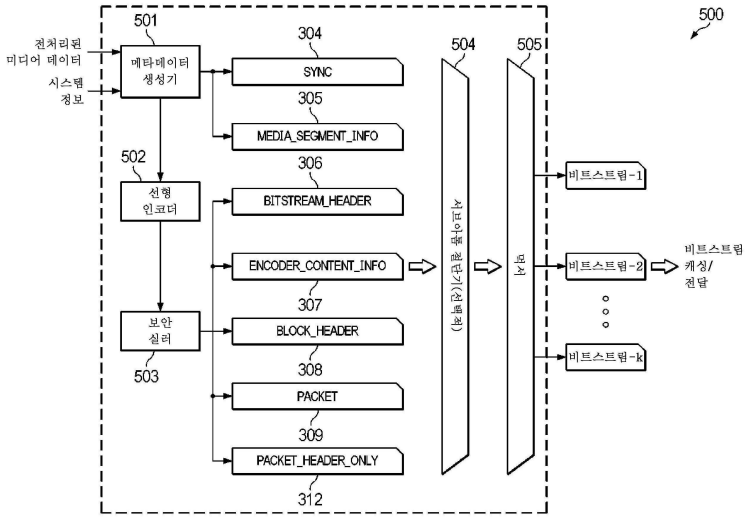
전체 청구항 수 : 총 67 항

(54) 발명의 명칭 **코딩된 미디어에 대한 멀티소스 방법들 및 시스템들**

(57) 요 약

코딩된 미디어에 대한 멀티소스 방법들 및 시스템들의 실시예들이 포함된다. 일부 실시예들에서, 방법은, 제1 디바이스에서, 미디어 자산을 나타내는 미디어 데이터를 수신하는 단계; 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 획득하는 단계; 미디어 데이터의 적어도 일부를 제1 코딩 프로세스에 따라 미디어 자산에 대응하는 코딩된 데이터로 인코딩하는 단계; 제1 코딩 프로세스와 연관된 정보에 기반하여 제1 복수의 데이터 요소와 상이한 제2 복수의 데이터 요소를 생성하는 단계; 제1 복수의 데이터 요소 및 제2 복수의 데이터 요소를 미디어 자산을 나타내는 하나 이상의 코딩된 비트스트림으로 결합하는 단계; 및 하나 이상의 네트워크 경로를 이용하여 하나 이상의 제2 디바이스에 하나 이상의 코딩된 비트스트림을 전송하는 단계를 포함한다.

대 표 도



(52) CPC특허분류
H04L 65/80 (2022.05)

(72) 발명자
위, 밍차오
호주 2060 뉴 사우스 웨일즈 시드니 도리스 스트리트 401/22
클라우드, 제이슨 마이클
미국 94103 캘리포니아 샌프란시스코 마켓 스트리트 1275 돌비 레버러토리즈, 인크. 내
오스본, 엘리엇
미국 94103 캘리포니아 샌프란시스코 마켓 스트리트 1275 돌비 레버러토리즈, 인크. 내
앤티오치, 토마스 프랭클린
미국 94103 캘리포니아 샌프란시스코 마켓 스트리트 1275 돌비 레버러토리즈, 인크. 내

(30) 우선권주장
63/332,212 2022년04월18일 미국(US)
63/443,933 2023년02월07일 미국(US)

명 세 서

청구범위

청구항 1

방법으로서,

제1 디바이스에서,

적어도 하나의 프로세서에 의해, 미디어 자산을 나타내는 미디어 데이터를 수신하는 단계;

상기 적어도 하나의 프로세서에 의해, 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 획득하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 미디어 자산에 대응하는 코딩된 데이터를 획득하기 위해 제1 코딩 프로세스에 따라 상기 미디어 데이터의 적어도 일부를 인코딩하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 제1 코딩 프로세스와 연관된 정보에 기반하여 상기 제1 복수의 데이터 요소와 상이한 제2 복수의 데이터 요소를 생성하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 제1 복수의 데이터 요소 및 상기 제2 복수의 데이터 요소를 상기 미디어 자산을 나타내는 하나 이상의 코딩된 비트스트림으로 결합하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 하나 이상의 네트워크 경로를 이용하여 상기 하나 이상의 코딩된 비트스트림을 하나 이상의 제2 디바이스에 전송하는 단계

를 포함하는, 방법.

청구항 2

제1항에 있어서,

상기 제1 코딩 프로세스는 상기 미디어 자산에 대응하는 코딩된 데이터를 획득하기 위한 선형 코딩 프로세스인, 방법.

청구항 3

제1항에 있어서,

상기 제1 코딩 프로세스는 상기 미디어 데이터를 다른 선형 코딩 프로세스와 일치하는 상기 코딩된 데이터로 변환하는 비선형 코딩 프로세스인, 방법.

청구항 4

제1항에 있어서,

상기 제1 복수의 데이터 요소는 비트스트림 식별 데이터, 및 콘텐츠 특정 인코드 데이터 또는 미디어 세그먼트 데이터 중 적어도 하나를 포함하는, 방법.

청구항 5

제1항에 있어서,

상기 데이터 요소들은 데이터 구조들인, 방법.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

상기 제1 복수의 데이터 요소는 동기화 데이터 요소를 포함하고, 상기 하나 이상의 코딩된 비트스트림 각각은 동기화 데이터 요소로 시작하는, 방법.

청구항 7

제1항 내지 제6항 중 어느 한 항에 있어서,

상기 동기화 데이터 요소는,

상기 하나 이상의 코딩된 비트스트림의 각각의 코딩된 비트스트림을 공통 소스 미디어 자산으로부터 생성된 것으로서 식별하는 식별 데이터;

상기 하나 이상의 코딩된 비트스트림의 각각의 코딩된 비트스트림을 상기 공통 소스 미디어 자산으로부터 생성된 다른 코딩된 비트스트림들과 동기화하기 위한 동기화 데이터; 및

상기 하나 이상의 코딩된 비트스트림의 각각의 코딩된 비트스트림의 계보 데이터(lineage data)

중 적어도 하나를 포함하는, 방법.

청구항 8

제7항에 있어서,

계보 데이터는 상기 각각의 코딩된 비트스트림에 포함된 코딩된 심볼들을 생성하는데 이용되는 원래의 또는 코딩된 심볼들을 식별하는 데이터, 상기 각각의 코딩된 비트스트림에 포함된 상기 코딩된 심볼들을 생성하기 위해 원래의 또는 코딩된 심볼들이 어떻게 결합되었는지를 식별하는 데이터, 및 상기 각각의 코딩된 비트스트림에 포함된 상기 코딩된 심볼들을 생성하는데 이용되는 인코딩 계수들을 식별하는 데이터를 포함하는, 방법.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서,

상기 하나 이상의 코딩된 비트스트림 전체에 걸쳐 동기화 데이터 요소들을 인터리빙하는 단계를 더 포함하고, 상기 동기화 데이터 요소들은 상기 하나 이상의 코딩된 비트스트림 중 하나의 코딩된 비트스트림을 수신하는 제2 전자 디바이스가 상기 하나 이상의 코딩된 비트스트림 중 상기 하나의 코딩된 비트스트림 내의 복수의 포인트로부터 디코딩을 개시할 수 있게 하는, 방법.

청구항 10

제1항 내지 제9항 중 어느 한 항에 있어서,

상기 제1 복수의 데이터 요소 또는 상기 제2 복수의 데이터 요소는 수신된 미디어 데이터와 연관된 시스템 데이터에 적어도 부분적으로 기반하여 생성되는, 방법.

청구항 11

제1항 내지 제9항 중 어느 한 항에 있어서,

제1 복수의 데이터 요소를 획득하는 단계는 수신된 미디어 데이터 또는 상기 수신된 미디어 데이터와 연관된 시스템 데이터로부터 상기 제1 복수의 데이터 요소를 생성하는 단계를 포함하는, 방법.

청구항 12

제1항 내지 제11항 중 어느 한 항에 있어서,

제1 코딩 프로세스에 따라 상기 미디어 데이터를 인코딩하는 것은 상기 미디어 데이터를 복수의 원래의 심볼로 분할하고 상기 복수의 원래의 심볼에 기반하여 복수의 코딩된 심볼을 생성하는 것을 포함하는, 방법.

청구항 13

제12항에 있어서,

코딩된 심볼은 네트워크 코딩 기술에 따라 상기 복수의 원래의 심볼의 서브세트를 선형적으로 결합함으로써 생성되는, 방법.

청구항 14

제1항 내지 제13항 중 어느 한 항에 있어서,

상기 미디어 데이터는 선형 코딩된 비트스트림을 포함하는, 방법.

청구항 15

제1항 내지 제14항 중 어느 한 항에 있어서,

제1 복수의 데이터 요소를 획득하는 단계는,

상기 미디어 데이터로부터 메타데이터를 추출하는 단계; 및

추출된 메타데이터로부터 상기 제1 복수의 데이터 요소를 계산하는 단계

를 포함하는, 방법.

청구항 16

제1항 내지 제15항 중 어느 한 항에 있어서,

메타데이터는 선형 디코딩을 수행하지 않고 상기 미디어 데이터로부터 추출되는, 방법.

청구항 17

제1항 내지 제16항 중 어느 한 항에 있어서,

제1 코딩 프로세스에 따라 상기 미디어 데이터를 인코딩하는 것은 상기 코딩된 비트스트림 내의 선형 코딩된 심볼들을 상기 미디어 자산을 나타내는 새로운 선형 코딩된 심볼들로 재인코딩하는 것을 포함하는, 방법.

청구항 18

제1항 내지 제17항 중 어느 한 항에 있어서,

결합하기 전에, 제1 크기를 갖는 하나 이상의 데이터 요소를 상기 제1 크기보다 작은 제2 크기를 갖는 데이터 요소들로 분할하는 단계를 더 포함하는, 방법.

청구항 19

제1항 내지 제18항 중 어느 한 항에 있어서,

상기 제2 복수의 데이터 요소 중 하나 이상의 데이터 요소에 해시 데이터를 추가하여 하나 이상의 비트스트림을 수신하는 디바이스에 의한 검증, 인증, 또는 무결성 중 적어도 하나를 가능하게 하는 단계를 더 포함하는, 방법.

청구항 20

제1항 내지 제19항 중 어느 한 항에 있어서,

상기 미디어 자산에 대응하는 파일, 상기 미디어 자산의 블록, 상기 미디어 자산의 세그먼트 및 상기 미디어 자산의 패킷 중 하나 이상과 연관되는 상기 제2 복수의 데이터 요소 중 하나 이상의 데이터 요소에 해시 데이터를 추가하는 단계를 더 포함하는, 방법.

청구항 21

제1항 내지 제20항 중 어느 한 항에 있어서,

결합하는 단계는 디렉토리 데이터 요소를 상기 하나 이상의 코딩된 비트스트림에 삽입하는 단계를 포함하는, 방법.

청구항 22

제21항에 있어서,

상기 디렉토리 데이터 요소는 수신 디바이스가 각각의 비트스트림 내의 서브아톰들(subatoms)을 식별하는 것을

돕는데 이용되는, 방법.

청구항 23

제1항 내지 제22항 중 어느 한 항에 있어서,

상기 하나 이상의 코딩된 비트스트림은 제1 유형의 데이터 요소들을 포함하는 제1 코딩된 비트스트림 및 상기 제1 유형의 데이터 요소들을 포함하지 않는 제2 코딩된 비트스트림을 포함하는, 방법.

청구항 24

제1항 내지 제23항 중 어느 한 항에 있어서,

클라이언트 상에서 디코더를 초기화하는데 이용되는 상기 데이터 요소들 또는 서브아톰들은 레이턴시 민감 데이터가 아닌, 방법.

청구항 25

제1항 내지 제24항 중 어느 한 항에 있어서,

비트스트림은 sync() 서브아톰, 및 bitstream_header(), block_header(), encoder_content_info() 또는 media_segment_info() 서브아톰들 중 적어도 하나를 포함하는, 방법.

청구항 26

제1항 내지 제25항 중 어느 한 항에 있어서,

상기 데이터 요소들 또는 서브아톰들은 인코딩된 데이터 또는 레이턴시 민감 데이터를 운반하는, 방법.

청구항 27

제1항 내지 제26항 중 어느 한 항에 있어서,

비트스트림은 packet() 서브아톰들 및 sync() 서브아톰들을 포함하지만, bitstream_header(), block_header(), encoder_content_info() 또는 media_segment_info() 서브아톰들 중 적어도 하나를 포함하지 않는, 방법.

청구항 28

제1항 내지 제27항 중 어느 한 항에 있어서,

전송하는 단계는,

제1 유형의 통신 채널을 이용하여 제1 코딩된 비트스트림을 전송하는 단계; 및

상기 제1 유형과 상이한 제2 유형의 통신 채널을 이용하여 제2 코딩된 비트스트림을 전송하는 단계

를 포함하는, 방법.

청구항 29

제28항에 있어서,

상기 제2 유형의 통신 채널은 더 높은 레이턴시 채널, 더 낮은 대역폭 채널, 신뢰성 있는 채널 또는 제1 유형의 전송 프로토콜과 상이한 제2 유형의 전송 프로토콜을 이용하는 채널 중 적어도 하나인, 방법.

청구항 30

방법으로서,

제1 디바이스에서,

적어도 하나의 프로세서에 의해, 미디어 자산과 연관된 하나 이상의 코딩된 비트스트림을 수신하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 하나 이상의 코딩된 비트스트림을 버퍼에 저장하는 단계;

상기 버퍼에 저장된 상기 하나 이상의 코딩된 비트스트림을 처리하는 단계 - 상기 하나 이상의 코딩된 비트스트

림을 처리하는 단계는,

상기 적어도 하나의 프로세서에 의해, 상기 하나 이상의 코딩된 비트스트림으로부터 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 추출하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 하나 이상의 코딩된 비트스트림으로부터, 상기 하나 이상의 코딩된 비트스트림을 생성하는데 이용된 각각의 코딩 프로세스들과 연관되는 정보를 포함하는 제2 복수의 데이터 요소를 추출하는 단계

를 포함함 - ; 및

상기 적어도 하나의 프로세서에 의해, 상기 제2 복수의 데이터 요소에 기반하여 상기 미디어 자산을 나타내는 디코딩된 데이터를 생성하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 제2 복수의 데이터 요소에 기반하여 상기 미디어 자산을 나타내는 새롭게 코딩된 데이터를 생성하는 단계

중 하나를 수행하는 단계

를 포함하는, 방법.

청구항 31

제30항에 있어서,

디코딩된 데이터를 생성하는 단계는 코딩된 데이터를 생성하는데 이용되는 인코딩 유형의 결정에 따라 복수의 디코더의 각각의 디코더로 상기 코딩된 데이터를 디코딩하는 단계를 포함하는, 방법.

청구항 32

제31항에 있어서,

상기 인코딩 유형은 추출된 제1 복수의 데이터 요소로부터 결정되는, 방법.

청구항 33

제31항에 있어서,

상기 인코딩 유형은 추출된 제2 복수의 데이터 요소로부터 결정되는, 방법.

청구항 34

제31항에 있어서,

상기 인코딩 유형은 추출된 제2 복수의 데이터 요소를 이용하지 않고 상기 제1 복수의 데이터 요소로부터 결정되는, 방법.

청구항 35

제31항에 있어서,

상기 인코딩 유형은 추출된 제1 복수의 데이터 요소를 이용하지 않고 상기 제2 복수의 데이터 요소로부터 결정되는, 방법.

청구항 36

제31항 내지 제35항 중 어느 한 항에 있어서,

상기 디코딩된 데이터 및 상기 제1 복수의 데이터 요소로부터 상기 미디어 자산에 대응하는 소비가능한 데이터를 생성하는 단계를 더 포함하는, 방법.

청구항 37

제33항에 있어서,

상기 소비가능한 데이터는 제1 전자 디바이스 상에서 실행되는 애플리케이션 또는 애플리케이션에 의해 지원되는 미디어 파일 유형과 호환가능한, 방법.

청구항 38

제30항 내지 제37항 중 어느 한 항에 있어서,

디코딩된 데이터를 생성하는 단계는,

선형 코딩을 수행하지 않고 제1 유형의 코딩된 데이터를 제2 유형의 코딩된 데이터로 처리하는 단계; 및

상기 제2 유형의 코딩된 데이터를 디코딩할 수 있는 디코더를 이용하여 상기 제2 유형의 코딩된 데이터를 디코딩하는 단계

를 더 포함하는, 방법.

청구항 39

제38항에 있어서,

상기 코딩된 데이터를 처리하는 것은 계수들을 정렬하거나 상기 코딩된 데이터를 공통 유한 필드에 투영하는 코딩 통합기에 의해 수행되는, 방법.

청구항 40

제30항 내지 제39항 중 어느 한 항에 있어서,

새롭게 코딩된 데이터를 생성하는 단계는 코딩된 데이터를 생성하는데 이용되는 인코딩 유형의 결정에 따라 복수의 리인코더(re-encoder)의 각각의 리인코더로 상기 제2 복수의 데이터 요소에 포함된 상기 코딩된 데이터를 처리하는 단계를 포함하는, 방법.

청구항 41

제40항에 있어서,

상기 인코딩 유형은 추출된 제1 복수의 데이터 요소로부터 결정되는, 방법.

청구항 42

제40항에 있어서,

상기 인코딩 유형은 추출된 제2 복수의 데이터 요소로부터 결정되는, 방법.

청구항 43

제40항에 있어서,

상기 인코딩 유형은 추출된 제2 복수의 데이터 요소를 이용하지 않고 상기 제1 복수의 데이터 요소로부터 결정되는, 방법.

청구항 44

제40항에 있어서,

상기 인코딩 유형은 추출된 제1 복수의 데이터 요소를 이용하지 않고 상기 제2 복수의 데이터 요소로부터 결정되는, 방법.

청구항 45

제40항에 있어서,

새롭게 코딩된 데이터를 생성하는 단계는,

선형 디코딩을 수행하지 않고 제1 유형의 코딩된 데이터를 제2 유형의 코딩된 데이터로 처리하는 단계; 및

상기 제2 유형의 코딩된 데이터를 인코딩할 수 있는 리인코더를 이용하여 상기 제2 유형의 코딩된 데이터를 재인코딩하는 단계

를 포함하는, 방법.

청구항 46

제30항 내지 제45항 중 어느 한 항에 있어서,

코딩된 데이터를 처리하는 단계는 계수들을 정렬하거나 상기 코딩된 데이터를 공통 유한 필드에 투영하는 코딩 통합기에 의해 수행되는, 방법.

청구항 47

제30항 내지 제46항 중 어느 한 항에 있어서,

상기 새롭게 코딩된 데이터를 저장하거나 상기 새롭게 코딩된 데이터로부터 하나 이상의 비트스트림을 생성하는 단계를 더 포함하는, 방법.

청구항 48

제30항 내지 제47항 중 어느 한 항에 있어서,

상기 버퍼에 저장된 상기 하나 이상의 코딩된 비트스트림을 처리하기 전에, 각각이 제1 크기를 갖는, 상기 제1 복수의 데이터 요소 또는 상기 제2 복수의 데이터 요소의 제1 수량의 데이터 요소들을 상기 제1 크기보다 큰 제2 크기를 갖는 더 적은 수량의 데이터 요소들로 결합하는 단계를 더 포함하는, 방법.

청구항 49

제48항에 있어서,

결합하는 단계는 CHUNKED_SUBATOM 서브아톰 내의 데이터에 기반하는, 방법.

청구항 50

제30항 내지 제48항 중 어느 한 항에 있어서,

디코딩 또는 재인코딩 전에, 상기 제2 복수의 데이터 요소로부터 무결성 데이터를 추출하는 단계; 및

각각의 무결성 데이터에 기반한 각각의 데이터 요소들에 대한 성공적인 검증의 결정에 따라, 디코딩 또는 재인코딩 중 적어도 하나를 수행하는 단계; 및

각각의 무결성 데이터에 기반한 각각의 데이터 요소들에 대한 성공적이지 않은 검증의 결정에 따라, 디코딩 또는 재인코딩 중 적어도 하나를 수행하지 않는 단계

를 더 포함하는, 방법.

청구항 51

제50항에 있어서,

상기 무결성 데이터는 해시 데이터를 포함하고, 상기 해시 데이터의 검증은 키 정보에 기반하는, 방법.

청구항 52

제51항에 있어서,

키 정보는 공개 키 또는 개인 키 중 적어도 하나를 포함하는, 방법.

청구항 53

제51항에 있어서,

키 정보는 인증 서버로부터 수신되는, 방법.

청구항 54

제51항 내지 제53항 중 어느 한 항에 있어서,

디코딩 또는 재인코딩 전에, 상기 제2 복수의 데이터 요소가 각각의 암호화된 데이터를 포함한다는 결정에 따라, 상기 각각의 암호화된 데이터를 해독 키를 이용하여 해독하는 단계를 더 포함하는, 방법.

청구항 55

제51항에 있어서,

상기 키 정보 또는 해시 데이터는 bitstream_encryption_key_id와 연관되는, 방법.

청구항 56

제30항에 있어서,

미디어 자산의 소유자 또는 생성자에 의해 상기 미디어 자산과 연관된 수신된 하나 이상의 코딩된 비트스트림에 암호화가 적용되어, 비허가된 액세스, 소비, 디코딩 또는 재인코딩을 방지하는, 방법.

청구항 57

제30항에 있어서,

제1 전자 디바이스와 연관된 식별 정보를 상기 제1 전자 디바이스와 상이한 제2 전자 디바이스에 전송하는 단계; 및

미디어 자산과 연관된 상기 하나 이상의 코딩된 비트스트림의 액세스, 소비, 디코딩 또는 재인코딩을 가능하게 하는 해독 키를 상기 제2 전자 디바이스로부터 수신하는 단계

를 더 포함하는, 방법.

청구항 58

방법으로서,

제1 전자 디바이스에서,

적어도 하나의 프로세서에 의해, 제1 유형의 제1 코딩된 비트스트림을 획득하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 제1 유형과 상이한 제2 유형의 제2 코딩된 비트스트림을 생성하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 제2 코딩된 비트스트림을 하나 이상의 네트워크 경로를 통해 하나 이상의 제2 전자 디바이스에 전송하는 단계

를 포함하는, 방법.

청구항 59

제58항에 있어서,

제1 유형의 제1 비트스트림은 제1 유형의 선형 코딩 프로토콜 또는 표준에 따라 생성되는, 방법.

청구항 60

제58항 또는 제59항에 있어서,

제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 체계적 패킷들 및 상기 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하는, 방법.

청구항 61

제58항 내지 제60항 중 어느 한 항에 있어서,

제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 체계적 패킷들을 포함하고, 상기 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하지 않는, 방법.

청구항 62

제58항 내지 제61항 중 어느 한 항에 있어서,

제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하고, 상기 데이터 세그먼트와 연관된 체계적 패킷들을 포함하지 않는, 방법.

청구항 63

제59항 내지 제62항 중 어느 한 항에 있어서,

제2 유형의 제2 비트스트림은 제2 유형의 선형 코딩 프로토콜 또는 표준에 따라 생성되는, 방법.

청구항 64

제59항 내지 제63항 중 어느 한 항에 있어서,

상기 하나 이상의 네트워크 경로는 하나 초과의 네트워크, 하나 초과의 네트워크 유형, 하나 초과의 네트워크, 하나 초과의 네트워크 전송 매체, 또는 상이한 성능 특성들을 갖는 통신 채널들을 포함하는, 방법.

청구항 65

방법으로서,

제1 전자 디바이스에서,

적어도 하나의 프로세서에 의해, 제1 유형의 코딩으로 인코딩된 미디어 자산과 연관된 제1 비트스트림을 수신하는 단계;

상기 제1 비트스트림이 제2 유형의 코딩과 연관된 디코더로 디코딩되어야 한다는 결정에 따라, 상기 제1 비트스트림을 제2 유형의 선형 코딩의 제2 비트스트림으로 처리하는 단계 - 상기 제1 비트스트림을 제2 유형의 선형 코딩의 제2 비트스트림으로 처리하는 단계는,

상기 적어도 하나의 프로세서에 의해, 상기 제1 비트스트림의 필드들의 세트로부터 데이터를 추출하고, 상기 데이터를 상기 제2 비트스트림의 필드들의 제2 및 대응하는 세트에 매핑하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 제1 비트스트림으로부터 상기 미디어 자산과 연관된 메타데이터를 추출하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 제2 유형의 코딩과 연관된 디코더로 상기 제2 비트스트림을 상기 미디어 자산의 코딩되지 않은 표현으로 디코딩하는 단계

를 포함함 - ;

상기 제1 비트스트림이 제2 유형의 코딩과 연관된 디코더로 디코딩되지 않아야 한다는 결정에 따라, 상기 제1 비트스트림을 처리하는 단계 - 상기 제1 비트스트림을 처리하는 단계는,

상기 적어도 하나의 프로세서에 의해, 상기 미디어 자산과 연관된 메타데이터 및 상기 제1 유형의 코딩과 연관된 디코더로 상기 제1 비트스트림을 상기 미디어 자산의 코딩되지 않은 표현으로 디코딩하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 미디어 자산의 코딩되지 않은 표현을 상기 미디어 자산과 연관된 추출된 메타데이터를 갖는 애플리케이션에 전달하는 단계

를 포함함 -

를 포함하는, 방법.

청구항 66

비일시적 컴퓨터 판독가능한 저장 매체로서,

컴퓨팅 장치에 의해 실행될 때, 상기 컴퓨팅 장치로 하여금 제1항 내지 제65항 중 어느 한 항의 방법을 수행하게 하는 명령어들을 저장한, 비일시적 컴퓨터 판독가능한 저장 매체.

청구항 67

컴퓨팅 장치로서,

적어도 하나의 프로세서; 및

상기 적어도 하나의 프로세서에 의해 실행될 때, 상기 컴퓨팅 장치로 하여금 제1항 내지 제65항 중 어느 한 항의 방법을 수행하게 하는 명령어들을 저장한 메모리

를 포함하는, 컴퓨팅 장치.

발명의 설명

기 술 분 야

[0001] 관련 출원들에 대한 상호 참조

[0002] 본 출원은 2022년 4월 18일자로 출원된 미국 가특허 출원 제63/332,210호(참조 번호: D20228AUSP1), 2022년 4월 18일자로 출원된 미국 가특허 출원 제63/332,212호(참조 번호: D20228BUSP1), 및 2023년 2월 7일자로 출원된 미국 가특허 출원 제63/443,933호(참조 번호: D20228AUSP2)의 우선권 이익을 주장하며, 이들 각각은 그 전체가 참조로 포함된다.

[0003] 기술 분야

[0004] 본 개시내용은 일반적으로 미디어의 인코딩 및 디코딩에 관한 것이다.

배 경 기 술

[0005] 애플리케이션 계층 코딩은 하위 계층 프로토콜들, 네트워크들 및 채널들이 하나 이상의 소스로부터 하나 이상의 수신기로의 데이터의 전송의 전체 품질에 미치는 영향들을 완화하는데 이용될 수 있는 접근법이다. 네트워크를 통한 데이터 전송 레이트는 네트워크의 특성(예를 들어, 정체, 패킷 손실, 레이턴시 등)에 크게 의존한다. 애플리케이션 계층 코딩은 네트워크 특성(예를 들어, 네트워크 용량, 레이턴시 등), 특정한 이용 사례(예를 들어, 온라인 스트리밍) 및 전체 목적(예를 들어, 레이턴시 최소화)에 대한 지식을 고려하여 적절한 코딩 유형을 선택함으로써 데이터 전송을 최적화하고자 한다.

발명의 내용

[0006] 코딩된 미디어에 대한 멀티소스 방법들 및 시스템들의 실시예들이 포함된다.

[0007] 일부 실시예들에서, 방법은, 제1 디바이스에서, 미디어 자산을 나타내는 미디어 데이터를 수신하는 단계; 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 획득하는 단계; 미디어 데이터의 적어도 일부를 제1 코딩 프로세스에 따라 미디어 자산에 대응하는 코딩된 데이터로 인코딩하는 단계; 제1 코딩 프로세스와 연관된 정보에 기반하여 제1 복수의 데이터 요소와 상이한 제2 복수의 데이터 요소를 생성하는 단계; 제1 복수의 데이터 요소 및 제2 복수의 데이터 요소를 미디어 자산을 나타내는 하나 이상의 코딩된 비트스트림으로 결합하는 단계; 및 하나 이상의 네트워크 경로를 이용하여 하나 이상의 제2 디바이스에 하나 이상의 코딩된 비트스트림을 전송하는 단계를 포함한다.

[0008] 일부 실시예들에서, 제1 코딩 프로세스는 미디어 자산에 대응하는 코딩된 데이터를 획득하기 위한 선형 코딩 프로세스이다.

[0009] 일부 실시예들에서, 제1 코딩 프로세스는 미디어 데이터를 다른 선형 코딩 프로세스와 일치하는 코딩된 데이터로 변환하는 비선형 코딩 프로세스이다.

[0010] 일부 실시예들에서, 데이터 요소들은 데이터 구조들이다.

[0011] 일부 실시예들에서, 제1 복수의 데이터 요소는 동기화 데이터 요소를 포함하고, 하나 이상의 코딩된 비트스트림

각각은 동기화 데이터 요소로 시작한다.

[0012] 일부 실시예들에서, 동기화 데이터 요소는 하나 이상의 코딩된 비트스트림의 각각의 코딩된 비트스트림을 공통 소스 미디어 자산으로부터 생성된 것으로서 식별하는 식별 데이터; 하나 이상의 코딩된 비트스트림의 각각의 코딩된 비트스트림을 공통 소스 미디어 자산으로부터 생성된 다른 코딩된 비트스트림들과 동기화하기 위한 동기화 데이터; 및 하나 이상의 코딩된 비트스트림의 각각의 코딩된 비트스트림의 계보 데이터(lineage data) 중 적어도 하나를 포함한다.

[0013] 일부 실시예들에서, 계보 데이터는 각각의 코딩된 비트스트림에 포함된 코딩된 심볼들을 생성하는데 이용되는 원래의 또는 코딩된 심볼들을 식별하는 데이터, 각각의 코딩된 비트스트림에 포함된 코딩된 심볼들을 생성하기 위해 원래의 또는 코딩된 심볼들이 어떻게 결합되었는지를 식별하는 데이터, 및 각각의 코딩된 비트스트림에 포함된 코딩된 심볼들을 생성하는데 이용되는 인코딩 계수들을 식별하는 데이터를 포함한다.

[0014] 일부 실시예들에서, 이 방법은 하나 이상의 코딩된 비트스트림 전체에 걸쳐 동기화 데이터 요소들을 인터리빙하는 단계를 더 포함하고, 동기화 데이터 요소들은 하나 이상의 코딩된 비트스트림 중 하나를 수신하는 제2 전자 디바이스가 하나 이상의 코딩된 비트스트림 중 하나 내의 복수의 포인트로부터 디코딩을 개시할 수 있게 한다.

[0015] 일부 실시예들에서, 제1 복수의 데이터 요소 또는 제2 복수의 데이터 요소는 수신된 미디어 데이터와 연관된 시스템 데이터에 적어도 부분적으로 기반하여 생성된다.

[0016] 일부 실시예들에서, 이 방법은 제1 복수의 데이터 요소를 획득하는 단계가 수신된 미디어 데이터 또는 수신된 미디어 데이터와 연관된 시스템 데이터로부터 제1 복수의 데이터 요소를 생성하는 단계를 포함하는 것을 더 포함한다.

[0017] 일부 실시예들에서, 제1 코딩 프로세스에 따라 미디어 데이터를 인코딩하는 단계는 미디어 데이터를 복수의 원래의 심볼로 분할하고 복수의 원래의 심볼에 기반하여 복수의 코딩된 심볼을 생성하는 단계를 포함한다.

[0018] 일부 실시예들에서, 코딩된 심볼은 네트워크 코딩 기술에 따라 복수의 원래의 심볼들의 서브세트를 선형적으로 결합함으로써 생성된다.

[0019] 일부 실시예들에서, 미디어 데이터는 선형 코딩된 비트스트림을 포함한다.

[0020] 일부 실시예들에서, 제1 복수의 데이터 요소를 획득하는 단계는 미디어 데이터로부터 메타데이터를 추출하는 단계; 및 추출된 메타데이터로부터 제1 복수의 데이터 요소를 계산하는 단계를 포함한다.

[0021] 일부 실시예들에서, 선형 디코딩을 수행함이 없이 미디어 데이터로부터 메타데이터가 추출된다.

[0022] 일부 실시예들에서, 제1 코딩 프로세스에 따라 미디어 데이터를 인코딩하는 단계는 코딩된 비트스트림 내의 선형 코딩된 심볼들을 미디어 자산을 나타내는 새로운 선형 코딩된 심볼들로 재인코딩하는 단계를 포함한다.

[0023] 일부 실시예들에서, 이 방법은, 결합하기 전에, 제1 크기를 갖는 하나 이상의 데이터 요소를 제1 크기보다 작은 제2 크기를 갖는 데이터 요소로 분할하는 단계를 더 포함한다.

[0024] 일부 실시예들에서, 이 방법은 제2 복수의 데이터 요소 중 하나 이상의 데이터 요소에 해시 데이터를 추가하여 하나 이상의 비트스트림을 수신하는 디바이스에 의한 검증, 인증 또는 무결성 중 적어도 하나를 가능하게 하는 단계를 더 포함한다.

[0025] 일부 실시예들에서, 이 방법은 미디어 자산에 대응하는 파일, 미디어 자산의 블록, 미디어 자산의 세그먼트 및 미디어 자산의 패킷 중 하나 이상과 연관되는 제2의 복수의 데이터 요소 중 하나 이상의 데이터 요소에 해시 데이터를 추가하는 단계를 더 포함한다.

[0026] 일부 실시예들에서, 결합하는 단계는 디렉토리 데이터 요소를 하나 이상의 코딩된 비트스트림에 삽입하는 단계를 포함한다.

[0027] 일부 실시예들에서, 디렉토리 데이터 요소는 수신 디바이스가 각각의 비트스트림 내의 서브아톰들(subatoms)을 식별하는 것을 돕는데 이용된다.

[0028] 일부 실시예들에서, 하나 이상의 코딩된 비트스트림은 제1 유형의 데이터 요소를 포함하는 제1 코딩된 비트스트림 및 제1 유형의 데이터 요소를 포함하지 않는 제2 코딩된 비트스트림을 포함한다.

[0029] 일부 실시예들에서, 클라이언트 상에서 디코더를 초기화하는데 이용되는 데이터 요소들 또는 서브아톰들은 레이

턴시 민감 데이터가 아니다.

[0030] 일부 실시예들에서, 비트스트림은 sync() 서브아톰, 및 bitstream_header(), block_header(), encoder_content_info() 또는 media_segment_info() 서브아톰들 중 적어도 하나를 포함한다.

[0031] 일부 실시예들에서, 데이터 요소들 또는 서브아톰들은 인코딩된 데이터 또는 레이턴시 민감 데이터를 운반한다.

[0032] 일부 실시예들에서, 비트스트림은 packet() 서브아톰들 및 sync() 서브아톰들을 포함하지만, bitstream_header(), block_header(), encoder_content_info() 또는 media_segment_info() 서브아톰들 중 적어도 하나를 포함하지 않는다.

[0033] 일부 실시예들에서, 전송하는 단계는 제1 유형의 통신 채널을 이용하여 제1 코딩된 비트스트림을 전송하는 단계; 및 제1 유형과 상이한 제2 유형의 통신 채널을 이용하여 제2 코딩된 비트스트림을 전송하는 단계를 포함한다.

[0034] 일부 실시예들에서, 제2 유형의 통신 채널은 더 높은 레이턴시 채널, 더 낮은 대역폭 채널, 신뢰성 있는 채널, 또는 제1 유형의 전송 프로토콜과는 상이한 제2 유형의 전송 프로토콜을 이용하는 채널 중 적어도 하나이다.

[0035] 일부 실시예들에서, 방법은, 제1 디바이스에서, 미디어 자산과 연관된 하나 이상의 코딩된 비트스트림을 수신하는 단계; 하나 이상의 코딩된 비트스트림을 버퍼에 저장하는 단계; 하나 이상의 코딩된 비트스트림으로부터 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 추출하는 단계; 및 하나 이상의 코딩된 비트스트림으로부터, 하나 이상의 코딩된 비트스트림을 생성하는데 이용된 각각의 코딩 프로세스들과 연관되는 정보를 포함하는 제2 복수의 데이터 요소를 추출하는 단계를 포함하는, 버퍼에 저장된 하나 이상의 코딩된 비트스트림을 처리하는 단계; 및 제2 복수의 데이터 요소에 기반하여 미디어 자산을 나타내는 디코딩된 데이터를 생성하는 단계; 및 제2 복수의 데이터 요소에 기반하여 미디어 자산을 나타내는 새롭게 코딩된 데이터를 생성하는 단계 중 하나를 수행하는 단계를 포함한다.

[0036] 일부 실시예들에서, 디코딩된 데이터를 생성하는 단계는 코딩된 데이터를 생성하는데 이용되는 인코딩 유형의 결정에 따라 복수의 디코더의 각각의 디코더로 코딩된 데이터를 디코딩하는 단계를 포함한다.

[0037] 일부 실시예들에서, 인코딩 유형은 추출된 제1 복수의 데이터 요소로부터 결정된다.

[0038] 일부 실시예들에서, 인코딩 유형은 추출된 제2 복수의 데이터 요소로부터 결정된다.

[0039] 일부 실시예들에서, 인코딩 유형은 추출된 제2 복수의 데이터 요소를 이용하지 않고 제1 복수의 데이터 요소로부터 결정된다.

[0040] 일부 실시예들에서, 인코딩 유형은 추출된 제1 복수의 데이터 요소를 이용하지 않고 제2 복수의 데이터 요소로부터 결정된다.

[0041] 일부 실시예들에서, 이 방법은 디코딩된 데이터 및 제1 복수의 데이터 요소로부터 미디어 자산에 대응하는 소비가능한 데이터를 생성하는 단계를 더 포함한다.

[0042] 일부 실시예들에서, 소비가능한 데이터는 제1 전자 디바이스 상에서 실행되는 애플리케이션 또는 애플리케이션에 의해 지원되는 미디어 파일 유형과 호환가능하다.

[0043] 일부 실시예들에서, 디코딩된 데이터를 생성하는 단계는 선형 코딩을 수행하지 않고 제1 유형의 코딩된 데이터를 제2 유형의 코딩된 데이터로 처리하는 단계; 및 제2 유형의 코딩된 데이터를 디코딩할 수 있는 디코더를 이용하여 제2 유형의 코딩된 데이터를 디코딩하는 단계를 더 포함한다.

[0044] 일부 실시예들에서, 코딩된 데이터를 처리하는 단계는 계수들을 정렬하거나 코딩된 데이터를 공통 유한 필드에 투영하는 코딩 통합기에 의해 수행된다.

[0045] 일부 실시예들에서, 새롭게 코딩된 데이터를 생성하는 단계는 코딩된 데이터를 생성하는데 이용되는 인코딩 유형의 결정에 따라 복수의 리인코더(re-encoder)의 각각의 리인코더로 제2 복수의 데이터 요소에 포함된 코딩된 데이터를 처리하는 단계를 포함한다.

[0046] 일부 실시예들에서, 인코딩 유형은 추출된 제1 복수의 데이터 요소로부터 결정된다.

[0047] 일부 실시예들에서, 인코딩 유형은 추출된 제2 복수의 데이터 요소로부터 결정된다.

[0048] 일부 실시예들에서, 인코딩 유형은 추출된 제2 복수의 데이터 요소를 이용하지 않고 제1 복수의 데이터 요소로

부터 결정된다.

[0049] 일부 실시예들에서, 인코딩 유형은 추출된 제1 복수의 데이터 요소를 이용하지 않고 제2 복수의 데이터 요소로부터 결정된다.

[0050] 일부 실시예들에서, 새롭게 코딩된 데이터를 생성하는 단계는 선형 디코딩을 수행하지 않고 제1 유형의 코딩된 데이터를 제2 유형의 코딩된 데이터로 처리하는 단계; 및 제2 유형의 코딩된 데이터를 인코딩할 수 있는 리인코더를 이용하여 제2 유형의 코딩된 데이터를 재인코딩하는 단계를 포함한다.

[0051] 일부 실시예들에서, 코딩된 데이터를 처리하는 단계는 계수들을 정렬하거나 코딩된 데이터를 공통 유한 필드에 투영하는 코딩 통합기에 의해 수행된다.

[0052] 일부 실시예들에서, 이 방법은 새롭게 코딩된 데이터를 저장하거나 새롭게 코딩된 데이터로부터 하나 이상의 비트스트림을 생성하는 단계를 더 포함한다.

[0053] 일부 실시예들에서, 이 방법은 버퍼에 저장된 하나 이상의 코딩된 비트스트림을 처리하기 전에, 각각이 제1 크기를 갖는, 제1 복수의 데이터 요소 또는 제2 복수의 데이터 요소의 제1 수량의 데이터 요소들을 제1 크기보다 큰 제2 크기를 갖는 더 적은 수량의 데이터 요소들로 결합하는 단계를 더 포함한다.

[0054] 일부 실시예들에서, 결합하는 단계는 CHUNKED_SUBATOM 서브아톰 내의 데이터에 기반한다.

[0055] 일부 실시예들에서, 이 방법은, 디코딩 또는 재인코딩 전에, 제2 복수의 데이터 요소로부터 무결성 데이터를 추출하는 단계; 각각의 무결성 데이터에 기반한 각각의 데이터 요소들에 대한 성공적인 검증의 결정에 따라, 디코딩 또는 재인코딩 중 적어도 하나를 수행하는 단계; 및 각각의 무결성 데이터에 기반한 각각의 데이터 요소들에 대한 성공적이지 않은 검증의 결정에 따라, 디코딩 또는 재인코딩 중 적어도 하나를 수행하지 않는 단계를 더 포함한다.

[0056] 일부 실시예들에서, 무결성 데이터는 해시 데이터를 포함하고, 해시 데이터의 검증은 키 정보에 기반한다. 일부 실시예들에서, 키 정보는 공개 키 또는 개인 키 중 적어도 하나를 포함한다.

[0057] 일부 실시예들에서, 키 정보는 인증 서버로부터 수신된다.

[0058] 일부 실시예들에서, 이 방법은, 디코딩 또는 재인코딩 전에, 제2 복수의 데이터 요소가 각각의 암호화된 데이터를 포함한다는 결정에 따라, 각각의 암호화된 데이터를 해독 키를 이용하여 해독하는 단계를 더 포함한다.

[0059] 일부 실시예들에서, 키 정보 또는 해시 데이터는 bitstream_encryption_key_id와 연관된다.

[0060] 일부 실시예들에서, 미디어 자산의 소유자 또는 생성자에 의해 미디어 자산과 연관된 수신된 하나 이상의 코딩된 비트스트림에 암호화가 적용되어, 비허가된 액세스/소비/디코딩/재인코딩을 방지한다.

[0061] 일부 실시예들에서, 이 방법은, 제1 전자 디바이스와 연관된 식별 정보를 제1 전자 디바이스와 상이한 제2 전자 디바이스에 전송하는 단계; 및 미디어 자산과 연관된 하나 이상의 코딩된 비트스트림의 액세스, 소비, 디코딩 또는 재인코딩을 가능하게 하는 해독 키를 제2 전자 디바이스로부터 수신하는 단계를 더 포함한다.

[0062] 일부 실시예들에서, 방법은, 제1 전자 디바이스에서, 적어도 하나의 프로세서에 의해, 제1 유형의 제1 코딩된 비트스트림을 획득하는 단계; 적어도 하나의 프로세서에 의해, 제1 유형과 상이한 제2 유형의 제2 코딩된 비트스트림을 생성하는 단계; 및 적어도 하나의 프로세서에 의해, 제2 코딩된 비트스트림을 하나 이상의 네트워크 경로를 통해 하나 이상의 제2 전자 디바이스에 전송하는 단계를 포함한다.

[0063] 일부 실시예들에서, 제1 유형의 제1 비트스트림은 제1 유형의 선형 코딩 프로토콜 또는 표준에 따라 생성된다.

[0064] 일부 실시예들에서, 제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 체계적 패킷들 및 데이터 세그먼트와 연관된 비체계적 패킷들을 포함한다.

[0065] 일부 실시예들에서, 제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 체계적 패킷들을 포함하고, 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하지 않는다.

[0066] 일부 실시예들에서, 제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하고, 데이터 세그먼트와 연관된 체계적 패킷들을 포함하지 않는다.

[0067] 일부 실시예들에서, 제2 유형의 제2 비트스트림은 제2 유형의 선형 코딩 프로토콜 또는 표준에 따라 생성된다.

[0068] 일부 실시예들에서, 하나 이상의 네트워크 경로는 하나 초과의 네트워크, 하나 초과의 네트워크 유형, 하나 초과의 네트워크, 하나 초과의 네트워크 전송 매체, 또는 상이한 성능 특성들을 갖는 통신 채널들을 포함한다.

[0069] 일부 실시예들에서, 방법은, 제1 전자 디바이스에서, 제1 유형의 코딩으로 인코딩된 미디어 자산과 연관된 제1 비트스트림을 수신하는 단계; 제1 비트스트림의 필드들의 세트로부터 데이터를 추출하고, 데이터를 제2 비트스트림의 필드들의 제2 및 대응하는 세트에 매핑하는 단계; 제1 비트스트림으로부터 미디어 자산과 연관된 메타데이터를 추출하는 단계; 및 제2 유형의 코딩과 연관된 디코더로 제2 비트스트림을 미디어 자산의 코딩되지 않은 표현으로 디코딩하는 단계를 포함하는, 제1 비트스트림이 제2 유형의 코딩과 연관된 디코더로 디코딩되어야 한다는 결정에 따라, 제1 비트스트림을 제2 유형의 코딩의 제2 비트스트림으로 처리하는 단계; 미디어 자산과 연관된 메타데이터 및 제1 유형의 코딩과 연관된 디코더로 제1 비트스트림을 미디어 자산의 코딩되지 않은 표현으로 디코딩하는 단계; 및 미디어 자산의 코딩되지 않은 표현을 미디어 자산과 연관된 추출된 메타데이터를 갖는 애플리케이션에 전달하는 단계를 포함하는, 제1 비트스트림이 제2 유형의 코딩과 연관된 디코더로 디코딩되지 않아야 한다는 결정에 따라, 제1 비트스트림을 처리하는 단계를 포함한다.

[0070] 일부 실시예들에서, 비일시적 컴퓨터 판독가능한 저장 매체는, 컴퓨팅 장치에 의해 실행될 때, 컴퓨팅 장치로 하여금 이전의 방법들 중 임의의 것을 수행하게 하는 명령어들을 저장한다.

[0071] 일부 실시예들에서, 컴퓨팅 장치는 적어도 하나의 프로세서; 및 적어도 하나의 프로세서에 의해 실행될 때, 컴퓨팅 장치로 하여금 이전의 방법들 중 임의의 것을 수행하게 하는 명령어들을 저장한 메모리를 포함한다.

[0072] 본 명세서에 개시된 다른 실시예들은 시스템, 장치 및 컴퓨터 판독가능한 매체에 관한 것이다. 개시된 실시예들의 상세들은 첨부 도면들 및 아래의 설명에 제시된다. 다른 특징들, 목적들 및 이점들은 설명, 도면들 및 청구항들로부터 명백하다.

[0073] 본 명세서에 개시된 특정 실시예들은 다음의 이점들 중 하나 이상을 제공한다. 코딩 프로세스(예를 들어, 선형 코딩 프로세스)와 연관된 데이터 요소들을 전송 전에 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 데이터 요소들과 결합하는 것은 상이한 성능 특성들을 갖는 네트워크들을 통해 다양한 처리 능력들(예를 들어, 인코딩/디코딩 능력들)을 갖는 디바이스들로의/로부터의 미디어 데이터의 보다 효율적인 전달을 가능하게 한다. 결합된 데이터는 보다 효율적인 디코딩을 제공하고 호환성을 보장하기 위해 다수의 중복 유형들의 비트스트림들을 생성하고 전송할 필요성을 감소시킴으로써, 네트워킹 및 처리 리소스들을 절약한다.

[0074] 개시된 멀티소스 방법들 및 시스템들은 공통 표현 및 인터페이스를 갖는 상이한 애플리케이션들을 지원하는 고유 비트스트림 컨테이너 포맷(이하, "개시된 비트스트림"으로도 지칭됨)을 이용하는 애플리케이션 계층 코딩을 지원한다. 개시된 비트스트림은 하나 이상의 선형 코드의 이용을 지원하는 확장가능하고 상호운용가능한 방법이 멀티미디어의 클라우드 및 네트워크 접속된 처리, 저장, 분배, 전달 및 재생의 네트워크 대면 효율 및 신뢰성을 향상시키는 것을 가능하게 한다. 개시된 비트스트림은 상이한 코딩 유형들의 표준화된 인코더 및 디코더 구현들에 따르기 위해 호환가능한 포맷으로 메타데이터 및 다른 정보(예를 들어, 코딩된 심볼들, 블록 카운트, 블록당 심볼들의 수, 코딩 행렬에 관한 정보, 코딩 행렬 계수들의 필드 크기)를 운반하도록 또한 설계된다.

[0075] 개시된 방법들 및 시스템들은 또한 하나 이상의 네트워크 및/또는 액세스 경로를 통한 단일 소스 및 멀티소스 미디어 분배를 포함하는 몇몇 코드 유형들 및 이용 사례들의 신규한 이용을 가능하게 하는 상호교환 비트스트림 포맷을 지원한다.

[0076] 개시된 방법들 및 시스템들은 또한 단일 처리 엔티티가 복수의 시그널링된 코드 유형들을 즉각적으로 끊김없이 처리 및/또는 디코딩할 수 있는 실시예를 지원한다.

도면의 간단한 설명

[0077] 도 1a 및 도 1b는 각각 선형 인코딩 및 디코딩의 개념을 도시한다.

도 2a 및 도 2b는 각각 일부 실시예들에 따른, 비트스트림 인코딩 및 디코딩의 블록도들이다.

도 3a는 일부 실시예들에 따른, 개시된 비트스트림의 데이터 요소들의 계층적 조직 구조를 도시한다.

도 3b는 일부 실시예들에 따른, 개시된 비트스트림의 서브아톰에서의 예시적인 데이터 요소들을 도시한다.

도 4는 일부 실시예들에 따른, 미디어 처리 및 전달 시스템의 블록도이다.

도 5는 일부 실시예들에 따른, 도 4에 도시된 비트스트림 생성기의 블록도이다.

도 6은 일부 실시예들에 따른, 도 4에 도시된 비트스트림 디코더의 블록도이다.

도 7은 일부 실시예들에 따른, 비트스트림 리인코더의 블록도이다.

도 8a는 일부 실시예들에 따른, 통합 디코더의 블록도이다.

도 8b는 일부 실시예들에 따른, 디코더 스위치를 갖는 통합 디코더의 블록도이다.

도 9a는 일부 실시예들에 따른, 통합 비트스트림 리인코더의 블록도이다.

도 9b는 일부 실시예들에 따른, 리인코더 스위치를 갖는 통합 비트스트림 리인코더의 블록도이다.

도 10은 일부 실시예들에 따른, 다른 코딩 비트스트림 표준들을 따르는 비트스트림 매핑을 도시한다.

도 11은 일부 실시예들에 따른, 멀티소스 전달의 비트스트림 예이다.

도 12는 일부 실시예들에 따른, 멀티소스, 다중 경로 전달 시스템을 도시한다.

도 13은 일부 실시예들에 따른, 낮은 레이턴시 유니캐스트 비트스트림 전달 시스템을 도시한다.

도 14a는 일부 실시예들에 따른, 낮은 레이턴시 유니캐스트 인코딩 및 디코딩을 위한 시스템의 블록도이다.

도 14b는 일부 실시예들에 따른, 대역외 가변 데이터 크기의 처리를 도시한다.

도 15a는 일부 실시예들에 따른, 별개의 선형 인코더-디코더 쌍들을 이용한 낮은 레이턴시 유니캐스트 인코딩을 도시한다.

도 15b는 일부 실시예들에 따른, 교차 생성 디코더를 이용한 낮은 레이턴시 유니캐스트 인코딩을 도시한다.

도 16은 일부 실시예들에 따른, 디지털 저작권 관리(DRM)를 위해 개시된 비트스트림을 이용하는 것을 도시한다.

도 17은 일부 실시예들에 따른, 개시된 비트스트림에 의해 인에이블된 데이터 인증을 갖는 분산 미디어 전달 시스템의 블록도이다.

도 18은 일부 실시예들에 따른, 비선형 또는 선형 코딩된 데이터로부터 비트스트림을 생성하는 프로세스의 흐름도이다.

도 19는 일부 실시예들에 따른, 디코딩/재인코딩 프로세스의 흐름도이다.

도 20은 일부 실시예들에 따른, 다운스트림 호환성을 위해 새로운 코드 유형을 변환하는 프로세스의 흐름도이다.

도 21은 일부 실시예들에 따른, 다운스트림 호환성을 위해 새로운 코드 유형을 변환하는 다른 프로세스의 흐름도이다.

도 22는 도 1 내지 도 21을 참조하여 설명된 시스템들 및 방법들을 구현하기에 적합한 예시적인 하드웨어 아키텍처의 블록도이다.

도면들에서, 디바이스들, 유닛들, 명령어 블록들 및 데이터 요소들을 나타내는 것들과 같은 개략적 요소들의 특정 배열들 또는 순서들은 설명의 편의를 위해 도시된다. 그러나, 본 기술분야의 통상의 기술자라면, 도면들 내의 개략적 요소들의 특정 순서 또는 배열은 처리의 특정 순서 또는 시퀀스, 또는 프로세스들의 분리가 요구됨을 암시하도록 의도되지 않는다는 것을 이해해야 한다. 또한, 도면 내의 개략적 요소의 포함은 이러한 요소가 모든 실시예들에서 요구되거나 이러한 요소에 의해 표현되는 특징들이 일부 구현들에서 다른 요소들에 포함되거나 다른 요소들과 조합되지 않을 수 있음을 암시하도록 의도되지 않는다.

또한, 실선들 또는 파선들 또는 화살표들과 같은 연결 요소들이 2개 이상의 다른 개략적 요소들 간의 연결, 관계, 또는 연관을 예시하는데 이용되는 도면들에서, 임의의 이러한 연결 요소들의 부재는 연결, 관계, 또는 연관이 존재하지 않을 수 있음을 암시하도록 의도되지 않는다. 다시 말해서, 요소들 사이의 일부 연결들, 관계들, 또는 연관들은 본 개시내용을 모호하게 하지 않도록 도면들에 도시되지 않는다. 또한, 예시의 용이함을 위해, 단일 연결 요소는 요소들 사이의 다수의 연결들, 관계들 또는 연관들을 나타내는데 이용된다. 예를 들어, 연결 요소가 신호들, 데이터, 또는 명령어들의 통신을 나타내는 경우, 이러한 요소는 통신에 영향을 미치기 위해 필요에 따라 하나 또는 다수의 신호 경로를 나타낸다는 점이 본 기술분야의 통상의 기술자에 의해 이해되어야 한

다.

다양한 도면들에서 이용된 동일한 참조 부호는 유사한 요소들을 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0078] 다음의 상세한 설명에서는, 다양한 설명된 실시예들의 철저한 이해를 제공하기 위해 다수의 특정 상세가 제시된다. 본 기술분야의 통상의 기술자에게는 다양한 설명된 구현들이 이러한 특정 상세들 없이 실시될 수 있다는 것이 명백할 것이다. 다른 경우들에서, 잘 알려진 방법들, 절차들, 컴포넌트들, 및 회로들은 실시예들의 양태들을 불필요하게 모호하게 하지 않기 위해 상세히 설명되지 않았다. 각각이 서로 독립적으로 또는 다른 특징들의 임의의 조합과 함께 이용될 수 있는 몇몇 특징들이 이하에서 설명된다.

[0079] *명명법*

[0080] 본 명세서에서 이용되는 바와 같이, "포함한다(includes)"라는 용어 및 그 변형들은 "포함하지만, 이에 제한되는 것은 아니다"를 의미하는 개방형 용어들로서 읽혀져야 한다. "또는"이라는 용어는 문맥이 명확하게 달리 표시하지 않는 한 "및/또는"으로서 읽혀져야 한다. "~에 기반하는"이라는 용어는 "적어도 부분적으로 ~에 기반하는"으로서 읽혀져야 한다. "하나의 예시적인 구현" 및 "예시적인 구현"이라는 용어는 "적어도 하나의 예시적인 구현"으로서 읽혀져야 한다. "다른 구현"이라는 용어는 "적어도 하나의 다른 구현"으로서 읽혀져야 한다. "결정된다", "결정한다" 또는 "결정하는"이라는 용어들은 획득, 수신, 컴퓨팅, 계산, 추정, 예측 또는 도출하는 것으로서 읽혀져야 한다. 또한, 이하의 설명 및 청구항들에서, 달리 정의되지 않는 한, 본 명세서에서 이용되는 모든 기술적 및 과학적 용어들은 본 개시내용이 속하는 기술분야의 통상의 기술자에 의해 일반적으로 이해되는 바와 동일한 의미를 갖는다.

[0081] 위의 정의들에 추가하여, 이하의 추가적인 정의들이 이하의 설명에 적용될 것이다.

[0082] "인코딩"은 소스 코딩된/비선형 코딩된 미디어 데이터 입력을 인코딩하는 것, 또는 선형 코딩을 이용하여 선형 코딩된 미디어 입력을 재코딩하는 것을 지칭한다.

[0083] "유한 필드"는 유한 순서를 갖는 필드이며, 여기서 필드 내의 숫자들은 산술적으로 닫혀 있고 완전하다.

[0084] "호모모픽 해시(homomorphic hash)"는 복합 데이터 블록의 해시가 (예를 들어, 개별 데이터 블록들을 선형적으로 결합 또는 연접시키는 것을 통해) 개별 데이터 블록들의 해시들을 이용하여 도출될 수 있는 암호 해싱 알고리즘들의 계열이다.

[0085] "선형 코드들"은 데이터 세그먼트들의 선형 결합들이 중복 정보로서 생성되는 중복 코드들의 계열이다. 선형 코드들의 예들은 리드-솔로몬(Reed-Solomon) 코드, RaptorQ 코드, 랜덤 선형(네트워크) 코드, 해밍(Hamming) 코드, 컨볼루션 코드, 저밀도 패리티 체크(LDPC) 코드 및 BCH(Bose-Chaudhuri-Hocquenghem) 코드를 포함하지만 이에 제한되지 않는다.

[0086] "선형 인코딩"은 데이터 세그먼트들을 선형적으로 결합함으로써 선형 중복을 생성하는 동작이다.

[0087] "선형 디코딩"은 선형 방정식들을 풀어서 원래의 데이터를 검색하는 동작이다.

[0088] "선형 재인코딩"은 선형 결합들을 디코딩한 다음 인코딩하거나 또는 추가로 선형적으로 결합함으로써 새로운 선형 중복을 생성하는 동작이다.

[0089] "메타데이터"는 수신기가 미디어 유형, 프레임 레이트, 비트레이트 등과 같은 미디어 데이터를 처리하는 것을 돕는 상위 레벨 설명 데이터이다.

[0090] "후처리"는 예를 들어 언패키징(unpackaging) 및/또는 압축해제(decompression)를 적용함으로써 전달된 데이터를 소비가능한 미디어로 변환하는 처리이다.

[0091] "전처리"는 미가공 미디어 데이터를 적합한 포맷으로 처리하는 것이다. 예를 들어, 미가공 미디어 데이터에 소스 코딩 기술들(예를 들어, 압축)을 적용하는 것, 및/또는 HLS(HTTP-like streaming) 및 DASH(dynamic adaptive streaming over HTTP)와 같은 특정 표현들을 이용하여 미디어 데이터를 패키징하는 것이다.

[0092] *애플리케이션 계층 코딩 개요*

[0093] 독자가 개시된 실시예들을 더 잘 이해하는 것을 돕기 위해, 애플리케이션 계층 코딩, 특히 선형 코딩의 개요가 아래에 논의된다.

[0094] 대다수의 애플리케이션 계층 코드들은 선형 코드들이며, 데이터를 인코딩하는 것은 통상적으로 3개의 기본 단계들을 따른다: 1) 데이터를 다수의 블록들로 분할하며, 여기서 블록들의 수는 데이터의 크기에 따라 작거나(예를 들어, 1) 클 수 있고; 2) 각각의 블록을 다수의 소스 심볼들(X_i)로 분할하며, 여기서 i는 블록 내의 원래의 심볼의 인덱스이고, 각각의 소스 심볼은 지정된 크기(예를 들어, 바이트 단위)이고, 최종 소스 심볼은 필요한 경우 제로 값 바이트들로 패딩되며, 소스 심볼들의 수는 통상적으로 애플리케이션 및 이용된 코딩 유형에 기반하여 선택되고; 3) 소스 심볼들을 인코딩하여 (코딩 유형에 따라) 동일하거나 더 많은 수의 코딩된 심볼들(Y_i)을 생성한다.

[0095] 선형 코드 인코딩 및 선형 코드 디코딩 단계들의 수학적 표현들이 도 1a 및 도 1b에 각각 도시되어 있다. 도시된 예에서, 원래의 데이터 파일 또는 스트림은 다수의 블록들로 분할되는 것으로 도시되어 있다. 이 예가 하나의 블록의 선형 인코딩에 초점을 맞추고 있지만, 선형 코드 인코딩/디코딩 단계들은 데이터 파일 또는 스트림 내의 각각의 블록에 대해 반복된다.

[0096] 도 1a를 참조하면, 일반적인 선형 코드가 주어지면 데이터를 인코딩하는 단계들은 다음과 같다.

[0097] 단계 1: 각각의 데이터 블록을 다수의 소스 심볼로 분할한다. 이 예에서, 입력 블록 데이터로부터 4개의 소스 심볼이 생성된다.

[0098] 단계 2: 소스 심볼들의 벡터($\hat{X}$)를 공식화한다.

[0099] 단계 3: 코딩 행렬(G)을 생성하며, 행렬 내의 각각의 요소는 갈르와 필드 GF{q}로부터 선택되고, q = 2m이다. 이 예에서, 코딩 행렬은 6 x 4이다.

[0100] 단계 4: 코딩 행렬($\mathbf{G}$)을 입력 소스 심볼들($\hat{X}$)과 곱함으로써 코딩된 심볼들(Y)을 계산한다. GF{2}의 갈르와 필드가 이용되고, $\hat{y}_1$에 대한 계수 벡터가 $[a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}]$ = [1 0 0 0]인 것으로 선택되는 경우, 계수 벡터가 오직 하나의 비-제로 계수를 가지므로, 제1 코딩된 심볼(y_1)은 제1 소스 심볼(x_1)과 등가이다. 이 경우, y_1은 체계적 심볼로서 지칭될 수 있다.

[0101] 인코딩 후에, 코딩 행렬의 코딩된 심볼들 및 코딩 계수들은 디코더에 전송된다. 다른 실시예들에서, 인코딩된 심볼들은 네트워크, CDN 등에서 캐싱될 수 있고, 적절한 디코더를 포함하는 하나 이상의 클라이언트 디바이스로부터의 요청 시에 전달될 수 있다. 코딩 행렬의 코딩 계수들은 (a) 디코더에 전송될 수 있거나; (b) 코딩 계수들을 생성하기 위한 수단이 파라미터화되어 디코더에 전송될 수 있거나; 또는 (c) 계수들이 인코더와 디코더 양쪽 모두에서 알려지고 가정된다.

[0102] 디코더에서 코딩 계수들을 알면, 예를 들어, 가우시안 제거와 같은 접근법이 코딩된 심볼들로부터 원래의 심볼들에 대한 해를 구하는데 이용될 수 있다. 이 프로세스는 체계적 심볼들이 선형적으로 독립적이도록 보장되기 때문에 체계적 심볼들이 존재할 때 단순화된다. 최소한, 소스 심볼들의 수와 동일하거나 그보다 큰 수의 코딩된 심볼들이 디코딩하기 위해 디코더에 의해 수신되어야 한다. 일부 경우들에서, 생성된 코딩 계수들이 서로 선형적으로 독립적이지 않은 경우 더 많은 코딩된 심볼들이 요구된다.

[0103] 도 1b를 참조하면, 위의 예시적인 선형 코드를 디코딩하는 단계들은 다음과 같다.

[0104] 단계 1: 각각의 코딩된 심볼과 연관된 계수 벡터를 결정하고 수신된 코딩 계수 행렬($\mathbf{A}$)을 생성한다.

[0105] 단계 2: $\mathbf{A}$의 역수, 즉 $\mathbf{A}^{-1}$을 계산한다.

[0106] 단계 3: 수신된 코딩 계수 행렬의 역수($\mathbf{A}^{-1}$)를 수신된 코딩된 심볼들의 벡터($\hat{Y}$)와 곱함으로써 원래의 심볼들의 벡터($\hat{X}$)를 복구한다.

[0107] 단계 4: 원래의 블록 데이터를 재형성하기 위해 $\hat{X}$에서의 각각의 심볼을 연접시킨다.

[0108] *컨테이너로서의 개시된 비트스트림*

[0109] 도 2a 및 도 2b는 각각 일부 실시예들에 따른 비트스트림 인코딩 및 디코딩의 블록도들이다. 개시된 비트스트

림(도 3a, 도 3b 참조)은 상이한 애플리케이션 계층 코딩 유형들을 지원하는 컨테이너로서 간주될 수 있다. 특정 코딩 유형은 특정 이용 사례에 대해 선택될 수 있다. 많은 코딩 유형들이 상이한 종류들의 비트스트림들을 생성하는 그들 자신의 포맷들을 갖지만, 개시된 비트스트림의 목적은 공통 표현 및 인터페이스를 갖는 상이한 애플리케이션들을 지원하는 것이다.

[0110] 애플리케이션 계층 코딩이 이용될 때, 정보는 원래의 데이터를 복구하기 위해 수신기에 전송된다. 예를 들어, 코딩된 심볼들에 더하여, 블록 카운트, 블록당 심볼들의 수, 코딩 행렬에 관한 정보, 및 코딩 행렬 계수들의 필드 크기가 수신기에 전송된다. 도 3a, 도 3b에 도시된 개시된 비트스트림은 상이한 코딩 유형들의 표준화된 인코더 및 디코더 구현들에 따르도록 호환가능한 포맷으로 메타데이터 및 이 정보를 운반하도록 설계된다. 비트스트림 매퍼는 애플리케이션 계층 인코더에 의해 생성된 정보를 개시된 비트스트림에 의해 지원되는 데이터 요소 유형들로 변환하는데 이용된다.

[0111] 도 2a를 참조하면, 시스템(200a)은 미디어 파일(201), 애플리케이션 계층 인코더들(202a...202n), 비트스트림 매퍼(203) 및 비트스트림 인코더(204)를 포함한다. 비트스트림 매퍼(203)는 비트스트림 인코더들(202a...202n)에 의해 생성된 정보를 비트스트림 인코더(204)에 의해 생성된 개시된 비트스트림에 의해 지원되는 데이터 요소 유형들로 변환한다. 비트스트림 인코더(204)는 비트스트림 매퍼(203)에 의해 출력된 지원되는 데이터 요소 유형들 및 콘텐츠 메타데이터(205)를 포함하는 개시된 비트스트림을 생성한다.

[0112] 도 2b를 참조하면, 시스템(200b)은 비트스트림 디코더/파서(206), 디매퍼(207) 및 애플리케이션 계층 인코더(208a...208n)를 포함한다. 비트스트림 디코더/파서(206)는 콘텐츠 메타데이터(205)를 이용하여 수신된 개시된 비트스트림을 디코딩하고 개시된 비트스트림 내의 데이터 요소를 파싱한다. 비트스트림 디매퍼(207)는 데이터 요소 유형을 표준화된 애플리케이션 계층 디코더(208a...208n)에 의해 요구되는 것으로 다시 변환한다. 애플리케이션 계층 디코더(208...208n)의 출력은 콘텐츠 미디어 데이터(205)와 함께 미디어 파일(209)로 결합된다.

[0113] *예시적인 개시된 비트스트림 구조*

[0114] 도 3a는 일부 실시예들에 따른, 개시된 비트스트림(301)의 계층적 조직(300)을 도시한다. 개시된 비트스트림(301)은 기존의 방법들에 의해 현재 제공되는 것보다 멀티미디어 애플리케이션들에 대한 여러 새롭고, 더 효율적이고, 신뢰성 있는 저장, 캐싱, 분배, 및 전달 패러다임들을 가능하게 하는 확장가능한 컨테이너를 제공한다. 이러한 새로운 멀티미디어 애플리케이션들은 단일 및/또는 복수의 통신 채널/네트워크를 통해 미디어를 분배하기 위한 오늘날의 소스 코딩 및 고도로 복제적인 저장 방법들을 이용할 수 없다. 더 구체적으로는, 개시된 비트스트림(301)은 하나 이상의 선형 코드의 이용을 지원하는 확장가능하고 상호운용가능한 방법이 멀티미디어의 클라우드 및 네트워크 접속된 처리, 저장, 분배, 전달 및 재생의 네트워크 대면 효율 및 신뢰성을 향상시키는 것을 가능하게 한다.

[0115] 예시적인 실시예에서, 소스 코딩된 멀티미디어 자산은 하나 이상의 선형 코딩 프로세스에 의해 하나 이상의 기능적으로 동등한 이진 표현 또는 "비트스트림"에 매핑되어, 도 3a에 도시된 개시된 비트스트림(301)의 포맷에 따라 복수의 기능적으로 동등한(FE) 비트스트림들을 생성한다. 일부 실시예들에서, 선형 코딩 프로세스에 의해 생성된 FE 비트스트림들 각각은 SYNC 서브아톰 데이터 구조(302)로 시작한다. SYNC 데이터 구조(302)에 정의된 정보는 개시된 비트스트림(301)에 동기화되고, 이를 식별하고, 해석하기 위해 후속 프로세스들에 이용된다. 일부 실시예들에서, SYNC 데이터 구조(302) 및 SYNC 데이터 구조(302)에 정의된 정보는 또한 서브아톰(비트스트림 데이터 구조) 내에 포함될 수 있고, 재동기화를 요구하는 애플리케이션들을 위해 개시된 비트스트림(301) 내의 임의의 포인트에서 주입될 수 있다. SYNC 데이터 구조(302) 내에 포함된 정보는, 미래의 선형 코드 유형들 및 서브아톰들을 갖는 미래의 애플리케이션들과의 순방향 호환성을 포함하지만 이에 제한되지 않는, 개시된 실시예들에서 설명된 애플리케이션들을 위해 생성된 비트스트림들에 대한 범용 엔트리 포인트를 제공한다.

[0116] SYNC 데이터 구조(302)에 이어서, 각각의 개시된 비트스트림(301)은 하나 이상의 애플리케이션을 인에이블하는데 필요한 정보를 포함하는 복수의 서브아톰 데이터 요소들(303a...302n)(이하, "서브아톰들"이라고 함)을 포함한다. 각각의 서브아톰(303a...303n)은 개시된 비트스트림(301)이 하나 이상의 애플리케이션을 인에이블하는데 필요한 정보 및 하나 이상의 고유 서브아톰을 설명 및 운반할 수 있게 하는 subatom_id 필드(도시되지 않음)를 포함한다. 위에서 언급한 바와 같이, 개시된 subatom_id 비트필드의 이용은 개시된 비트스트림(301)의 포맷에 대한 여러 고유 서브아톰 데이터 구조들의 정의 및 이용을 가능하게 한다.

[0117] 도 3a에 도시된 바와 같이, 각각의 서브아톰(303a, 303b, 303c...303n)은 상이한 유형들의 데이터로 채워질 수 있는 일반 컨테이너이다. 서브아톰이 하나의 유형의 데이터만을 운반할 수 있기 때문에, 서브아톰 내에서 운반

되는 데이터의 유형은 서브아톰 유형 및 연관된 데이터 구조를 정의한다. 예를 들어, BITSTREAM_HEADER 서브아톰(305)은 비트스트림 헤더 동조 구조(BITSTREAM_HEADER)를 운반하는 서브아톰(303a) 구문 구조의 인스턴스이다.

[0118] 이하에서, 서브아톰들 및 그 각각의 비트필드들은 보다 일반적으로 "데이터 요소들"이라고 지칭될 수 있다. 그에 추가하여, 서브아톰 유형은 그 이름(예컨대, 비트스트림 서브아톰) 또는 그 연관된 구문 구조(BITSTREAM_HEADER())에 의해 지칭될 수 있다. 일부 실시예들에서, 각각의 서브아톰(303a, 303b, 303c...303n)은 다음과 같은 구문 구조를 포함한다: SYNC(304), BITSTREAM_HEADER(305), MEDIA_SEGMENT INFO(306), ENCODER_CONTENT_INFO(307), BLOCK_HEADER(308), PACKET(309), BLOCK_DIRECTORY(310), CHUNKED_SUBATOM(311) 및 PACKET_HEADER_ONLY(312). 유효한 개시된 비트스트림(301)이 단일의 서브아톰을 포함할 수 있지만, 디코더는, 개시된 비트스트림(301)을 성공적으로 디코딩하기 위해, BITSTREAM_HEADER, BLOCK_HEADER(들) 및 PACKET(들)를 포함하는 다수의 서브아톰들을 수신한다. 신택스는 추가의 서브아톰 데이터 유형들이 새로운 애플리케이션들에 대해 미래에 정의될 수 있게 한다.

[0119] 도 3b는 일부 실시예들에 따른, 개시된 비트스트림(301) 컨테이너 포맷의 각각의 서브아톰에서의 데이터 요소들을 추가로 도시한다. 아래에 설명되는 서브아톰의 고유 인스턴스들에 의해 운반되는 데이터는 다른 서브아톰들에 독립적이거나 종속적일 수 있다. 서브아톰의 정확한 파싱/해석 및/또는 디코딩은 이전 서브아톰 인스턴스로부터의 데이터가 정확하게 파싱 및/또는 처리될 것을 요구할 수 있다. 서브아톰 데이터 요소들은 아래에 설명된다.

[0120] SYNC 서브아톰(304)은 동일한 소스 미디어 자산으로부터 생성된 2개의 비트스트림들을 식별하고 동기화하는데 이용된다. SYNC 서브아톰(304) 데이터 요소들(예를 들면, 비트필드들)의 시맨틱스/이용의 일부 예들이 이하에 설명된다.

[0121] • syncword: 개시된 비트스트림이 개시된 비트스트림(301)의 포맷을 준수한다는 것을 식별하는데 이용되는 고유 바이트 시퀀스이다.

[0122] • version: 후속하는 비트스트림 신택스의 특정 버전이다. 이 비트필드는 비트스트림이 역방향 및 순방향 호환 가능한 방식으로 확장될 수 있게 한다.

[0123] • content_encode_uuid: 일 예에서, (예를 들어, 인코더들, 디코더들, 리인코더들, 파서들, 프로세서들 등을 포함하는) 멀티소스 미디어 시스템이 미디어 자산의 특정 '인코딩'을 암시적으로 시그널링할 수 있게 하는 고유 식별자이다. 일 예에서, 이 정보는 다운스트림 프로세스들이 하나 이상의 인코더 또는 리인코더에 의해 생성된 인코딩된 비트스트림(들)의 계보를 효율적으로(그리고 독립적으로) 결정하는데 유용하다. 이러한 레벨의 암시적인 계보 시그널링을 갖는 인코딩된 비트스트림은 (네트워크 내의 저장소/캐시들을 포함하는 복수의 노드들에 걸쳐 분배되고 이용가능한) 모든 코딩된 미디어 소스들이 미디어의 고유하고 기능적으로 동등한 코딩된 표현들을 포함하는 것을 보장함으로써 멀티소스 미디어 시스템이 가장 효율적으로 동작할 수 있게 한다는 점에서 유리하다.

[0124] BITSTREAM_HEADER 서브아톰(305)은, 이용되는 선형 인코딩 방법, 비트스트림 프로파일 정보는 물론, 비트스트림의 부분들이 암호화되어 있는지의 표시와 같은, 선형 인코딩된 데이터에 관한 상위 레벨 일반 파일 정보를 포함한다. bitstream_header 비트 필드들의 시맨틱스/이용의 일부 예들은 다음과 같다.

[0125] • content_source_size: 바이트 단위의 선형 인코딩된 미디어의 크기를 나타낸다.

[0126] • content_source_type: 인코딩된 스트림이 원래의 콘텐츠, 인코딩된 변형 또는 라이브 스트림으로부터 소싱되었는지와 같은, 후속하는 선형 인코딩된 미디어 소스에 관한 더 많은 정보를 나타낸다.

[0127] • linear_code_type: 콘텐츠를 인코딩하기 위해 어느 유형의 선형 코드가 이용되는지를 나타낸다.

[0128] • profile_information: 특정 비트스트림이 제약된 또는 제한된 능력들을 갖는 특정 디코더/파서에 적합한지를 효율적으로 결정하기 위해 애플리케이션들에서 유용한 특정 프로파일에 따르는 비트스트림들의 처리를 가능하게 한다.

[0129] • encrypted_coefficients: 코딩 계수 암호화가 활성인 암시적 시그널링을 가능하게 한다.

[0130] 코딩 계수들의 암호화는 디지털 저작권 관리의 형태로 간주될 수 있고, 따라서, 선형 코딩된 미디어 에센스 자체를 완전히 암호화할 필요성을 감소시키거나 제거할 수 있다.

[0131] MEDIA_SEGMENT_INFO 서브아톰(306)은, 네트워크 및 OTT(over-the-top) 미디어 스트리밍 애플리케이션들/서비스들에서, 콘텐츠가 세그먼트들, 예컨대, 오디오 및 비디오 세그먼트들로 분할되는 미디어 스트리밍 애플리케이션들에 대해 이용된다. 이하, MEDIA_SEGMENT_INFO는 또한 보다 일반적으로 "미디어 세그먼트 데이터"라고도 지칭된다. MEDIA_SEGMENET_INFO 서브아톰(306)의 구조는, 그 오디오/비디오 특성들(예컨대, 프레임 레이트, 비트레이트, 코덱 유형 등)과 같은, 세그먼트 내의 미디어에 관한 정보를 제공한다. 이 정보는 전형적으로 매니페스트 파일(예컨대, DASH 매니페스트 또는 HLS 매니페스트)에서 발견될 수 있다. 이 정보는 선형 인코딩된 데이터를 검사하기 위해 비트스트림을 디코딩해야만 하는 것을 피하기 위해 이 구조 내에 포함된다. MEDIA_SEGMENT_INFO 서브아톰(306)의 예시적인 비트 필드들은 다음과 같다.

[0132] • segment_index: 현재 서브아톰에 표현된 세그먼트의 미디어 인덱스를 식별한다.

[0133] • content_type: 오디오, 비디오, 텍스트, 또는 애플리케이션 데이터와 같은, 이 미디어 세그먼트에 포함된 멀티미디어의 유형을 나타낸다.

[0134] • aspect_ratio: 비디오의 폭과 높이 사이의 비율이다.

[0135] • frame_rate: 초당 비디오 프레임들의 수이다.

[0136] • audio_channel_config: 오디오의 구성(예를 들어, 채널들, 위치들, 객체, 가상화 등)이다.

[0137] ENCODER_CONTENT_INFO 서브아톰(307)은 기저 미디어 콘텐츠를 나타내는 고유 식별자와 함께 비트스트림을 생성하는데 이용되는 특정 선형 인코더를 식별한다. ENCODER_CONTENT_INFO 서브아톰(307)은 또한 기저 스트림/파일/블록 정보의 무결성(신뢰성)의 검증을 가능하게 하는 정보 및 디코딩될 때 기저 스트림/파일/블록 정보의 지속기간에 관한 정보를 운반한다. ENCODER_CONTENT_INFO 서브아톰(307) 비트필드들의 시맨틱스/이용의 일부 예들은 다음과 같다.

[0138] • encoder_uuid: 일 예에서, 멀티소스 미디어 시스템(예를 들어, 인코더들, 디코더들, 리코더들, 파서들, 프로세서들 등을 포함함)이 하나 이상의 인코더 또는 리코더에 의해 생성된 인코딩된 비트스트림들의 계보를 고유하게 시그널링할 수 있게 하는 고유 식별자이다. 계보의 고유 시그널링을 갖는 인코딩된 비트스트림은, (네트워크 내의 저장소/캐시들을 포함하는, 복수의 노드들에 걸쳐 분배되고 이용가능한) 모든 코딩된 미디어 소스들이 미디어의 고유하고 기능적으로 동등한 코딩된 표현들로 구성되는 것을 보장함으로써, 멀티소스 미디어 시스템이 가장 효율적으로 동작할 수 있게 하는 개시된 실시예들의 핵심 기능이다.

[0139] • content_id_type: EIDR(Entertainment Identification Registry), Ad-iD(Advertising Digital Identifier) 또는 사용자 정의 값들과 같은, 콘텐츠를 식별하는데 이용되는 식별 포맷의 유형을 나타낸다.

[0140] • content_id: content_id 비트필드에 따라 ID 값을 운반한다.

[0141] 일부 실시예들에서, 전술한 데이터 요소들은 분배 시스템(콘텐츠 중심 네트워크들을 포함함) 전체에 걸친 처리 엔티티들이 미디어를 디코딩하거나, 중앙집중형 서버와 통신하거나 또는 스트림/세그먼트 매니페스트(적용가능한 경우)를 찾을 필요 없이 기저 코딩된 미디어에 관한 정보를 효율적으로 식별할 수 있게 한다. 이 특징은 HTTP delivery.file_integrity, 즉 수신되고 디코딩된 스트림/블록이 원래의 소스의 것과 매칭하는 것을 검증하기 위한 메커니즘을 제공하는 고유 데이터 구조를 이용하지 않을 미래의 미디어 분배 패러다임들의 여러 핵심 인에이블러들 중 하나이다. delivery.file_integrity 구조는, 예를 들어, 파일/스트림/블록을 해싱하는데 이용되는 해싱 함수, 해시 크기, 및 실제 해시 값을 포함하는 정보를 추가로 지정하는 fb_integrity() 비트필드들에 매핑된다. 일 예에서, fb_integrity() 비트필드들은 네트워크 내의 임의의 처리 노드가 코딩된 멀티소스 비트스트림들의 진정성을 독립적으로 검증할 수 있게 한다. 이 기능은 또한 모바일 디바이스들, 개인용 컴퓨터들(PC들), 셋톱 박스들, 디지털 미디어 어댑터들, 홈 게이트웨이들과 같은 전통적으로 신뢰할 수 없는 미디어 소스들의 이용이 멀티소스 코딩된 미디어를 저장하기 위해 활용되는 것을 가능하게 한다.

[0142] BLOCK_HEADER(308)은 블록 크기, 심볼 크기, 심볼들의 수, 및 유한 필드 크기와 같은, 데이터의 블록 내의 데이터에 적용되는 선형 인코딩 파라미터들에 관한 정보를 포함한다. BLOCK_HEADER 데이터 요소들의 시맨틱스/이용

의 일부 예들은 다음과 같다.

[0143] • block_index: 멀티소스 코딩된 미디어 세그먼트는 블록들로 분할될 수 있고, 여기서 각각의 블록은 별개로 인코딩된다. 이 필드는 블록의 인덱스를 나타낸다.

[0144] • symbol_size: 바이트 단위의 체계적 및 선형 코딩된 심볼들의 크기이다.

[0145] • number_of_symbols: 선형 코딩을 위해 블록이 분할되는 코딩된 심볼들의 수를 시그널링한다.

[0146] • field_size: 선형 코딩이 동작하는 유한 필드의 크기이다.

[0147] • 암호화 파라미터들: 이들 필드들은 암호화된 코딩 계수들을 해독하는데 이용된다.

[0148] • prng_seed: 선형 코드는 현재 블록에 대한 랜덤화된 코딩 결정들을 생성하기 위해 의사 난수 생성기를 이용할 수 있다. prng_seed는 디코더 또는 리코더에서 의사 난수 생성기를 초기화하는데 이용되는 시드이다.

[0149] • block_integrity: 수신되고 디코딩된 블록이 원래의 소스의 것과 매칭하는 것을 검증하기 위한 메커니즘을 제공하는 구조이다. 이 구조는 블록을 해싱하는데 이용되는 해싱 함수, 해시 크기, 및 실제 해시와 같은 정보를 포함할 수 있다. 일 예에서, 구현은 SHA1 또는 호모모픽 해싱을 이용하여 현재 블록 내의 모든 체계적 심볼들을 해싱하고, 그 다음 머클 트리와 같은 해시 집성기를 이용하여 심볼 해시를 집성하고 블록 해시를 획득할 수 있다.

[0150] PACKET 서브아톰(309)은 선형 인코딩된 데이터의 기본 부분, 즉 코딩된 심볼을 포함한다. 이것은 또한 이것이 체계적 심볼인지 여부, 윈도우 크기, 및 코딩 계수들과 같은, 이 코딩된 심볼에 관한 정보를 제공하는 패킷 헤더 데이터 구조를 포함한다. 일부 예시적인 PACKET 서브아톰(309) 데이터 요소들은 다음과 같다:

[0151] • packet_header: 패킷을 디코딩하는 것을 돕는 비트 필드들의 세트이며, 예를 들어, 1) block_index: 세그먼트는 블록들로 분할될 수 있고, 각각의 블록은 개별적으로 인코딩된다. 이 필드들은 이 패킷이 속하는 블록의 인덱스를 나타낸다. 2) b_systematic: 이 패킷이 체계적 심볼 또는 선형 코딩된 심볼을 운반하는지를 나타낸다. 3) poly_index: 이것은 패킷을 인덱싱하는데 이용되는 다목적 필드이다. 이것은 체계적 패킷들에 대한 원래의 심볼 인덱스를 식별하는데 이용될 수 있다. 이것은 또한 패킷 헤더 전용 서브아톰들 및 패킷 서브아톰들을 동기화하는데 이용될 수 있다. 4) coeff_vector: 체계적 패킷들을 선형적으로 결합하여 이 심볼을 생성하는데 이용되는 유한 필드 스칼라들의 시퀀스이다. 5) window_start_index: 패킷은 체계적 심볼들의 연속적인 서브세트(즉, 윈도우)를 이용하여 생성될 수 있다. 이 필드는 이 윈도우의 제1 체계적 심볼의 인덱스를 나타낸다. 6) window_start_index: 패킷은 체계적 심볼들의 연속적인 서브세트(즉, 윈도우)를 이용하여 생성될 수 있다. 이 필드는 이 윈도우의 마지막 체계적 심볼의 인덱스를 나타낸다. 7) prng_seed: 이 심볼의 계수들을 생성하는데 이용되는 의사 난수 생성기 시드이다. 8) packet_integrity: 수신된 패킷이 디코딩 또는 재인코딩을 위해 이 패킷을 이용하기 전에 정확하게 생성되는 것을 검증하기 위한 메커니즘을 제공하는 구조이다. 이 구조는 패킷을 해싱하는데 이용되는 해싱 함수, 해시 크기, 및 실제 해시와 같은 정보를 포함할 수 있다.

[0152] 일부 실시예들에서, 최소 packet_header이 PACKET 서브아톰(309)에 포함될 수 있지만 완전한 packet_header을 개별적으로/독립적으로(예컨대, 대역외, 별도의 네트워크 세션 등) 저장 및 전달할 수 있다. 최소 packet_header만으로는, 페이로드가 데이터를 디코딩하는데 이용가능할 가능성이 없을 것이다. 이 접근법은, 예를 들어, 전체 packet_header에의 액세스를 제어하는 것을 통해 디지털 저작권 관리의 추가의 간단한 방법을 가능하게 한다.

[0153] BLOCK_DIRECTORY 서브아톰(310)은 개시된 비트스트림(301) 내의 블록 헤더 서브아톰 인스턴스들의 시작을 나타내는 필드들을 포함한다. 일부 실시예들에서, 블록 디렉토리 서브아톰(310)은 다음의 필드들, 즉 동일한 개시된 비트스트림(301)에서, 바이트 단위로, BLOCK_DIRECTORY 서브아톰(310)의 끝으로부터 BLOCK_HEADER 서브아톰(들)(308)의 시작까지의 오프셋을 설명하는 block_header_subatom_offsets() 중 하나 이상을 포함한다.

[0154] CHUNKED_SUBATOM(311)은 큰 서브아톰 데이터를 더 작은 청크들로 분할하기 위한 메커니즘을 제공한다. 예를 들어, PACKET 서브아톰(309) 내의 코딩된 심볼 크기가 이용 중인 전송 프로토콜에 대해 너무 크다면(예를 들어, UDP 패킷이 약 1400 바이트의 페이로드를 갖는다면), PACKET 서브아톰(309)은 전송 요건들에 맞는 더 작은 청크들로 분할될 수 있다. 각각의 청크는 그 후 하나의 CHUNKED_SUBATOM(311)에 의해 운반된다. 일부 예시적인 비

트 필드들은 다음과 같다. chunked_segment_id: 동일한 서브아톰 데이터의 상이한 부분들을 포함하는 것으로서 일련의 chunked_subatom 인스턴스들을 식별한다. 서브아톰 데이터가 청킹될 때, 생성된 모든 CHUNKED_SUBATOM(311) 인스턴스들은 동일한 chunk_segment_id 필드를 가질 것이다.

[0155] • chunked_segment_index: chunked_subatom 인스턴스 내에 포함된 서브아톰 데이터의 청크 세그먼트의 인덱스를 식별한다. 파서는 청킹된 서브아톰 데이터를 그 원래의 순서로 재조립하기 위해 이 정보를 이용할 수 있다.

[0156] • num_chunked_segments: (주어진 chunk_segment_id와 연관된) 서브아톰 데이터가 분할된 청크 세그먼트들의 수를 설명한다.

[0157] • original_subatom_id: 이 chunked_subatom 인스턴스 내에 포함된 세그먼트 데이터의 subatom_id를 지정한다.

[0158] • chunked_data: 이 chunked_subatom 인스턴스에 의해 운반되는 데이터의 청크 세그먼트이다.

[0159] PACKET_HEADER_ONLY 서브아톰(312)은 실제 코딩된 심볼을 포함하지 않는다는 점을 제외하고는 PACKET 서브아톰(309)과 동일하다.

[0160] *예시적인 미디어 처리 및 전달 시스템*

[0161] 도 4는 일부 실시예들에 따른 미디어 처리 및 전달 시스템(400)의 블록도이다. 시스템(400)은 전처리 블록들(401a, 401b...401n), 비트스트림 생성기들(402a, 402b...402n), 통신 네트워크들(403), 비트스트림 디코더(404) 및 후처리 블록(405)을 포함한다. 네트워크/클라우드(402)는 재처리(406) 및 비트스트림 재생성기(407)를 포함한다. 일부 실시예들에서, 시스템(400)은, 예를 들어, CDN(content delivery network) 서버, PoP(Point-of-presence) 서버, 애플리케이션 서버(예를 들어, 전자 뉴스 수집 애플리케이션, 콘텐츠 브로드캐스팅 애플리케이션, 유니캐스트 애플리케이션 등을 호스팅하는 서버), 콘텐츠 캐시들, 피어 디바이스들, 로컬 캐싱 디바이스들, 에지 디바이스들(예를 들어, 5G/6G 에지 애플리케이션 서버) 또는 클라이언트 디바이스들(예를 들어, 미디어 소비 디바이스들, 컴퓨터, 모바일 디바이스, 게임 시스템, 스트리밍 디바이스, 오디오/비디오 장비 등)과 같은 서버들 중 하나 이상에서 부분적으로 또는 완전히 구현될 수 있다.

[0162] 이 예시적인 실시예에서, 미디어 데이터는 하나 이상의 서버에 공급된다. 미디어 데이터는 이미지들, 오디오, 비디오, 게임 상태들, 게임 커맨드들 등을 포함하지만 이에 제한되지 않는 미디어 자산들로부터 획득될 수 있다. 일부 실시예들에서, 미디어 데이터는 소스 코딩된 데이터, 스트리밍 프로토콜(예를 들어, HLS, DASH 등)에 따라 패키징된 데이터이다. 일부 실시예들에서, 미디어 데이터는 선형 코딩된 데이터 또는 미디어 자산과 연관된 코딩된 비트스트림(예를 들어, 코딩된 콘텐츠 및 연관된 메타데이터를 포함하는 코딩된 비트스트림)이다. 일부 실시예들에서, 미디어 자산을 나타내는 미디어 데이터는 복수의 네트워크 경로들로부터 수신된다.

[0163] 서버들에서의 전처리 블록들(401a...401n)은 미가공 미디어 데이터를 전처리한다. 전처리의 예들은, 예를 들어, HLS 및 DASH 표준들에 의해 정의된 바와 같이, 소스 압축 기술들을 적용하는 것 및/또는 미가공 미디어 데이터를 패키징하는 것을 포함하지만, 이에 제한되지는 않는다. 서버들은 또한 비트스트림들을 생성하기 위해 각각의 비트스트림 생성기들(402a...402n)을 포함한다. 상이한 비트스트림 생성기들(402a...402n)은, 예를 들어, 상이한 포맷들 또는 상이한 페이로드들을 갖는 상이한 비트스트림들을 생성할 수 있다. 그 후, 비트스트림들은 비트스트림들을 캐싱, 라우팅 및/또는 전달할 수 있는 통신 네트워크들(403)을 이용하여 전송된다. 통신 네트워크들(403)은 또한 비트스트림들을 재전송하기 전에, 이들을 재처리(406) 및/또는 재생성(407)하는 것과 같이, 비트스트림들을 추가로 수정할 수 있는 엔티티들을 포함할 수 있다.

[0164] 비트스트림의 수신기(예컨대, 소비자 디바이스)는 하나 또는 다수의 비트스트림을 다운로드하고 비트스트림 디코더(404)를 이용하여 처리된 미디어 데이터를 복구한다. 처리된 미디어 데이터는 이어서 후처리기(405)에 의해 후처리되고 이용을 위해 준비되며, 예컨대 최종 사용자에 의해 소비되도록 판독된다. 선택적으로, 비트스트림 디코더(404)는 피드백(408)을 비트스트림 생성기(402a...402n)에 전송하여, 시스템(400)에서의 업스트림 프로세스가 통신 네트워크(403)에서의 동적 조건에 적응하는 것을 돕는다.

[0165] *예시적인 비트스트림 생성*

[0166] 도 5는 일부 실시예들에 따른 비트스트림 생성기(500)(예를 들어, 도 4의 비트스트림 생성기들(402a...402n) 중 하나)의 블록도이다. 전처리된 미디어 데이터, 및 선택적으로 결정을 도울 수 있는 시스템 정보가 비트스트림 생성기(500)에 공급된다. 비트스트림 생성기(500)는 도 3a 및 도 3b를 참조하여 설명된 복수의 서브아톰을 포

함하는 개시된 비트스트림(301)을 생성한다. 이 예시적인 실시예에서, 메타데이터 생성기(501)는 전처리된 미디어 데이터를 분석하여 SYNC 서브아톰(304) 및 MEDIA_SEGMENT_INFO 서브아톰(306)을 생성한다. 선형 인코더(502)는 예를 들어 도 1a 및 도 1b를 참조하여 설명된 바와 같이 미디어 데이터를 원래의 (소스) 심볼들로 분할한 후에 원래의 심볼들의 선형 결합들을 선형 코딩된 심볼들로서 생성함으로써 미디어 데이터에 대해 선형 인코딩을 수행한다. 선형 인코딩 정보는, 인코딩된 데이터와 함께, 도 3a 및 도 3b를 참조하여 설명된 데이터 요소들 BITSTREAM_HEADER, ENCODER_CONTENT_INFO, BLOCK_HEADER 및 PACKET 서브아톰들에 의해 운반되며, 각각의 패킷 서브아톰은 하나의 선형 코딩된 심볼에 대응한다.

[0167] 일부 실시예들에서, 보안 실러(security sealer)(503)는 파일, 데이터 블록, 및 패킷에 관한 해싱 정보를 대응하는 ENCODER_CONTENT_INFO, BLOCK_HEADER 및 PACKET 서브아톰들에 추가하는 것에 의해 선형 코딩된 미디어 데이터에 전자적으로 서명한다. 이 단계는 개시된 발명에 따라 생성된 코딩된 비트스트림들/파일들의 독립적인 검증/인증을 위해 미디어 시스템 전체에 걸쳐 후속 처리에 필요한 정보의 고유 시그널링을 가능하게 한다.

[0168] 다른 예시적인 실시예에서, 비트스트림 생성기(500)는 또한 패킷 서브아톰으로부터 패킷 헤더 정보를 분리하고, 대응하는 PACKET 서브아톰(309)에 대한 액세스를 제어하기 위해 전용 PACKET_HEADER_ONLY 서브아톰(312)을 이용하여 패킷 헤더 정보를 운반할 수 있다. PACKET 서브아톰(309) 또는 PACKET_HEADER_ONLY 서브아톰(312) 내의 코딩 계수들은 미디어 콘텐츠에 보호 계층을 추가하기 위해 암호화된다. 일부 실시예들에서, 각각의 서브아톰은 서브아톰 크기를 감소시키기 위해 서브아톰 절단기(504)에 의해 추가로 절단되고 다수의 CHUNKED_SUBATOM 서브아톰들(311)에 의해 운반될 수 있다. 이 실시예는 전송 프로토콜이 데이터 단위 크기의 제한을 가질 때 특히 유용하다.

[0169] 서브아톰들은 이어서 하나 이상의 비트스트림(들)(301)을 형성하기 위해 멀티플렉서(505)("먹서")에 의해 멀티플렉싱된다. 상이한 비트스트림들(301)은 상이한 서브아톰들을 운반할 수 있고, 상이하게 처리(예컨대, 캐싱, 전달)될 수 있다. 수신기가 각각의 비트스트림 내에서 서브아톰들을 탐색하는데 도움을 주기 위해, 멀티플렉서(505)는 BLOCK_DIRECTORY 서브아톰(310)을 생성하여 각각의 비트스트림에 즉각적으로 삽입할 수 있다.

[0170] *예시적인 개시된 비트스트림 디코딩*

[0171] 도 6은 일부 실시예들에 따른, 비트스트림 디코더(600)(예를 들어, 도 4에 도시된 비트스트림 디코더들(404))의 블록도이다. 비트스트림 디코더(600)는 개시된 비트스트림 포맷을 갖는 하나 또는 다수의 비트스트림으로부터 데이터를 수신하고, 여기서 각각의 비트스트림은 하나 이상의 서브아톰을 포함한다. 비트스트림들의 다운로드는, 미디어 소스들을 발견하고 선택하는 것, 접속들을 확립하고, 청취하고, 종료하는 것, 및 각각의 미디어 소스로부터 다운로드된 데이터의 양을 가속하는 것을 제어하는 것을 포함하지만 이에 제한되지 않는, (예를 들어, 버퍼(들)를 이용하여) 다운로드를 관리하는, 다운로더(601)에 의해 실행된다. 일부 실시예들에서, 서브아톰 파서(602)는 비트스트림을 파싱하여 수신된 CHUNKED_SUBATOM 서브아톰들(311)을 추출하고 일시적으로 캐싱하고, 청킹된 서브아톰 병합기(603)는 CHUNKED_SUBATOM 서브아톰들(311)을 병합한다. 병합된 CHUNKED_SUBATOM 서브아톰들(311)은 그 후 보안/인증/무결성 데이터가 미디어 자산에 대해 존재할 때 보안 검사기(604)에 의해 검증된다.

[0172] 검증된 서브아톰들은 디멀티플렉서(605)에 의해 디멀티플렉싱되고, 여기서 미디어 메타데이터가 추출되어 후처리기(607)(예를 들어, 오디오/비디오 디코더)에 공급되고, 선형 코딩된 데이터가 선형 디코더(606)에 공급되어 원래의 미디어 데이터를 복구한다. 디코딩된 미디어 데이터는 추출된 미디어 메타데이터와 함께 후처리를 위해 후처리기(607)에 공급되어, 소비가능한 미디어 데이터가 후처리기(607)의 출력에서 획득될 수 있게 한다.

[0173] *예시적인 개시된 비트스트림 재인코딩*

[0174] 도 7은 일부 실시예들에 따른 비트스트림 리인코더(700)의 블록도이다. 리인코더(700)는 하나 이상의 개시된 비트스트림(701)으로부터 데이터를 다운로드하고, 각각의 비트스트림(701)은 하나 이상의 서브아톰(304-309, 312)를 포함한다. 다운로드는, 미디어 콘텐츠 소스들을 발견 및 선택하고, 접속들을 확립하고, 청취하고, 종료하고, 각각의 미디어 소스로부터 다운로드된 데이터의 속도 및 양을 제어하는 것을 포함하지만 이에 제한되지 않는, (예를 들어, 버퍼(들)를 이용하여) 다운로드를 관리하는 다운로더(702)에 의해 실행된다. 서브아톰 파서(703)는 비트스트림(701)을 파싱하여 수신된 CHUNKED_SUBATOM 서브아톰들(311)을 추출 및 일시적으로 캐싱하고, 청킹된 서브아톰 병합기(704)는 CHUNKED_SUBATOM 서브아톰들(311)을 병합한다. 병합된 CHUNKED_SUBATOM 서브아톰들(311)은 그 후 미디어 자산에 대해 보안/인증/무결성 데이터가 존재할 때 보안 검사기(705)에 의해 검증된다.

[0175] 이어서, 검증된 서브아톰들은 디멀티플렉서(706)에 의해 디멀티플렉싱되고, 여기서 미디어 메타데이터는 SYNC 서브아톰(304) 및 MEDIA_SEGMENT_INFO 서브아톰(306)을 재생성하는데 이용되며, 선형 코딩된 데이터는 새로운 선형 코딩된 데이터를 생성하기 위해 선형 리인코더(707)에 공급된다.

[0176] 새로운 선형 코딩된 데이터의 생성은 다양한 방식들로 구현될 수 있다. 예를 들면, 리인코더(700)는 먼저 선형 디코딩을 적용하여 원래의 심볼들을 디코딩하고 그 후 원래의 심볼들의 새로운 선형 결합들을 새로운 선형 코딩된 심볼들로서 생성할 수 있다. 다른 예로서, 리인코더(700)는 선형 코딩된 심볼들을 직접 선형적으로 결합하여 새로운 선형 코딩된 심볼들을 생성할 수 있다.

[0177] 일부 실시예들에서, 선형 인코딩된 데이터와 함께, 선형 인코딩 정보는 BITSTERAM_HEADER 서브아톰, ENCODER_CONTENT_INFO 서브아톰, BLOCK_HEADER 서브아톰, PACKET 서브아톰 및 PACKET_HEADER_ONLY 서브아톰에 의해 운반되고, 여기서 각각의 패킷 서브아톰은 하나의 새로운 선형 코딩된 심볼에 대응한다. 보안 실러(708)는 그 후 파일, 데이터 블록, 및 패킷에 관한 해싱 정보를 위에서 설명된 대응하는 encoder_content_info, block_header 및 패킷 서브아톰들에 추가함으로써 미디어 데이터에 전자적으로 서명한다.

[0178] 일부 실시예들에서, 각각의 서브아톰은 서브아톰 크기를 감소시키기 위해 서브아톰 절단기(709)에 의해 추가로 절단되고 다수의 chunked_subatom 서브아톰들에 의해 운반될 수 있다. 이것은 전송 프로토콜이 데이터 단위 크기의 제한을 가질 때 특히 유용하다. 그 후 서브아톰들은 멀티플렉서(710)에 의해 멀티플렉싱되어 하나 또는 다수의 비트스트림(들)(711)을 형성하고 출력한다. 상이한 비트스트림들은 상이한 서브아톰들을 운반할 수 있고 상이하게 처리(예를 들어, 캐싱, 전달)될 수 있다. 수신기가 각각의 비트스트림 내에서 서브아톰들을 탐색하는 것을 돕기 위해, 멀티플렉서(710)는 block_directory 서브아톰을 생성하여 각각의 비트스트림에 삽입한다.

[0179] *예시적인 통합 디코더들*

[0180] 도 8a는 일부 실시예들에 따른, 상이한 선형 코딩 방법들을 구현한 다수의 비트스트림들로부터 미디어 데이터를 복구할 수 있는 통합 디코더(800a)의 블록도이다. 상이한 비트스트림들(801)은 비트스트림(801)의 BITSTREAM_HEADER 서브아톰(305) 내의 code_type 비트필드마다 시그널링되는 상이한 선형 코드들을 이용할 수 있다. 다운로드는, 미디어 소스들을 발견 및 선택하고, 접속들을 확립하고, 청취하고, 종료하고, 각각의 미디어 소스로부터 다운로드된 데이터의 속도 및 양을 제어하는 것을 포함하지만 이에 제한되지 않는, (예를 들어, 버퍼(들)를 이용하여) 다운로드를 관리하는 다운로더(802)에 의해 실행된다. 서브아톰 파서(803)는 비트스트림(801)을 파싱하여, 청킹된 서브아톰 병합기(804)에 의해 병합되는 수신된 CHUNKED_SUBATOM 서브아톰들(311)을 추출하고 일시적으로 캐싱한다. 병합된 CHUNKED_SUBATOM 서브아톰들(311)은 그 후 이 미디어 자산에 대해 보안 실링이 인에이블될 때 보안 검사기(805)에 의해 검증된다.

[0181] 검증된 서브아톰들은 디멀티플렉서(806)에 의해 디멀티플렉싱되고, 여기서 미디어 메타데이터가 추출되어 후처리기(809)(예를 들어, 오디오/비디오 디코더)에 공급되고, 상이한 선형 코덱들의 선형 코딩된 데이터가 코딩 통합기(807)에 공급된다. 코딩 통합기(807)는, 예를 들어, 계수들을 정렬하고/하거나 코딩된 데이터를 동일한 유한 필드에 투영함으로써, 상이한 선형 코덱들로부터의 선형 코딩된 데이터를 통합한다. 통합된 선형 코딩된 데이터는 그 후 선형 디코더(808)에 공급되어 원래의 미디어 데이터를 복구한다. 디코딩된 미디어 데이터는 그 후 추출된 미디어 메타데이터를 이용한 후처리를 위해 후처리기(809)에 공급되어, 소비가능한 미디어 데이터가 후처리기(809)의 출력에서 획득될 수 있게 한다.

[0182] 도 8b는 일부 실시예들에 따른, 디코더 스위치(810)를 갖는 통합 디코더(800b)의 블록도이다. 통합 디코더(800b)는 다수의 상이한 디코더 유형을 캡슐화하는 선형 디코더 모듈(808)을 포함한다. 모듈(808)은 디코더 스위치(810)를 통해 특정 유형의 선형 코덱에의 액세스를 가능하게 한다. 보다 구체적으로는, 디코더(800b)는 하나 이상의 비트스트림(801)으로부터 데이터를 수신하고, 여기서 각각의 비트스트림(801)은 하나 이상의 서브아톰(303)으로 구성된다(도 3a 및 도 3b 참조). 상이한 비트스트림(801)은 상이한 선형 코덱을 이용한다. 다운로드는, 미디어 소스를 발견 및 선택하는 것, 접속을 확립, 청취 및 종료하는 것, 및 각각의 미디어 소스로부터 다운로드된 데이터의 속도 및 양을 제어하는 것을 포함하지만 이에 제한되지 않는, (예를 들어, 버퍼(들)를 이용하여) 다운로드를 관리하는 다운로더(802)에 의해 실행된다.

[0183] 일부 실시예들에서, 서브아톰 파서(803)는 비트스트림(801)을 파싱하여 서브아톰들을 추출하고, 청킹된 서브아톰 병합기(804)에 의해 병합되는 수신된 CHUNKED_SUBATOM 서브아톰들(311)을 일시적으로 캐싱한다. 병합된 CHUNKED_SUBATOM 서브아톰들(311)은 그 후 보안 실링이 이 미디어 자산에 대해 인에이블될 때 보안 검사기(805)에 의해 검증된다. 완전한 서브아톰은 그 후 보안 실링이 미디어 자산에 대해 인에이블될 때 보안 검사기

(805)에 의해 검증된다.

[0184] 검증된 서브아톰들은 이후 디멀티플렉서(806)에 의해 디멀티플렉싱되고, 여기서 미디어 메타데이터가 추출되어 후처리기(809)(예를 들어, 오디오/비디오 디코더)에게 공급되고, 특정 선형 코덱을 이용하는 선형 코딩된 데이터가 통합 선형 디코더 모듈(808)에 공급된다. 통합 선형 디코더 모듈(808)은 개시된 비트스트림의 bitstream_header()에서의 code_type 비트필드마다 시그널링되는, 이용되는 선형 코덱의 유형을 식별하고, 원래의 미디어 데이터를 디코딩하기 위해 디코더 스위치(810)를 적절한 선형 디코더 유형에게 동적으로 호출하고/하거나 스위칭한다. 디코딩된 미디어 데이터는 이후 추출된 미디어 메타데이터의 도움으로 후처리를 위해 후처리기(809)에 공급되어, 소비가능한 미디어 데이터가 후처리기(809)의 출력으로부터 획득될 수 있게 한다.

[0185] *예시적인 통합 비트스트림 리인코더들*

[0186] 도 9a는 일부 실시예들에 따른, 통합 비트스트림 리인코더(900a)의 블록도이다. 리인코더(900a)는 상이한 선형 코덱들을 이용하는 하나 이상의 비트스트림으로부터 새로운(선형 독립을 포함함) 비트스트림(들)을 생성할 수 있다. 보다 구체적으로는, 리인코더(900a)는 하나 또는 다수의 비트스트림(901)으로부터 데이터를 다운로드하고, 여기서 각각의 비트스트림(901)은 하나 이상의 서브아톰(303)을 포함한다(도 3a 및 도 3b 참조). 다운로드는, 소스들을 발견 및 선택하는 것, 접속들을 확립, 청취, 및 종료하는 것, 및 각각의 미디어 소스로부터 다운로드된 데이터의 속도 및 양을 제어하는 것을 포함하지만 이들로 제한되지 않는, (예컨대, 버퍼(들)를 이용하여) 다운로드를 관리하는 다운로더(902)에 의해 실행된다. 서브아톰 파서(903)는 비트스트림(901)을 파싱하여, 이후 청킹된 서브아톰 병합기(904)에 의해 병합되는 CHUNKED_SUBATOM 서브아톰들(311)을 추출하고 일시적으로 캐싱한다. 병합된 CHUNKED_SUBATOM 서브아톰들(311)은 이어서 미디어 자산에 대해 보안 실링이 인에이블될 때 보안 검사기(905)에 의해 검증된다.

[0187] 이어서, 검증된 서브아톰들은 디멀티플렉서(906)에 의해 디멀티플렉싱되며, 여기서 미디어 메타데이터는 SYNC 서브아톰(304) 및 MEDIA_SEGMENT_INFO 서브아톰(306)를 재생성하는데 이용되고, 상이한 선형 코덱들의 선형 코딩된 데이터가 코딩 통합기(907)에 공급된다. 코딩 통합기(907)는 예를 들어 계수들을 정렬하고/하거나 코딩된 데이터를 동일한 유한 필드에 투영함으로써 상이한 선형 코덱들의 선형 코딩된 데이터를 통합한다. 통합된 선형 코딩된 데이터는 선형 리인코더(908)에 공급되어 새로운 선형 코딩된 데이터를 생성한다.

[0188] 새로운 선형 코딩된 데이터의 생성은 다양한 방식들로 구현될 수 있다. 예를 들어, 리인코더(900a)는 먼저 선형 디코딩을 적용하여 원래의 심볼들을 디코딩한 후에 원래의 심볼들의 새로운 선형 결합들을 새로운 선형 코딩된 심볼들로서 생성할 수 있다. 다른 예로서, 리인코더(900a)는 선형 코딩된 심볼들을 직접 선형적으로 결합하여 새로운 선형 코딩된 심볼들을 생성할 수 있다.

[0189] 일부 실시예들에서, 선형 인코딩된 데이터와 함께, 선형 인코딩 정보는 BITSTREAM_HEADER 서브아톰(306), ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER 서브아톰(308), PACKET 서브아톰(309) 및 PACKET_HEADER_ONLY 서브아톰(310)에서 운반되고, 여기서 각각의 PACKET 서브아톰(309)은 하나의 선형 코딩된 심볼에 대응한다. 보안 실러(909)는 그 후 파일, 데이터 블록, 및 패킷에 관한 해싱 정보를 대응하는 ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER 서브아톰(308) 및 PACKET 서브아톰(309)에 추가함으로써 미디어 데이터의 보안을 실링한다.

[0190] 일부 실시예들에서, 각각의 서브아톰은 서브아톰 크기를 감소시키기 위해 서브아톰 절단기(910)에 의해 절단되고 다수의 CHUNKED_SUBATOM 서브아톰(311)에 의해 운반될 수 있다. 이것은 전송 프로토콜이 데이터 단위 크기의 제한을 가질 때 특히 유용하다. 그 후 서브아톰들은 멀티플렉서(911)에 의해 멀티플렉싱되어 하나 이상의 비트스트림(들)(912)의 출력을 형성한다. 상이한 비트스트림들(912)은 상이한 서브아톰들을 운반할 수 있고 상이하게 처리(예를 들어, 캐싱, 전달)될 수 있다. 수신기가 각각의 비트스트림(901) 내에서 서브아톰들을 탐색하는 것을 돕기 위해, 멀티플렉서(911)는 BLOCK_DIRECTORY 서브아톰(310)을 생성하여 각각의 비트스트림(901)에 삽입할 수 있다.

[0191] 도 9b는 일부 실시예들에 따른, 리인코더 스위치(914)를 갖는 통합 비트스트림 리인코더(900b)의 블록도이다. 통합 비트스트림 리인코더(900b)는 도 9a에 도시되어 있는 개시된 통합 비트스트림 리인코더(900a)의 다른 예시적인 구현을 보여주며, 이는 다수의 유형들의 선형 코덱들의 리인코더를 구현하고 이를 선형 리인코더 모듈(913) 내에 캡슐화한다. 선형 리인코더 모듈(913)은 스위치(914)를 통해 특정 유형의 선형 코덱에 대한 액세스를 허용한다.

[0192] 보다 구체적으로, 리인코더(900b)는 하나 또는 다수의 비트스트림(901)으로부터 데이터를 다운로드하고, 각각의

비트스트림(901)은 하나 이상의 서브아톰(303)을 포함한다(도 3a 및 도 3b 참조). 다운로드는, 소스들을 발견 및 선택하는 것, 접속들을 확립, 청취, 및 종료하는 것, 및 각각의 미디어 소스로부터 다운로드된 데이터의 속도 및 양을 제어하는 것을 포함하지만 이들로 제한되지 않는, (예컨대, 버퍼(들)를 이용하여) 다운로드를 관리하는 다운로더(902)에 의해 실행된다.

[0193] 일부 실시예들에서, 서브아톰 파서(903)는 비트스트림(901)을 파싱하여 비트스트림(901)으로부터 CHUNKED_SUBATOM 서브아톰들(311)을 추출하고 일시적으로 캐싱하고, 이는 그 후 청킹된 서브아톰 병합기(904)에 의해 병합된다. 병합된 CHUNKED_SUBATOM 서브아톰들(311)은 그 후 미디어 자산에 대해 보안 실링이 인에이블될 때 보안 검사기(905)에 의해 검증된다.

[0194] 이어서, 검증된 서브아톰들은 디멀티플렉서(906)에 의해 디멀티플렉싱되고, 여기서 미디어 메타데이터는 SYNC 서브아톰(304) 및 MEDIA_SEGMENT_INFO 서브아톰들(306)을 재생성하는데 이용되며, 상이한 선형 코덱들을 갖는 선형 코더 모듈(913)의 데이터가 선형 리인코더 모듈(913)에 공급된다. 선형 리인코더 모듈(913)은 리인코더 스위치(914)를 호출하여, 예를 들어 비트스트림 메타데이터 내의 스위치 신호에 기반하여 특정 리인코더 유형을 선택한다.

[0195] 새로운 선형 코딩된 데이터의 생성은 다양한 방식들로 구현될 수 있다. 예를 들면, 리인코더(900b)는 먼저 선형 디코딩을 적용하여 원래의 심볼들을 디코딩하고 그 후 원래의 심볼들의 새로운 선형 결합들을 새로운 선형 코딩된 심볼들로서 생성할 수 있다. 다른 예로서, 리인코더(900b)는 선형 코딩된 심볼들을 직접 선형적으로 결합하여 새로운 선형 코딩된 심볼들을 생성할 수 있다.

[0196] 일부 실시예들에서, 선형 인코딩된 데이터와 함께, 선형 인코딩 정보는 BITSTREAM_HEADER 서브아톰(306), ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER 서브아톰(308), PACKET 서브아톰(309), 및 PACKET_HEADER_ONLY 서브아톰(312)에서 운반되고, 여기서 각각의 PACKET 서브아톰(309)은 하나의 선형 코딩된 심볼에 대응한다. 보안 실러(909)는 그 후 파일, 데이터 블록, 및 패킷에 관한 해싱 정보를 대응하는 ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER(308), 및 PACKET 서브아톰(309)에 추가함으로써 미디어 데이터의 보안을 실링한다.

[0197] 일부 실시예들에서, 각각의 서브아톰은 서브아톰 크기를 감소시키기 위해 서브아톰 절단기(910)에 의해 절단되고 다수의 CHUNKED_SUBATOM 서브아톰들에 의해 운반될 수 있다. 이것은 전송 프로토콜이 데이터 단위 크기의 제한을 가질 때 특히 유용하다. 그 후 서브아톰들은 멀티플렉서(911)에 의해 멀티플렉싱되어 하나 이상의 비트스트림(들)(912)의 출력을 형성한다. 상이한 비트스트림들(912)은 상이한 서브아톰들을 운반할 수 있고 상이하게 처리(예를 들어, 캐싱, 전달)될 수 있다. 수신기가 각각의 비트스트림(901) 내에서 서브아톰들을 탐색하는 것을 돕기 위해, 멀티플렉서(911)는 block_directory 서브아톰을 생성하여 각각의 비트스트림(901)에 삽입할 수 있다.

[0198] *표준 준수를 위한 예시적인 비트스트림 매핑*

[0199] 도 10은 일부 실시예들에 따른, 다른 코딩 비트스트림 표준을 준수하기 위한 비트스트림 매핑 프로세스(1000)를 나타낸다. 개시된 비트스트림(301) 및 그 구현은 또한, 이들이 이전에 지정되거나 표준화되어 있더라도, 다양한 선형 코딩 기술을 이용하여 생성된 선형 인코딩된 데이터를 운반하기 위한 컨테이너 포맷으로서 역할을 할 수 있다. 프로세스(1000)는 비트스트림 매퍼(1003) 및 비트스트림 디매퍼(1007)를 이용함으로써 기존의 선형 코드를 지원한다. 개시된 매핑 프로세스(1000)는 개시된 비트스트림이 기존의 미디어 제품에서 표준화되고 구현된 기존의 선형 코딩 방법을 계속 준수하는 것을 가능하게 한다.

[0200] 도 10을 참조하면, 보호될 미디어 데이터가 먼저 고려된 선형 코딩 기술의 기존의 표준화된 선형 인코더(1002)를 이용하여 인코딩된다. 출력은 이 기술/방법에 대한 표준화된 비트스트림이다. 비트스트림 매퍼(1003)는 이어서 이 비트스트림의 필드들을 추출하고 이들을 도 3a, 도 3b를 참조하여 설명된 포맷을 갖는 개시된 비트스트림(301)에 매핑한다. 비트스트림 매퍼(1003)는 또한 개시된 비트스트림(301)에 메타데이터 관련 필드들을 채우기 위해 애플리케이션 계층(1001)으로부터 리치 미디어 메타데이터(rich media metadata)를 추출한다. 비트스트림은 이어서 통신 네트워크들(1004)을 통해 전달할 준비가 되어 있다.

[0201] 수신기 측에서, 그리고 기존의 표준화된 디코더가 이용되는지에 따라, 2가지 옵션이 있다. 첫째, 기존의 표준화된 디코더가 이용되는 경우(1005), 비트스트림 디매퍼(1007)는 코딩 관련 데이터를, 미디어 데이터를 애플리케이션 계층(1009)으로 출력하는, 표준화된 선형 디코더(1008)에 입력되는 기존의 표준화된 비트스트림으로 다시 매핑한다. 한편, 비트스트림 디매퍼(1007)는 또한 개시된 비트스트림(301)으로부터 리치 미디어 메타데이터

를 추출하여 이를 수신기의 애플리케이션 계층(1009)으로 전달한다. 기존의 표준화된 디코더가 이용될 필요가 없거나(1005), 존재하지 않는 경우, 수신기는 비트스트림 디코더(1006)를 이용하여 수신기의 애플리케이션 계층(1009)에 대한 미디어 데이터 및 메타데이터를 디코딩한다.

[0202] 도 11은 일부 실시예들에 따른, 개시된 비트스트림 및 비트 생성기를 이용하여 생성되는, 미디어 파일의 2개의 FE 코딩된 사본의 예시적인 구현을 나타낸 것이다. 각각의 코딩된 사본은 SYNC 서브아톰(304), BITSTREAM_HEADER 서브아톰(305), MEDIA_SEGMENT_INFO 서브아톰(306), ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER 서브아톰(308), PACKET 서브아톰(309), BLOCK_DIRECTORY 서브아톰(310) 및 PACKET_HEADER_ONLY 서브아톰(312)의 전부 또는 서브세트를 포함하는, 앞서 개시된 서브아톰들의 인스턴스들을 포함한다.

[0203] 특히, 상이한 사본들 내의 BLOCK_HEADER 서브아톰(310) 및 PACKET 서브아톰들(309)은 이용된 모든 미디어 소스들에 걸쳐 PACKET 서브아톰들(309) 사이에 최대 선형 독립성을 제공하는 방식으로 생성된다. 이것은 상이한 미디어 소스들에 대해 개시된 비트스트림 생성기를 고유하게 구성함으로써 달성된다. 선형 독립성을 낳을 수 있는 예시적인 구성들은, 상이한 비트스트림 생성기들에서, 상이한 랜덤 시드들 또는 상이한 코딩 계수 벡터들의 이용을 적용하는 것을 포함한다. 이러한 방식으로, 임의의 생성된 사본으로부터의 임의의 코딩된 미디어 PACKET 서브아톰들(309)은 높은 확률로 복구를 위해 유용할 것이고, 따라서 FE 사본들(이하, "FE 사본 또는 사본들"이라고도 지칭됨)이다. FE 사본이 미디어 스트림/파일 복구를 허용하는 충분히 선형적으로 독립적인 PACKET 서브아톰들(309)을 운반하는 경우, 이 사본은 전체 FE 코딩된 사본이라고 불린다. 그렇지 않은 경우, 이것은 부분 FE 코딩된 사본이라고 불린다.

[0204] 도 12는 일부 실시예들에 따른, 개시된 비트스트림 포맷에 의해 인에이블되는 분산 멀티소스 다중 경로 전달 시스템(1200)을 나타낸 것이다. 시스템(1200)은 1) 소스-코딩된 원래의 미디어 파일을 소유하는 기점 서버(1201); 2) 캐시들의 수개의 클러스터들(1202, 1203, 1204)을 포함하며, 각각의 클러스터는 데이터 파일들을 저장, 처리(예컨대, 재인코딩 및 디코딩 등) 및 서빙할 수 있는 접속된 캐시들을 포함한다. 클러스터들의 일부 예들은 CDN, 에지 컴퓨팅 노드들의 세트, 캐싱 능력을 갖는 ISP 노드들, 피어 사용자들의 세트, 상이한 유형의 캐시들의 하이브리드; 및 원래의 미디어 파일을 획득하여 소비하고자 하는 최종 사용자(1205)를 포함하지만, 이들로 제한되지 않는다.

[0205] 시스템(1200)의 기존의 구성들에서, 기점 서버(1201)는 원래의 미디어 파일을 복제를 위해 모든 접속된 캐시들(1202, 1203, 1204)에 배포한다. 대조적으로, 개시된 비트스트림(301)을 이용하는 개시된 시스템에서, 기점 서버(1201)는 개시된 비트스트림 생성기(도 4 참조)를 이용하여 원래의 미디어 파일의 상이한 FE 코딩된 사본들을 생성한 다음, 각각을 상이한 접속된 캐시(1202, 1203, 1204)에 배포한다.

[0206] 시스템(1200)의 기존의 구성들에서, 기점 서버(1201)에 접속되지 않은 캐시들은 그 이웃들 중 하나에 접속하여 원래의 미디어 파일의 복제를 생성할 것이다. 대조적으로, 개시된 비트스트림(301)을 이용하는 개시된 시스템에서, 이러한 캐시는 다수의 이웃들에 접속하고, 도 9를 참조하여 설명된 개시된 비트스트림 리인코더를 이용하여 그 자신의 FE 코딩된 사본을 생성하기 위해 이웃들의 FE 사본들을 효율적으로 이용한다. 새롭게 생성된 FE 사본들은 또한 전체 또는 부분 사본들일 수 있다. 캐시들(1202, 1203, 1204)은 심지어 피어로서 기여하기를 원하는 최종 사용자(1205)일 수 있다.

[0207] 시스템(1200)은 모든 캐시가 동일한 원래의 미디어 파일의 상이하지만 FE 코딩된 사본들을 저장한 분산 미디어 전달 시스템이다. 이것은 효율적인 멀티소스 다중 경로 다운로드를 가능하게 하며, 최종 사용자(1205)는 상이한 통신 링크들을 통해 다수의 캐시들(1202, 1203, 1204)로부터 병렬로 다운로드할 수 있고, 다운로드된 데이터의 총량이 (헤더 오버헤드 등으로 인해) 원래의 미디어 파일보다 약간 더 많은 한, 최종 사용자(1205)는 도 4에 도시된 개시된 비트스트림 디코더(404)를 이용하여 모든 다운로드들을 종료하고 원래의 미디어 파일을 복구할 수 있다. 이러한 방식으로, 최종 사용자(1205)는 다운로드 비용을 증가시키지 않고 모든 통신 링크들의 집성된 대역폭 및 안정성을 즐길 수 있다.

[0208] 전술한 다운로드 프로세스는, 최종 사용자(1205)가 원래의 미디어 파일을 다운로드하기 위해 캐시들(1202, 1203, 1204) 중 하나에만 접속하는 기존의 단일 소스 단일 경로 다운로드 프로세스보다 우수하다. 기존의 다운로드 프로세스의 성능은 캐시와 최종 사용자(1205) 사이의 단일 링크에 의해 제한되고, 캐시 또는 링크 장애에 취약하다. 적어도 기존의 복제 기반의 멀티소스 다중 경로 다운로드 프로세스가 미디어 소스들로부터 복제된 데이터를 다운로드하는 것을 피하기 위해 다운로드들 간에 시간 소모적이고 에러가 발생하기 쉬운 스케줄링을 필요로 하기 때문에, 개시된 다운로드 프로세스는 또한 기존의 복제 기반의 멀티소스 다중 경로 다운로드 프로세스보다 우수하다는 점에 유의한다.

[0209] 도 13은 일부 실시예들에 따른 낮은 레이턴시 유니캐스트 비트스트림 전달 시스템(1300)을 도시한다. 시스템(1300)은 낮은 레이턴시 유니캐스트 이용 애플리케이션에서의 개시된 비트스트림(301)의 예시적인 이용을 보여주며, 여기서 수신기는 손실 통신 채널을 통한 데이터 스트림의 낮은 레이턴시 및 신뢰성 있는 전달을 요구한다. 이러한 이용 사례의 경우, 미디어 스트림은 전처리기 유닛(1301)에 의해, 비디오 프레임들과 같이 짧은 시간량 내에 수신기에 전송될 수 있는 작은 청크들로 전처리된다. 비트스트림 생성기(1302)는 개시된 비트스트림(301)을 생성하기 위해 전처리된 비트스트림에 적용된다. 그 후, SYNC 서브아톰(304), BITSTREAM_HEADER 서브아톰(305), MEDIA_SEGMENT_INFO 서브아톰(306)과 같이, 메타데이터를 운반하고 빈번하게 변경되지 않는 서브아톰들(1303)은 다른 채널들보다 더 높은 레이턴시 및 더 낮은 대역폭을 가질 수 있는 신뢰성 있는 제어 채널을 통해 수신기에 전달된다.

[0210] 한편, BLOCK_HEADER 서브아톰(308) 및 CHUNKED_PACKET_SUBATOM 서브아톰(311)과 같은 선형 코딩 관련 서브아톰들(1304)은 낮은 레이턴시 손실 데이터 채널을 통해 수신기에 전송되어 후처리를 위한 데이터 청크들의 낮은 레이턴시 디코딩을 가능하게 한다. 선택적으로, 비트스트림 디코더(1305)는 채널 상태들을 관찰하고 피드백을 피드백 채널을 통해 비트스트림 생성기(1302)에 전송하여 비트스트림 생성기(1302)가 코딩 결정 통지를 행하는 것을 도울 수 있다.

[0211] 도 14a는 일부 실시예들에 따른, PACKET 서브아톰(309)에 초점을 둔 낮은 레이턴시 유니캐스트 인코딩 및 디코딩을 위한 시스템(1400)의 블록도이다. 전처리된 미디어 스트림(1401)은 시간 틱 단위로 비트스트림 생성기(1402)에 공급되고, 각각의 틱은 선형 인코딩될 하나 이상의 동일 길이 데이터 심볼을 포함한다. 도시된 예에서, 틱 1, 틱 2 및 틱 4 각각은 1개의 데이터 심볼을 포함하는 반면, 틱 3은 2개의 데이터 심볼(3.2, 3.1)을 포함한다. 비트스트림 생성기(1402)는 체계적 PACKET 서브아톰들(309)로서 각각의 데이터 심볼에 대해 하나의 PACKET 서브아톰(309)을 생성하고, 또한 이러한 데이터 심볼들에 선형 코딩을 적용함으로써 선형 코딩된 패킷들을 생성한다. 수신기 측에 있는 비트스트림 디코더(1403)는 수신된 체계적 PACKET 서브아톰들(309) 및 코딩된 패킷들을 다른 지원 서브아톰들과 함께 이용하여 틱 데이터를 디코딩하고 그 데이터를 수신기의 애플리케이션 계층에 전달한다.

[0212] 도 14b는 일부 실시예들에 따른, 크기 마커(1404)를 통해 대역외 가변 데이터 크기를 처리하는 것을 도시한다. 전처리기(1401)는 틱 단위로 데이터를 생성하고, 여기서 각각의 틱의 데이터는 상이한 크기를 가질 수 있다. 일부 실시예들에서, 크기 마커(1404)는 각각의 틱 데이터의 크기를 측정하고, 크기 정보를 비트스트림 생성기(1405)에 전송하기 전에 틱 데이터에 크기 정보를 프리펜딩(prepend)한다. 이 크기 정보는 비트스트림 생성기(1405)의 관점에서 데이터에 포함되고 선형 인코딩된다. 도시된 예에서, 데이터의 3개의 틱, 즉 P1, P2, 및 P3이 있다. 그 크기는 데이터의 일부가 되는 크기 마커(1404)에 의해 프리펜딩된다.

[0213] 비트스트림 생성기(1405) 내에서, 제로 패더(1406)는 선형 인코딩을 위한 그 길이를 동등하게 하기 위해 데이터를 제로들로 패딩한다. 일부 실시예들에서, 길이는 함께 인코딩될 틱들 중에서 가장 큰 틱 데이터이다. 일부 실시예들에서, 전역적 최대 길이가 적용된다. 도시된 예에서, 3개의 틱이 함께 선형 인코딩되고, P1이 가장 크기 때문에, 심볼 길이는 크기 프리펜딩된(size prepended) P1의 길이로 설정된다. 따라서, P2 및 P3은 선형 인코딩 전에 제로들로 패딩된다.

[0214] 체계적 PACKET 서브아톰들(309)로서 틱 데이터를 전송할 때, 패딩된 제로들은 전송될 필요가 없다. 이것은 비트스트림 디코더(1408)가 체계적 PACKET 서브아톰(309)을 수신할 때, PACKET 서브아톰(309) 내의 심볼을 후술하는 패딩 필러(1409)로 직접 전달할 수 있기 때문이다. 패딩 필러(1409)는 크기를 파싱하고, 필링할 제로들이 존재하지 않는다는 것을 인식한다. 패딩 필러(1409)는 크기 프리픽스를 제거하고, 틱 데이터를 애플리케이션에 다시 전달한다.

[0215] 도시된 예에서, P3의 전송이 체계적 PACKET 서브아톰(309)으로서 성공적이라고 가정하고, 따라서 비트스트림 디코더(1408)는 제로 패딩 없이 크기 프리펜딩된(size-prepended) P3을 수신하고, 이를 패딩 필러(1407)에 전달한다. *b_systematic* 플래그를 PACKET 서브아톰(309)에서 *참*으로 설정하는 것은 결정을 하는데 이용된다. 비트스트림 디코더(1408)가 인코딩된 PACKET 서브아톰(309)을 인코딩된 심볼의 크기를 통해 수신할 때, 비트스트림 디코더는 디코딩을 가능하게 하기 위해 자신이 수신한 체계적 심볼들에 얼마나 많은 제로들을 패딩해야만 하는지를 안다.

[0216] 체계적 심볼이 디코딩을 통해 검색되는 경우, 비트스트림 디코더(1408)는 얼마나 많은 제로들이 패딩되었는지를 알지 못한다. 이 경우에, 비트스트림 디코더(1408)는 제로 패딩된 심볼을 패딩 필러(1409)에 전달하고, 패딩 필러(1409)는 크기 정보를 파싱하고 그에 따라 필링한다. 도시된 예에서, P2의 체계적 PACKET 서브아톰(309)의

전송이 실패한 것으로 가정하면, P2는 디코딩을 통해 검색된다. 디코딩된 P2는 제로가 패딩되어 있고, 이는 크기 정보를 파싱한 후에 패딩 필러(1409)에 의해 제거될 것이다.

[0217] 도 15a는 일부 실시예들에 따른, 낮은 레이턴시 유니캐스트의 별개의 선형 인코더-디코더 쌍들을 나타낸다. 특히, 도 15a는 개시된 비트스트림 구현을 이용하여 데이터의 다수의 틱(1501)을 보호하는 방법을 보여주며, 선형 인코딩 및 디코딩에 중점을 둔다. 이 방법에서, 각각의 틱에 대해, 하나의 선형 인코더(1502)가 생성된다. 이 선형 인코더(1502)는 이 틱 및 소수의 더 오래된 틱들로부터의 데이터를 블록으로서 취급하고, 이 블록에 선형 코딩을 적용한다. 수신기 측에서, 각각의 선형 인코더(1502)는 선형 디코더(1503)와 쌍을 이루며, 이 선형 디코더는 대응하는 블록의 데이터를 디코딩하고, 디코딩된 틱 데이터를 애플리케이션에 직접 전달하거나, 버퍼(1504)에 전달하고, 이 버퍼는 틱 데이터를 재정렬하고 복제들을 제거한다.

[0218] 위에 설명된 방법은 적응적 방식으로 구현될 수 있다. 예를 들어, 함께 인코딩될 틱들의 수는 네트워크 품질, 시스템 레이턴시 예산 및 다른 적합한 인자들과 같은 인자들에 기반하여 동적으로 정해질 수 있다. 다른 예로서, 인코더가 생성되어야 하는 경우에 틱들은 또한 틱당 하나의 인코더 대신에, 네트워크 품질 및 시스템 레이턴시 예산과 같은 인자들에 기반하여 동적으로 결정될 수 있다.

[0219] 도 15b는 일부 실시예들에 따른, 교차 생성 디코더를 이용한 낮은 레이턴시 유니캐스트를 나타낸다. 특히, 도 15b는 개시된 비트스트림(301)을 이용하여 데이터의 다수의 틱(1501)을 보호하는 방법을 보여주며, 선형 인코딩 및 디코딩에 중점을 둔다. 이 방법에서, 각각의 틱에 대해, 하나의 선형 인코더(1502)가 생성된다. 이 선형 인코더는 이 틱 및 소수의 더 오래된 틱들로부터의 데이터를 블록으로서 취급하고, 이 블록에 선형 코딩을 적용한다. 수신기 측에서, 예를 들어, 선형 방정식들의 확장된 세트를 풀어냄으로써, 하나 이상의 블록으로부터 PACKET 서브아톰들(309)을 수집하고 그들을 함께 디코딩하는 교차 블록 선형 디코더(1505)가 구현된다. 교차 블록 디코더(1505)는 또한 틱 데이터를 재정렬하고 복제들을 제거하며, 그 후 데이터를 수신기의 애플리케이션 계층에 전달한다.

[0220] 일부 실시예들에서, 위에 설명된 방법은 적응적 방식으로 구현될 수 있다. 예를 들어, 함께 인코딩될 틱들의 수는 네트워크 품질 및 시스템 레이턴시 예산과 같은 인자들에 기반하여 동적으로 정해질 수 있다. 다른 예로서, 교차 블록 디코더(1505)가 함께 디코딩할 블록들의 수는 또한 네트워크 품질 및 시스템 레이턴시 예산과 같은 인자들에 기반하여 동적으로 결정될 수 있다.

[0221] 도 16은 일부 실시예들에 따른, 디지털 저작권 관리(DRM)를 위해 개시된 비트스트림(301)을 이용하는 시스템(1600)을 도시한다. 특히, 도 16은 제안된 비트스트림이 어떻게 간단한 디지털 저작권 보호를 가능하게 할 수 있는지를 보여준다. 선형 인코딩된 미디어 데이터를 디코딩 또는 재인코딩하기 위해, 수신기(1602)는 코딩 계수들, 생성 크기, 패킷 크기, 랜덤 시드 등과 같은 선형 코딩 정보를 획득한다. 그렇지 않으면, 수신기(1602)는 미디어 데이터를 복구할 수 없다. 따라서, 제안된 비트스트림의 대응하는 서브아톰들에서 이러한 정보를 암호화함으로써, 콘텐츠 소유자(1601)는 미디어 데이터에 대한 비허가 액세스를 방지할 수 있다. 이것은 제안된 비트스트림을 이용하는 간단한 DRM 시스템을 가능하게 한다.

[0222] 일부 실시예들에서, 콘텐츠를 준비할 때, 콘텐츠 소유자(1601)는 특정 암호의 암호화 방식(예를 들어, AES 등)을 이용하여 선형 코딩 정보를 암호화한다. 암호화된 비트스트림은 그 후 통신 네트워크들을 통해 수신기(1602)에 분배된다. 콘텐츠 소유자(1601)는 그 후 해독 키를 인증 서버(1603)에 저장한다. 인증 서버(1603)는 그 ID에 기반하여 수신기(1602)를 인증하고, 인증이 통과되면 해독 키를 수신기(1602)에 전송한다. 수신기(1602)는 그 후 해독 키를 이용하여 선형 코딩 정보를 해독하고 그 정보를 이용하여 미디어 데이터를 디코딩 또는 재인코딩한다.

[0223] 도 17은 일부 실시예들에 따른, 데이터 인증을 갖는 분산 미디어 전달 시스템(1700)의 블록도이다. 특히, 도 17은 개시된 비트스트림이 피어들과 같은 비신뢰 캐싱 노드들이 존재할 수 있는 분산 미디어 전달 시스템에서 데이터 인증을 어떻게 가능하게 할 수 있는지를 보여준다.

[0224] 도시된 예시적인 실시예에서, 콘텐츠 생성기(1701)는 보안 실링 특징이 인에이블된 개시된 비트스트림을 이용하여 미디어 데이터를 다수의 비트스트림으로 인코딩한다. 예를 들어, 파일 무결성은 ENCODER_CONTENT_INFO 서브아톰(307)에 포함될 수 있고, 블록 무결성은 BLOCK_HEADER 서브아톰(308)에 포함되며, 패킷 무결성은 PACKET 서브아톰(309)에 포함된다. 그 후, 콘텐츠 생성기(1701)는 비트스트림들을 캐싱 노드들(1704-1706)의 네트워크(1703)에 분배한다. 이 예에서는 3개의 캐싱 노드가 도시되어 있지만, 다수의 캐싱 노드가 네트워크(1703)에 포함될 수 있다.

[0225] 캐싱 노드들(1704-1706)은 비트스트림들을 변경되지 않은 채로 저장하거나 개시된 비트스트림 포맷에 의해 가능하게 되는 재인코딩 기능들을 통해 새로운 비트스트림들을 생성할 수 있다. 이들 중에서, 비신뢰 캐싱 노드들(1705, 1706)은 적대적일 수 있고 손상된 비트스트림, 특히 정확하게 선형 코딩되지 않은 PACKET 서브아톰들(309)을 생성할 수 있다.

[0226] (최종 사용자 또는 새로운 캐싱 노드일 수 있는) 수신기 노드(1707)는 먼저 올바른 ENCODER_CONTENT_INFO 서브아톰(307) 및 BLOCK_HEADER 서브아톰(308)을 다운로드하여 파일 무결성 정보 및 블록 무결성 정보를 획득함으로써 자신이 비신뢰 캐싱 노드들(1705, 1706)로부터 다운로드하는 PACKET 서브아톰들(309)을 인증할 수 있다. 이러한 정보는, (예컨대, 호모모픽 해싱을 이용하여) 적절히 생성될 때, 비신뢰 노드들에 의해 생성된 PACKET 서브아톰들(309)의 무결성을 검증하는데 이용될 수 있다.

[0227] 올바른 ENCODER_CONTENT_INFO 서브아톰(307) 및 BLOCK_HEADER 서브아톰(308)을 다운로드하기 위해, 수신기 노드(1707)는 허가된 당사자들에 의해 운영되는 것들과 같은 신뢰 캐싱 노드들(1704)을 이용할 수 있다. 대안적으로, ENCODER_CONTENT_INFO 서브아톰(307) 및 BLOCK_HEADER 서브아톰(308)이 각각의 미디어 자산에 대해 정적이기 때문에, 콘텐츠 생성기(1701)는 이들 2개의 서브아톰들을 암호로 서명할 수 있고, 따라서 수신기(1707)는 어디에서나 이들 2개의 서브아톰들을 다운로드할 수 있고, 콘텐츠 생성기의 암호 서명을 통해 이들을 검증할 수 있다.

[0228] 그에 추가하여, 수신기 노드(1707)가 노드로부터의 수신된 PACKET 서브아톰(309)이 손상된 것을 발견할 때, 수신기 노드는, 예컨대, 포렌식 사기 증거를 제출함으로써 이 위반을 시스템(1700)의 규제 관리자(1702)에게 보고할 수 있다. 그러면, 규제 관리자(1702)는 벌금 부과, 가입 취소, 및 네트워크(1703)에서의 이 노드의 블랙리스트 추가와 같은 수단에 의해 잘못을 행한 노드에 페널티를 가할 수 있다.

[0229] *예시적인 프로세스들*

[0230] 도 18은 일부 실시예들에 따른, 비선형 또는 선형 코딩된 데이터로부터 비트스트림을 생성하는 프로세스(1800)의 흐름도이다. 프로세스(1800)는, 예를 들어, 도 22를 참조하여 개시된 전자 디바이스 아키텍처를 이용하여 구현될 수 있다.

[0231] 프로세스(1800)는 제1 디바이스에서, 미디어 자산을 나타내는 미디어 데이터를 수신하는 단계(1801); 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 획득하는 단계(1802); 제1 선형 코딩 프로세스에 따라 미디어 데이터의 적어도 일부를 미디어 자산에 대응하는 코딩된 데이터로 인코딩하는 단계(1803); 제1 선형 코딩 프로세스와 연관된 정보에 기반하여 제1 복수의 데이터 요소와 상이한 제2 복수의 데이터 요소를 생성하는 단계(1804); 제1 복수의 데이터 요소 및 제2 복수의 데이터 요소를 미디어 자산을 나타내는 하나 이상의 코딩된 비트스트림으로 결합하는 단계(1805); 및 하나 이상의 네트워크 경로를 이용하여 하나 이상의 디바이스에 하나 이상의 코딩된 비트스트림을 전송하는 단계(1806)를 포함한다.

[0232] 일부 실시예들에서, 프로세스(1800)는 하나 이상의 서버(예를 들어, 콘텐츠 전달 네트워크(CDN) 서버, PoP(Point-of-presence) 서버, 하나 이상의 애플리케이션 서버(예를 들어, 전자 뉴스 수집 애플리케이션, 콘텐츠 브로드캐스팅 애플리케이션, 유니캐스트 애플리케이션 등을 호스팅하는 서버)), 콘텐츠 캐시, 피어 디바이스, 로컬 캐싱 디바이스, 에지 디바이스(예를 들어, 5G/6G 에지 애플리케이션 서버) 또는 클라이언트 디바이스(예를 들어, 미디어 소비 디바이스, 컴퓨터, 모바일 디바이스, 게임 시스템, 스트리밍 디바이스, 오디오/비디오 장비 등)를 포함하지만 이에 제한되지 않는 하나 이상의 디바이스에 의해 구현된다.

[0233] 일부 실시예들에서, 미디어 자산은 이미지들, 비디오, 게임 상태, 게임 커맨드 데이터 등을 포함하지만 이에 제한되지 않는다.

[0234] 일부 실시예들에서, 미디어 데이터는 소스 코딩된 데이터, 스트리밍 프로토콜(예를 들어, HLS(HTTP-like streaming), DASH(dynamic adaptive streaming over HTTP))에 따라 패키징된 데이터 등을 포함하지만, 이에 제한되지 않는다. 일부 실시예들에서, 미디어 데이터는 선형 코딩된 데이터 또는 미디어 자산과 연관된 코딩된 비트스트림(예를 들어, 코딩된 콘텐츠 및 연관된 메타데이터를 포함하는 코딩된 비트스트림)이다. 일부 실시예들에서, 미디어 자산을 나타내는 미디어 데이터는 복수의 네트워크 경로들로부터 수신된다.

[0235] 일부 실시예들에서, 미디어 세그먼트 데이터는 각각의 세그먼트 또는 블록 내의 미디어 자산의 특성들(예를 들어, MEDIA_SEGMENT_INFO)을 식별한다. 일부 실시예들에서, 제1 복수의 데이터 요소들은 미디어 자산과 연관된

(예를 들어, 그로부터 도출된, 그에 대응하는, 그에 의해 수신된) 메타데이터(예를 들어, SYNC 및 MEDIA_SEGMENT_INFO 서브아톰들)를 포함한다. 일부 실시예들에서, 제1 복수의 데이터 요소들은 선형 코딩된 페이로드 데이터를 포함하지 않는다.

[0236] 일부 실시예들에서, 제1 선형 코딩 프로세스(예로서, 지원되는 선형 코드 유형에 대응하는 선형 코딩 프로세스)에 따른 인코딩은 (예로서, 미디어 데이터의 적어도 하나의 부분에 대응하는 복수의 패킷의 결합들인 코딩된 패킷들을 생성하는 것을 포함하여) 선형 인코딩 기술을 미디어 데이터의 적어도 제1 부분을 포함하는 패킷들에 적용하는 것을 포함한다.

[0237] 일부 실시예들에서, 제1 선형 코딩 프로세스에 따른 인코딩은 미디어 데이터의 적어도 한 부분에 선형 코딩 기술을 적용하지 않고 수행된다(예를 들어, 프로세스는 미디어 데이터의 적어도 한 부분에 대응하는 복수의 패킷을 결합함으로써 코딩된 패킷들을 생성하는 것을 포함하지 않는다). 예를 들어, 인코딩 프로세스는 제1 선형 코딩 프로세스의 적용을 요구하지 않고 미디어 데이터의 적어도 일부를 코딩된 데이터로 변환하기 위해 제1 선형 코딩 프로세스와 연관된 매핑 함수(예를 들어, 하나 이상의 매핑 규칙 또는 테이블)를 적용하는 것을 포함할 수 있다.

[0238] 일부 실시예들에서, 제2 복수의 데이터 요소는 미디어 자산과 연관된 코딩된 페이로드 데이터 및 메타데이터(예를 들어, BITSTREAM_HEADER 서브아톰(305), ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER 서브아톰(308), PACKET 서브아톰(309), 또는 PACKET_HEADER_ONLY 서브아톰(312) 중 하나 이상 및 그 연관된 비트필드들)를 포함한다.

[0239] 일부 실시예들에서, 결합하는 것은 전송 또는 저장을 위해 순차적 신호 또는 표현으로 인터리빙, 멀티플렉싱 또는 다른 방식으로 변환하는 것을 포함한다.

[0240] 일부 실시예들에서, 하나 이상의 네트워크 경로는 하나 초과의 네트워크, 하나 초과의 네트워크 유형(예컨대, LAN, VLAN, WAN, SAN, WLAN, VPN 등), 하나 초과의 네트워크 표준(예컨대, 1G/2G/3G/4G/5G/6G 모바일, 블루투스, 802.11 기반 WIFI 등), 하나 초과의 네트워크 전송 매체(예컨대, 무선, 유선, 섬유 등), 상이한 성능 특성을 갖는 통신 채널(예컨대, 손실 프로토콜, 신뢰성 있는 프로토콜, 보다 낮은 레이턴시를 이용하는 채널)을 포함한다.

[0241] 도 19는 일부 실시예들에 따른 디코딩/재인코딩의 프로세스(1900)의 흐름도이다. 프로세스(1900)는, 예를 들어, 도 22를 참조하여 개시된 전자 디바이스 아키텍처를 이용하여 구현될 수 있다.

[0242] 프로세스(1900)는 제1 전자 디바이스에서, 미디어 자산과 연관된 하나 이상의 코딩된 비트스트림을 수신하는 단계(1901); 버퍼에 하나 이상의 코딩된 비트스트림을 저장하는 단계(1902); 버퍼에 저장된 하나 이상의 코딩된 비트스트림을 처리하는 단계(1903) - 이 처리하는 단계는 하나 이상의 코딩된 비트스트림으로부터 비트스트림 식별, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 추출하는 단계를 포함함 - ; 하나 이상의 코딩된 비트스트림으로부터, 하나 이상의 코딩된 비트스트림을 생성하는데 이용된 각각의 선형 코딩 프로세스들과 연관되는 정보를 포함하는 제2 복수의 데이터 요소를 추출하는 단계(1904); 및 제2 복수의 데이터 요소에 기반하여 미디어 자산을 나타내는 디코딩된 데이터를 생성하는 단계, 또는 제2 복수의 데이터 요소에 기반하여 미디어 자산을 나타내는 새롭게 코딩된 데이터를 생성하는 단계 중 적어도 하나를 수행하는 단계(1905)를 포함한다.

[0243] 일부 실시예들에서, 제1 디바이스는 서버(예를 들어, 콘텐츠 전달 네트워크(CDN) 서버, PoP(Point-of-presence) 서버, 애플리케이션 서버(예를 들어, 전자 뉴스 수집 애플리케이션, 콘텐츠 브로드캐스팅 애플리케이션, 유니캐스트 애플리케이션 등을 호스팅하는 서버)), 콘텐츠 캐시, 피어 디바이스, 로컬 캐싱 디바이스, 에지 디바이스(예를 들어, 5G/6G 에지 애플리케이션 서버), 클라이언트 디바이스(예를 들어, 미디어 소비 디바이스, 컴퓨터, 모바일 디바이스, 게임 시스템, 스트리밍 디바이스, 오디오/비디오 장비 등)이다.

[0244] 일부 실시예들에서, 하나 이상의 코딩된 비트스트림은 적어도 2개의 상이한 선형 코딩 프로토콜들/표준들 중의 하나에 따라 인코딩된 비트스트림들을 포함한다.

[0245] 일부 실시예들에서, 처리는 디멀티플렉싱 및/또는 파싱을 포함한다.

[0246] 일부 실시예들에서, 미디어 세그먼트 데이터는 각각의 세그먼트 또는 블록 내의 미디어 자산의 특성들(예를 들어, MEDIA_SEGMENT_INFO 서브아톰(306))을 식별하는데 이용된다.

[0247] 일부 실시예들에서, 제1 복수의 데이터 요소는 미디어 자산과 연관된(예를 들어, 미디어 자산으로부터 도출된,

미디어 자산에 대응하는, 미디어 자산에 의해 수신된) 메타데이터(예를 들어, SYNC 서브아톰(304) 및 MEDIA_SEGMENT_INFO 서브아톰(306))를 포함한다.

[0248] 일부 실시예들에서, 제1 복수의 데이터 요소는 선형 코딩된 페이로드 데이터를 포함하지 않는다.

[0249] 일부 실시예들에서, 제2 복수의 데이터 요소는 미디어 자산과 연관된 코딩된 페이로드 데이터 및 메타데이터(예를 들어, BITSTREAM_HEADER 서브아톰(305), ENCODER_CONTENT_INFO 서브아톰(307), BLOCK_HEADER 서브아톰(308), PACKET(309), PACKET_HEADER_ONLY 서브아톰(312) 등 중 하나 이상)를 포함한다.

[0250] 일부 실시예들에서, 코딩된 데이터는 적어도 2개의 상이한 선형 코딩 프로토콜/표준 중 상이한 하나에 대응한다.

[0251] 도 20은 일부 실시예들에 따른, 다운스트림 호환성을 위해 새로운 코드 유형을 변환하는 프로세스(2000)의 흐름도이다. 프로세스(2000)는, 예를 들어, 도 22를 참조하여 개시된 전자 디바이스 아키텍처를 이용하여 구현될 수 있다.

[0252] 프로세스(2000)는 제1 전자 디바이스에서, 제1 유형의 제1 코딩된 비트스트림을 획득하는 단계(2001); 제1 유형과 상이한 제2 유형의 제2 코딩된 비트스트림을 생성하는 단계(2002); 및 하나 이상의 네트워크 경로를 통해 하나 이상의 제2 전자 디바이스에 제2 코딩된 비트스트림을 전송하는 단계(2003)를 포함한다.

[0253] 일부 실시예들에서, 제1 유형의 제1 비트스트림은 제1 유형의 선형 코딩 프로토콜 또는 표준(예를 들어, RaptorQ™)에 따라 생성된다.

[0254] 일부 실시예들에서, 제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 체계적 패킷들 및 데이터 세그먼트와 연관된 비체계적 패킷들을 포함한다.

[0255] 일부 실시예들에서, 제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 체계적 패킷들을 포함하고 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하지 않는다.

[0256] 일부 실시예들에서, 제1 또는 제2 비트스트림들은 데이터 세그먼트와 연관된 비체계적 패킷들을 포함하고 데이터 세그먼트와 연관된 체계적 패킷들을 포함하지 않는다.

[0257] 일부 실시예들에서, 제2 유형의 제2 비트스트림은 제2 유형의 선형 코딩 프로토콜 또는 표준(예를 들어, 리드 솔로몬 코드)에 따라 생성된다.

[0258] 일부 실시예들에서, 하나 이상의 네트워크 경로는 하나 초과의 네트워크, 하나 초과의 네트워크 유형(예컨대, LAN, VLAN, WAN, SAN, WLAN, VPN 등), 하나 초과의 네트워크 표준(예컨대, 1G/2G/3G/4G/5G/6G 모바일, 블루투스, 802.11 기반 WIFI 등), 하나 초과의 네트워크 전송 매체(예컨대, 무선, 유선, 섬유 등), 상이한 성능 특성을 갖는 통신 채널(예컨대, 손실 프로토콜, 신뢰성 있는 프로토콜, 보다 낮은 레이턴시 프로토콜을 이용하는 채널) 등을 포함한다.

[0259] 도 21은 일부 실시예들에 따른, 다운스트림 호환성을 위해 새로운 코드 유형을 변환하는 다른 프로세스의 흐름도이다. 프로세스(2100)는, 예를 들어, 도 22를 참조하여 개시된 전자 디바이스 아키텍처를 이용하여 구현될 수 있다.

[0260] 프로세스(2100)는 제1 전자 디바이스에서, 제1 유형의 선형 코딩으로 인코딩된 미디어 자산과 연관된 제1 비트스트림을 수신하는 단계(2101); 제1 비트스트림이 제2 유형의 선형 코딩과 연관된 디코더로 디코딩되어야 한다는 결정에 따라, 제1 비트스트림의 필드들의 세트로부터 데이터를 추출하고, 데이터를 제2 비트스트림의 필드들의 제2 및 대응하는 세트에 매핑하는 단계를 포함하는, 제1 비트스트림을 제2 유형의 선형 코딩의 제2 비트스트림으로 처리하는 단계(2102); 제1 비트스트림으로부터 미디어 자산과 연관된 메타데이터를 추출하는 단계(2103); 제2 비트스트림을 제2 유형의 선형 코딩과 연관된 선형 디코더로 미디어 자산의 코딩되지 않은 표현으로 디코딩하는 단계(2104); 제1 비트스트림이 제2 유형의 선형 코딩과 연관된 디코더로 디코딩되지 않아야 한다는 결정에 따라, 미디어 자산과 연관된 메타데이터 및 제1 유형의 선형 코딩과 연관된 선형 디코더로 제1 비트스트림을 미디어 자산의 코딩되지 않은 표현으로 디코딩하는 단계를 포함하는, 제1 비트스트림을 처리하는 단계(2105); 및 미디어 자산의 코딩되지 않은 표현을 미디어 자산과 연관된 추출된 메타데이터를 갖는 애플리케이션에 전달하는 단계(2106)를 포함한다.

[0261] 위에 개시된 프로세스들(1800, 1900, 2000 및 2100)은 각각 상이한 성능 특성들을 갖는 네트워크들을 통해 다양한 처리 능력들(예를 들어, 인코딩/디코딩 능력들)을 갖는 디바이스들로의/로부터의 미디어 데이터의 보다 효율

적인 전달을 가능하게 하기 위해, 선형 코딩 프로세스와 연관된 데이터 요소들을 전송 전에 비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 데이터 요소들과 결합하는 것을 가능하게 한다. 결합된 데이터는 보다 효율적인 디코딩을 제공하고 호환성을 보장하기 위해 다수의 중복 유형의 비트스트림들을 생성하고 전송할 필요성을 감소시킴으로써, 네트워킹 및 처리 리소스들을 절약한다.

[0262] *예시적인 시스템 아키텍처*

[0263] 도 22는 본 개시내용의 예시적인 실시예들을 구현하는데 적합한 예시적인 전자 디바이스 아키텍처(2200)의 블록도를 나타낸다. 아키텍처(2200)는 도 1 내지 도 21을 참조하여 전술한 바와 같은 서버들 및 클라이언트 디바이스들을 포함하지만 이에 제한되지 않는다.

[0264] 도시된 바와 같이, 아키텍처(2200)는, 예를 들어, ROM(read only memory)(2202)에 저장된 프로그램 또는, 예를 들어, 저장 유닛(2208)으로부터 RAM(random access memory)(2203)으로 로딩된 프로그램에 따라 다양한 프로세스들을 수행할 수 있는 CPU(central processing unit)(2201)를 포함한다. RAM(2203)에서, CPU(2201)가 다양한 프로세스들을 수행할 때 요구되는 데이터가 또한 필요에 따라 저장된다. CPU(2201), ROM(2202) 및 RAM(2203)은 버스(2204)를 통해 서로 접속된다. 입력/출력(I/O) 인터페이스(2205)는 또한 버스(2204)에 접속된다.

[0265] 다음의 컴포넌트들이 I/O 인터페이스(2205)에 접속된다: 키보드, 마우스 등을 포함할 수 있는 입력 유닛(2206); 액정 디스플레이(LCD) 및 하나 이상의 스피커와 같은 디스플레이를 포함할 수 있는 출력 유닛(2207); 하드 디스크 또는 다른 적절한 저장 디바이스를 포함하는 저장 유닛(2208); 및 네트워크 카드(예를 들어, 유선 또는 무선)와 같은 네트워크 인터페이스 카드를 포함하는 통신 유닛(2209).

[0266] 일부 구현들에서, 입력 유닛(2206)은 다양한 포맷들(예를 들어, 모노, 스테레오, 공간, 몰입형, 및 다른 적절한 포맷들)로 오디오 신호들의 캡처를 가능하게 하는 (호스트 디바이스에 따른) 상이한 위치들에 있는 하나 이상의 마이크로폰을 포함한다.

[0267] 일부 구현들에서, 출력 유닛(2207)은 다양한 수의 스피커들을 갖는 시스템들을 포함한다. 출력 유닛(2207)은 (호스트 디바이스의 능력들에 따라) 다양한 포맷들(예를 들어, 모노, 스테레오, 몰입형, 바이너럴(binaural), 및 다른 적절한 포맷들)로 오디오 신호들을 렌더링할 수 있다.

[0268] 일부 실시예들에서, 통신 유닛(2209)은 (예를 들어, 네트워크를 통해) 다른 디바이스들과 통신하도록 구성된다. 드라이브(2210)는 또한 필요에 따라 I/O 인터페이스(2205)에 접속된다. 자기 디스크, 광학 디스크, 광자기 디스크, 플래시 드라이브 또는 다른 적합한 이동식 매체와 같은 이동식 매체(2211)는 드라이브(2210) 상에 장착되어, 그로부터 판독된 컴퓨터 프로그램이 필요에 따라 저장 유닛(2208) 내에 설치되게 한다. 본 기술분야의 통상의 기술자는 시스템(2200)이 위에 설명된 컴포넌트들을 포함하는 것으로 설명되지만, 실제 애플리케이션들에서, 이러한 컴포넌트들 중 일부를 추가, 제거, 및/또는 대체하는 것이 가능하고, 모든 이러한 수정들 또는 변경은 모두 본 개시내용의 범위 내에 속한다는 것을 이해할 것이다.

[0269] 본 개시내용의 예시적인 실시예에 따르면, 전술한 프로세스는 컴퓨터 소프트웨어 프로그램으로서 또는 컴퓨터 판독가능한 저장 매체 상에서 구현될 수 있다. 예를 들어, 본 개시내용의 실시예는 머신 판독가능한 매체 상에 유형으로 구현된 컴퓨터 프로그램을 포함하는 컴퓨터 프로그램 제품을 포함하고, 컴퓨터 프로그램은 방법을 수행하기 위한 프로그램 코드를 포함한다. 이러한 실시예에서, 컴퓨터 프로그램은, 도 22에 도시된 바와 같이, 통신 유닛(2209)을 통해 네트워크로부터 다운로드 및 장착되고/되거나 이동식 매체(2211)로부터 설치될 수 있다.

[0270] 일반적으로, 본 개시내용의 다양한 예시적인 실시예들은 하드웨어 또는 특수 목적 회로들(예컨대, 제어 회로), 소프트웨어, 로직, 또는 그 임의의 조합으로 구현될 수 있다. 예를 들어, 위에서 논의된 유닛들은 제어 회로(예컨대, 도 22의 다른 컴포넌트들과 조합한 CPU(2201))에 의해 실행될 수 있고, 이에 따라, 제어 회로는 본 개시내용에서 설명된 액션들을 수행할 수 있다. 일부 양태들은 하드웨어로 구현될 수 있는 반면, 다른 양태들은 제어기, 마이크로프로세서, 또는 다른 컴퓨팅 디바이스(예컨대, 제어 회로)에 의해 실행될 수 있는 펌웨어 또는 소프트웨어로 구현될 수 있다. 본 개시내용의 예시적인 실시예들의 다양한 양태들이 블록도들, 흐름도들로서, 또는 일부 다른 그림 표현을 이용하여 예시되고 설명되지만, 본 명세서에서 설명된 블록들, 장치, 시스템들, 기술들, 또는 방법들은 비제한적인 예들로서, 하드웨어, 소프트웨어, 펌웨어, 특수 목적 회로들 또는 로직, 범용 하드웨어 또는 제어기 또는 다른 컴퓨팅 디바이스들, 또는 그 일부 조합으로 구현될 수 있다는 것이 인식될 것이다.

[0271] 추가로, 흐름도들에 도시된 다양한 블록들은, 방법 단계들로서, 그리고/또는 컴퓨터 프로그램 코드의 동작으로부터 생기는 동작들로서, 그리고/또는 연관된 기능(들)을 수행하도록 구성된 복수의 결합된 로직 회로 요소들로서 간주될 수 있다. 예를 들어, 본 개시내용의 실시예들은 머신 판독가능한 매체 상에 유형으로 구현된 컴퓨터 프로그램을 포함하는 컴퓨터 프로그램 제품을 포함하고, 컴퓨터 프로그램은 전술한 방법들을 수행하도록 구성된 프로그램 코드들을 포함한다.

[0272] 본 개시내용의 맥락에서, 머신 판독가능한 매체는 명령어 실행 시스템, 장치, 또는 디바이스에 의해 또는 그와 관련하여 이용하기 위한 프로그램을 포함하거나 저장할 수 있는 임의의 유형의 매체일 수 있다. 머신 판독가능한 매체는 머신 판독가능한 신호 매체 또는 머신 판독가능한 저장 매체일 수 있다. 머신 판독가능한 매체는 비일시적일 수 있고, 전자, 자기, 광학, 전자기, 적외선, 또는 반도체 시스템, 장치, 또는 디바이스, 또는 전술한 것의 임의의 적합한 조합을 포함할 수 있지만, 이들로 제한되지 않는다. 머신 판독가능한 저장 매체의 보다 구체적인 예들은 하나 이상의 와이어를 갖는 전기 접속, 휴대용 컴퓨터 디스켓, 하드 디스크, 랜덤 액세스 메모리(RAM), 판독 전용 메모리(ROM), 소거가능 프로그래머블 판독 전용 메모리(EPROM 또는 플래시 메모리), 광섬유, 휴대용 컴팩트 디스크 판독 전용 메모리(CD-ROM), 광학 저장 디바이스, 자기 저장 디바이스, 또는 전술한 것의 임의의 적합한 조합을 포함할 것이다.

[0273] 본 개시내용의 방법들을 수행하기 위한 컴퓨터 프로그램 코드는 하나 이상의 프로그래밍 언어의 임의의 조합으로 작성될 수 있다. 이러한 컴퓨터 프로그램 코드들은 범용 컴퓨터, 특수 목적 컴퓨터, 또는 제어 회로를 갖는 다른 프로그래머블 데이터 처리 장치의 프로세서에 제공될 수 있어서, 프로그램 코드들은, 컴퓨터 또는 다른 프로그래머블 데이터 처리 장치의 프로세서에 의해 실행될 때, 흐름도들 및/또는 블록도들에서 지정된 기능들/동작들이 구현되게 한다. 프로그램 코드는 컴퓨터 상에서 전체적으로, 컴퓨터 상에서 부분적으로, 독립형 소프트웨어 패키지로서, 컴퓨터 상에서 부분적으로 그리고 원격 컴퓨터 상에서 부분적으로, 또는 원격 컴퓨터 또는 서버 상에서 전체적으로, 또는 하나 이상의 원격 컴퓨터 및/또는 서버에 걸쳐 분배되어 실행될 수 있다.

[0274] *다른 실시예들*

[0275] 일부 실시예들에서, 제1 선형 코딩 프로세스(예로서, 지원되는 선형 코드 유형에 대응하는 선형 코딩 프로세스)에 따른 인코딩은 (예로서, 미디어 데이터의 적어도 하나의 부분에 대응하는 복수의 패킷의 결합들인 코딩된 패킷들을 생성하는 것을 포함하여) 선형 인코딩 기술을 미디어 데이터의 적어도 제1 부분을 포함하는 패킷들에 적용하는 것을 포함한다.

[0276] 일부 실시예들에서, 제1 선형 코딩 프로세스에 따른 인코딩은 미디어 데이터의 적어도 한 부분에 선형 코딩 기술을 적용하지 않고 수행된다(예를 들어, 프로세스는 미디어 데이터의 적어도 한 부분에 대응하는 복수의 패킷을 결합함으로써 코딩된 패킷들을 생성하는 것을 포함하지 않는다). 예를 들어, 인코딩 프로세스는 제1 선형 코딩 프로세스의 적용을 요구하지 않고 미디어 데이터의 적어도 일부를 코딩된 데이터로 변환하기 위해 제1 선형 코딩 프로세스와 연관된 매핑 함수(예를 들어, 하나 이상의 매핑 규칙 또는 테이블)를 적용하는 것을 포함할 수 있다.

[0277] *개시된 실시예들의 이점들*

[0278] 개시된 비트스트림은 다음의 이점들을 제공한다.

[0279] 1. 개시된 실시예들은 블록 코드들을 포함하지만 이에 제한되지 않는 임의의 선형 코드를 포함하는, 동일한 비트스트림 내의 다수의 선형 코딩 기술들의 이용을 지원한다.

[0280] 2. 개시된 실시예들은 미디어 자산이 멀티소스 다중 경로 미디어 데이터 전달 시스템들을 통해 인코딩되고, 분배되고, 재인코딩되고, 캐싱되고, 그 후 전달 포인트에서 디코딩될 수 있게 하는 메타데이터를 포함한다.

[0281] 3. 개시된 비트스트림(301)(서브아톰들 및 메타데이터)은 중앙집중형 제어의 필요 없이 그리고 소스들 사이의 조정 없이 멀티소스 다중 경로 전달을 허용하고, 전달 포인트에서 최소 중복을 가능하게 한다.

[0282] 4. 개시된 실시예들은 서브아톰 시스템이 상이한 방법들을 이용하여 상이한 유형들의 데이터를 전송할 수 있게 하며, 이는 상이한 서브아톰들의 유연하고 이종인 전달을 가능하게 한다. 예를 들어, 메타데이터와 같은 일부 서브아톰들은 신뢰가능한 채널로 전달될 수 있고, 미디어 에센스와 같은 다른 것들은 신뢰가능하지 않은 채널들로 전달될 수 있다. 일부 서브아톰들은 암호화될 수 있고, 일부는 암호화되지 않을 수 있다.

[0283] 5. 개시된 실시예들은 CHUNKED_SUBATOM 서브아톰들의 이용을 통해 통신 프로토콜 데이터 단위 크기에 맞는 유연

한 비트스트림 서브아톰 크기를 허용한다.

[0284] 6. 개시된 실시예들은 미디어 자산을 더 작은 블록들로 청킹하고, 이후 상이한 블록들을 나타내기 위해 BLOCK_HEADER 서브아톰의 block_index 필드를 이용함으로써 상이한 블록들에 개별적으로 선형 코드를 적용하는 것을 허용한다.

[0285] 7. 개시된 실시예들은 서브아톰 구조를 변경하지 않고 특정 서브아톰들 내의 인코딩된 데이터의 선형 재인코딩을 허용한다(예를 들어, 출력은 동일한 서브아톰 구조이고, 어떠한 변경도 없이 디코더에 의해 소비될 수 있다).

[0286] 8. 개시된 실시예들은 서브아톰 구조를 변경하지 않고 상이한 디바이스들의 저장 제약을 충족시키기 위해 다양한 크기의 기능적으로 동등한 코딩된 사본들의 생성을 허용한다.

[0287] 9. 개시된 실시예들은 복수의 소스들(예를 들어, CDN들)을 통해 전달된 미디어 자산을 복수의 상이한 선형 코드들로 인코딩하는 능력을 지원하고, 이러한 복수의 비트스트림들이 디코딩 위치에서 이용가능할 때, 이들은 동일한 비트스트림 디코더를 이용하여 디코딩될 수 있다.

[0288] 10. 개시된 실시예들은 복수의 상이한 선형 코드들로 인코딩되었고 복수의 소스들(예를 들면, CDN들)을 통해 전달된 미디어 자산을 선형 재인코딩하는 능력을 지원하고, 이 복수의 비트스트림들이 재인코딩 위치에서 이용가능할 때, 이들은 동일한 비트스트림 코덱을 이용하여 재인코딩된다.

[0289] 11. 개시된 실시예들은 기존의 표준화된 선형 코드 포맷(예를 들어, RaptorQ, R-S)을 이용하여 비트스트림으로 인코딩된 미디어 자산의 변환을 허용하고, 그 후 이 표준화된 비트스트림은 매핑 함수를 통해 제안된 비트스트림으로 매핑될 수 있다. 제안된 비트스트림의 디코더에서 디매핑 함수를 적용하여 원래의 표준화된 비트스트림을 표준화된 포맷으로 다시 복구한다.

[0290] 12. 개시된 실시예들은 슬라이딩 윈도우 또는 블록 코드인 선형 코드의 선택 및 이용을 지원한다. 포맷은, 슬라이딩 윈도우 코드가 인코더에서 이용되는 경우, 디코더가 블록 또는 슬라이딩 윈도우 디코딩들 중 어느 하나를 이용할 수 있게 한다.

[0291] 13. 개시된 실시예들은 미디어 자산이 신뢰가능하지 않은 채널을 통해 전송되는 동안 신뢰가능한 채널들을 통해 전송될 수 있는 서브아톰들 및 적절한 선형 코드의 선택(예를 들어, 슬라이딩 윈도우 코드들에 대한 지원)을 통해 낮은 레이턴시 손실 미디어 데이터 유니캐스트 시스템들의 전달을 지원한다.

[0292] 14. 개시된 실시예들은 허가되지 않은 디코더가 미디어 자산(예를 들어, DRM)을 복구하는 것을 방지하기 위해 비트스트림 내의 선택된 서브아톰 및 필드들의 암호화를 가능하게 한다.

[0293] 15. 개시된 실시예들은 인증 정보를 운반하는 필드들에 의한 상이한 레벨들의 데이터 무결성 체크들(예를 들어, 파일 레벨, 블록 레벨, 및 패킷 레벨 체크들)을 지원한다. 특히, 이것은 디코딩 없이 선형 코딩된 패킷들의 무결성 체크를 가능하게 하기 위해 호모모픽 해싱과 같은 특수 해싱 알고리즘들의 이용을 지원한다.

[0294] 또한, 개시된 비트스트림 생성기는 다음의 이점들을 제공한다.

[0295] 1. 개시된 비트스트림 생성기는 통신 프로토콜 데이터 단위 크기에 맞도록 서브아톰들을 더 작은 부분들(CHUNKED_SUBATOM 서브아톰들(311))로 청킹할 수 있다.

[0296] 2. 개시된 비트스트림 생성기는 미디어 자산을 더 작은 블록들로 청킹하고 그 다음에 선형 코드를 상이한 블록들에 개별적으로 적용하고 BLOCK_HEADER 서브아톰의 block_index 필드를 이용하여 상이한 블록들을 나타낼 수 있다.

[0297] 3. 개시된 비트스트림 생성기는 서브아톰 멀티플렉서를 이용하여 다수의 상이한 비트스트림들을 생성할 수 있고, 상이한 비트스트림들은 상이한 네트워크 엔티티들에 전달될 수 있다.

[0298] 4. 개시된 비트스트림 생성기는 채널 정보를 수신하고 이를 이용하여 인코딩 방식을 적응시킬 수 있다.

[0299] 5. 개시된 비트스트림 생성기는 필드 linear_code_type를 이용하여 서브아톰 BITSTREAM_HEADER로 이것을 시그널링함으로써 미디어를 비트스트림으로 인코딩하기 위한 복수의 선형 코딩 기술들로부터 선형 코딩 기술을 선택한다.

[0300] 6. 개시된 비트스트림 생성기는 상이한 변형들에서, (동일한 SYNC, BITSTREAM_HEADER, ENCODER_CONTENT_INFO, 및 MEDIA_SEG_INFO 서브아톰들과 같은) 동일한 미디어 정보를 이 미디어 자산을 고유하게 식별하도록 두는

한편, (BLOCK_HEADER 서브아톰 내의 상이한 prng_seed, PACKET 서브아톰 내의 상이한 poly_index 및 상이한 계수들과 같은) 상이한 선형 코드 정보를 변형들을 구별하도록 둠으로써 동일한 미디어 자산의 복수의 선형 코딩된 변형들을 생성할 수 있다. 이러한 방식으로, 변형들은 멀티소스 데이터 전달 시스템에 걸쳐 분배될 수 있고 모두 미디어 자산의 디코딩에 기여할 수 있다.

[0301] 7. 개시된 비트스트림 생성기는 블록 코드 또는 슬라이딩 윈도우 코드가 다양한 필드들을 통해 이용되는지, 예컨대 BITSTREAM_HEADER 서브아톰의 linear_code_type 필드를 상이한 값들로 설정하는 것, 슬라이딩 윈도우 코드에 대해 BITSTREAM_HEADER 서브아톰의 content_source_size 필드를 제로로 설정하는 것, 및/또는 슬라이딩 윈도우 코드에 대해 PACKET 서브아톰의 동적 window_start_index 및 window_end_index를 이용하는 것을 시그널링할 수 있다.

[0302] 8. 개시된 비트스트림 생성기는 수신기 피드백에 기반하여 선형 코딩 기술 및 파라미터들을 동적으로 변경할 수 있다.

[0303] 9. 개시된 비트스트림 생성기는 미디어 자산의 허가되지 않은 디코딩을 방지하기 위해 코딩 정보(예컨대 PACKET 서브아톰의 coefficient_vector 필드)를 암호화하고 코딩 정보를 별도로(예컨대 PACKET_HEADER_ONLY 서브아톰을 이용하여) 운반할 수 있다.

[0304] 10. 개시된 비트스트림 생성기는 데이터 무결성 체크를 가능하게 하기 위해 다양한 해싱 기술들(SHA2, SHA256, 호모모픽 해싱 등)을 이용하여 다양한 방식들로 미디어 자산을 해싱(파일의 해싱, 블록의 해싱, 원래의 심볼들의 해싱 등)할 수 있다. 특히, 이것은 디코딩 없이 선형 코딩된 패킷들의 무결성 체크를 가능하게 하기 위해 호모모픽 해싱과 같은 특수 해싱 알고리즘들을 이용할 수 있다.

[0305] 또한, 개시된 비트스트림 리인코더 실시예들은 다음의 이점들을 제공한다.

[0306] 1. 개시된 비트스트림 리인코더는 상이한 네트워크 엔티티들에서 구현되어 비트스트림 리인코더를 수행하고 서로 조정하지 않고 기능적으로 동등한 전체 코딩된 사본들 및 부분 코딩된 사본들을 생성할 수 있다.

[0307] 2. 개시된 비트스트림 리인코더는 동일한 미디어 정보(동일한 SYNC, BITSTREAM_HEADER, ENCODER_CONTENT_INFO, 및 MEDIA_SEG_INFO 서브아톰들 등) 및 상이한 변형들로부터의 상이한 선형 코드 정보(BLOCK_HEADER 서브아톰 내의 상이한 prng_seed, PACKET 서브아톰 내의 상이한 poly_index 및 상이한 계수들 등)를 통해 동일한 미디어 자산의 상이한 선형 코딩된 변형들을 이용하여 식별 및 재인코딩할 수 있다.

[0308] 3. 개시된 비트스트림 리인코더는 상이한 선형 코드들을 이용하여 생성된 동일한 미디어 자산의 상이한 비트스트림들을 통합 및 재인코딩할 수 있다.

[0309] 4. 개시된 비트스트림 인코더는 미디어 자산의 허가되지 않은 디코딩을 방지하기 위해 (PACKET 서브아톰의 coefficient_vector 필드와 같은) 코딩 정보를 (재인코딩을 위해) 해독하고 이를 재암호화하고 (PACKET_HEADER_ONLY 서브아톰을 이용하는 것과 같이) 개별적으로 코딩 정보를 운반할 수 있다.

[0310] 5. 개시된 비트스트림 리인코더는 해싱 기술들(SHA2, SHA256, 호모모픽 해싱 등) 및 서브아톰들에 지정된 데이터 무결성 필드들에 따라 다양한 방식들(파일 검증, 블록 검증, 패킷 검증 등)로 미디어 자산의 무결성을 검증할 수 있다. 특히, 호모모픽 해싱과 같은 특수 해싱 기술들이 이용될 때, 이것은 디코딩 없이 선형 코딩된 패킷들을 검증할 수 있다.

[0311] 추가적으로, 개시된 비트스트림 디코더 실시예들은 다음의 특징들/이점들을 제공한다.

[0312] 1. 개시된 비트스트림 디코더는 동일한 미디어 자산의 복수의 상이한 비트스트림들을 식별 및 다운로드하고 이들을 이용하여 미디어 자산을 디코딩할 수 있다.

[0313] 2. 개시된 비트스트림 디코더는 동일한 미디어 자산의 상이한 블록들을 별도로 디코딩하고, 이들을 정확한 순서로 연접시켜, 표시자로서 BLOCK_HEADER 서브아톰의 block_index 필드를 이용함으로써 원래의 미디어 자산을 복구할 수 있다.

[0314] 3. 개시된 비트스트림 디코더는 커넥터로서 CHUNKED_SUBATOM 서브아톰들 내의 연관된 필드들(예컨대 original_subatom_id, chunk_size, chunked_segment_index, num_chunked_segments)을 이용하고, 아직 복구될 수 없는 CHUNKED_SUBATOM 서브아톰들을 일시적으로 저장하기 위해 버퍼를 이용하고, 복구를 시도하기 위해 병합기를 이용함으로써, 다수의 CHUNKED_SUBATOM 서브아톰들을 이용하여 완전한 서브아톰을 복구할 수 있다.

[0315] 4. 개시된 비트스트림 디코더는 상이한 선형 코드들을 이용하여 생성된 비트스트림들을 이용함으로써 미디어 자산을 통합하고 디코딩할 수 있다.

[0316] 5. 개시된 비트스트림 디코더는 인코딩 방식 및 레이턴시 요건에 기반하여 블록 디코딩 및 슬라이딩 윈도우 디코딩과 같은 다양한 유형의 디코딩을 지원한다.

[0317] 6. 개시된 비트스트림 디코더는 블록 코드 또는 슬라이딩 윈도우 코드가 BITSTREAM_HEADER 서브아톰의 linear_code_type 필드, BITSTREAM_HEADER 서브아톰의 content_source_size 필드, 및/또는 PACKET 서브아톰의 window_start_index 및 window_end_index와 같은 다양한 필드들을 통해 이용되는지를 식별할 수 있다.

[0318] 7. 개시된 비트스트림 디코더는 BITSTREAM_HEADER 서브아톰 및 필드 linear_code_type를 판독함으로써 복수의 선형 코딩 기술들로부터 비트스트림을 적절히 디코딩하기 위한 선형 코드 유형을 식별할 수 있다.

[0319] 8. 개시된 비트스트림 디코더는 동일한 미디어 정보(예를 들어, 동일한 SYNC, BITSTREAM_HEADER, ENCODER_CONTENT_INFO, 및 MEDIA_SEG_INFO 서브아톰들) 및 상이한 변형들 사이의 상이한 선형 코드 정보(예를 들어, BLOCK_HEADER 서브아톰에서의 상이한 prng_seed, PACKET 서브아톰에서의 상이한 poly_index 및 상이한 계수들)를 통해 동일한 미디어 자산의 상이한 선형 코딩된 변형들을 이용하여 식별 및 디코딩할 수 있다.

[0320] 9. 개시된 비트스트림 디코더는 윈도우 크기 및 생성 크기의 조합을 이용하여 선형 코드가 슬라이딩 윈도우인지 또는 블록 코드인지를 식별한다.

[0321] 10. 개시된 비트스트림 디코더는 수신된 서브아톰들에 기반하여 채널 정보를 분석할 수 있고, 정보를 인코딩 결정 통지를 위해 생성기에 보고할 수 있다.

[0322] 11. 개시된 비트스트림 디코더는 미디어 자산의 디코딩을 가능하게 하기 위해 코딩 정보(예를 들어, PACKET 서브아톰의 coefficient_vector 필드)를 해독할 수 있다.

[0323] 12. 개시된 비트스트림 디코더는 해싱 기술들(SHA2, SHA256, 호모모픽 해싱 등) 및 서브아톰들에 지정된 데이터 무결성 필드들에 따라 다양한 방식들(파일 검증, 블록 검증, 패킷 검증 등)로 미디어 자산의 무결성을 검증할 수 있다. 특히, 호모모픽 해싱과 같은 특수 해싱 기술들이 이용될 때, 이것은 디코딩 없이 선형 코딩된 패킷들을 검증할 수 있다.

[0324] 또한, 개시된 실시예들은 다음을 제공한다:

[0325] 1. 선형 코드 전의 데이터의 예비 처리를 이용하여 가변 크기 데이터 패킷들을 처리하기 위한 메커니즘이다. 따라서, 비트스트림은 미디어가 가변 크기 패킷들 내에 있음을 알지 못한다.

[0326] 2. 시간에 민감하지 않고, 신뢰가능하지만 레이턴시에 민감하지 않은 접속(TCP)을 통해 디코딩에 중요한 특정 서브아톰들을 전송하고, 신뢰가능하지 않은(그리고 낮은 레이턴시의) 접속들을 통해 인코딩된 미디어 스트림을 전송함으로써 미디어 스트림의 낮은 레이턴시 전달을 가능하게 할 수 있는 시스템이다.

[0327] 3. 기점 서버가 상이한 캐시들에 대해 동일한 미디어 자산의 상이한 기능적으로 동등한 선형 코딩된 사본들을 생성하고, 새로운 캐시들이 기존의 사본들을 재인코딩하여 새로운 기능적으로 동등한 선형 코딩된 사본들을 획득할 수 있고, 수신기가 복수의 소스들/캐시들로부터 데이터를 다운로드하여 원래의 미디어 자산을 복구하는 다중 경로 멀티소스 미디어 전달 시스템이다.

[0328] 4. 콘텐츠 생성기가 선형 코딩 정보를 암호화하고, 해독 키들을 허가된 당사자들에게만 제공하는 DRM 시스템이다. 이러한 방식으로, 허가된 당사자들만이 콘텐츠를 재코딩 및 디코딩할 수 있다.

[0329] 5. 데이터 무결성 인증을 허용하는 분산 미디어 전달 시스템이며, 네트워크 엔티티들은 수신된 데이터의 무결성을 검증하고 페널티를 주기 위해 적대적 노드들에 대한 사기 증거들을 규제 관리자에게 제출하기 위해 개시된 비트스트림의 데이터 무결성 필드들을 이용할 수 있다.

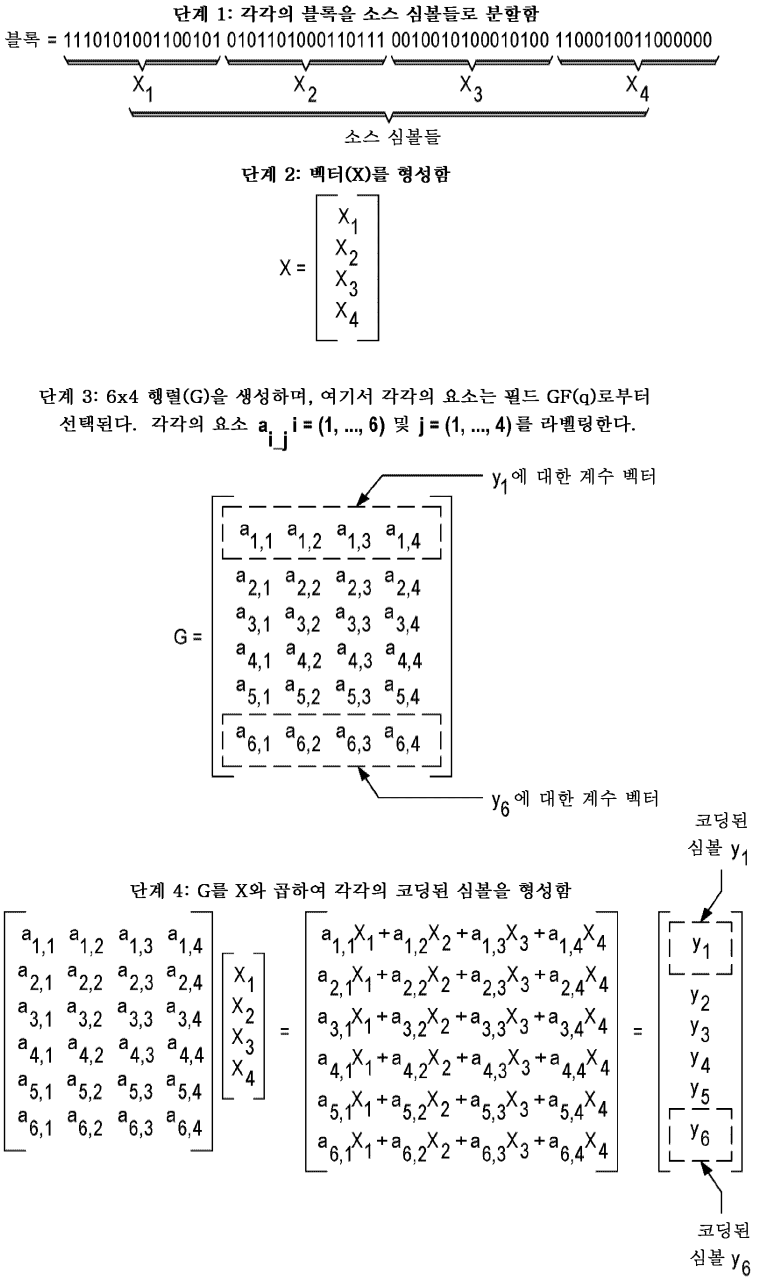
[0330] 6. 무선 업링크 및 다운링크 채널 양쪽 모두에서 미디어 및/또는 데이터의 코딩된 멀티소스 전달을 가능하게 한다. 업링크 채널에 대한 예시적인 실시예는 전자 뉴스 수집 애플리케이션을 포함한다. 그리고

[0331] 7. 동일한 5G/6G 에지 애플리케이션 서버가 레이턴시 민감 애플리케이션들에 대한 코딩된 유니캐스트 또는 브로드캐스트 트래픽으로의 실시간 변환을 지원하도록 허용하면서도, 5G/6G 에지 애플리케이션 서버 상에서의 효율적인/확장가능한 처리를 위해 개시된 발명에 따라 생성되고 포맷팅된 코딩된 멀티소스 미디어를 지원하는 엔드-투-엔드 솔루션을 가능하게 한다.

[0332] 본 문서는 많은 특정 구현 상세들을 포함하지만, 청구될 수 있는 것의 범위에 대한 제한들로서 해석되어서는 안 되고, 오히려 특정 실시예들에 특정될 수 있는 특징들의 설명들로서 해석되어야 한다. 별개의 실시예들의 맥락에서 본 명세서에서 설명되는 특정 특징들은 또한 단일 실시예에서 조합하여 구현될 수 있다. 반대로, 단일 실시예의 맥락에서 설명되는 다양한 특징들은 또한 다수의 실시예들에서 개별적으로 또는 임의의 적합한 하위 조합으로 구현될 수 있다. 더욱이, 특징들이 특정 조합들로 작용하는 것으로서 위에서 설명되고 심지어 그와 같이 초기에 청구될 수 있더라도, 청구되는 조합으로부터의 하나 이상의 특징은, 일부 경우들에서, 그 조합으로부터 삭제될 수 있고, 청구되는 조합은 하위 조합 또는 하위 조합의 변형에 관한 것일 수 있다. 도면들에 도시되는 로직 흐름들은 바람직한 결과들을 달성하기 위해, 도시된 특정 순서, 또는 순차적 순서를 요구하지 않는다. 또한, 다른 단계들이 제공될 수 있거나, 또는 단계들이 설명되는 흐름들로부터 제거될 수 있고, 다른 컴포넌트들이 설명되는 시스템들에 추가될 수 있거나, 또는 설명되는 시스템들로부터 제거될 수 있다. 따라서, 다른 구현들은 이하의 청구항들의 범위 내에 있다.

도면

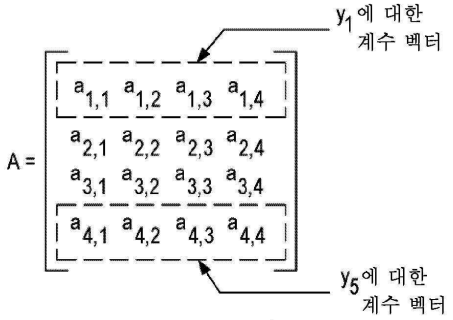
도면1a



도면1b

단계 1: 각각의 수신된 코딩된 심볼(Y_1)과 연관된 계수 벡터를 결정한다.

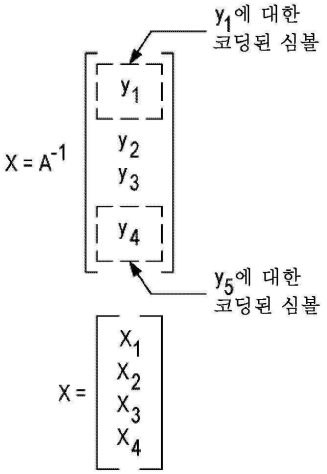
이 벡터는 행렬(A)에서 행 i가 된다.



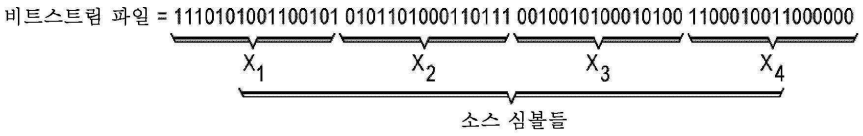
단계 2: A의 역수를 취하여 A^{-1}을 형성함

A^{-1} = INVERSE(A)

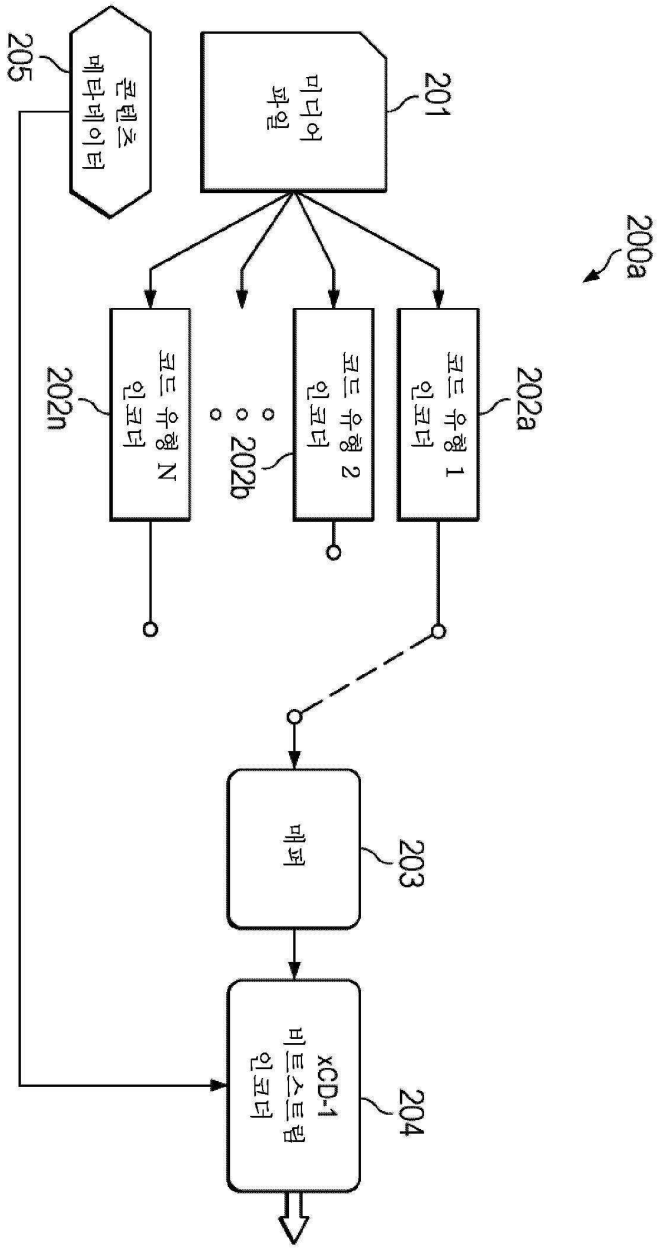
단계 3: A^{-1}을 코딩된 심볼들(y_1)의 벡터와 곱하여 x_1'a의 벡터를 복구함



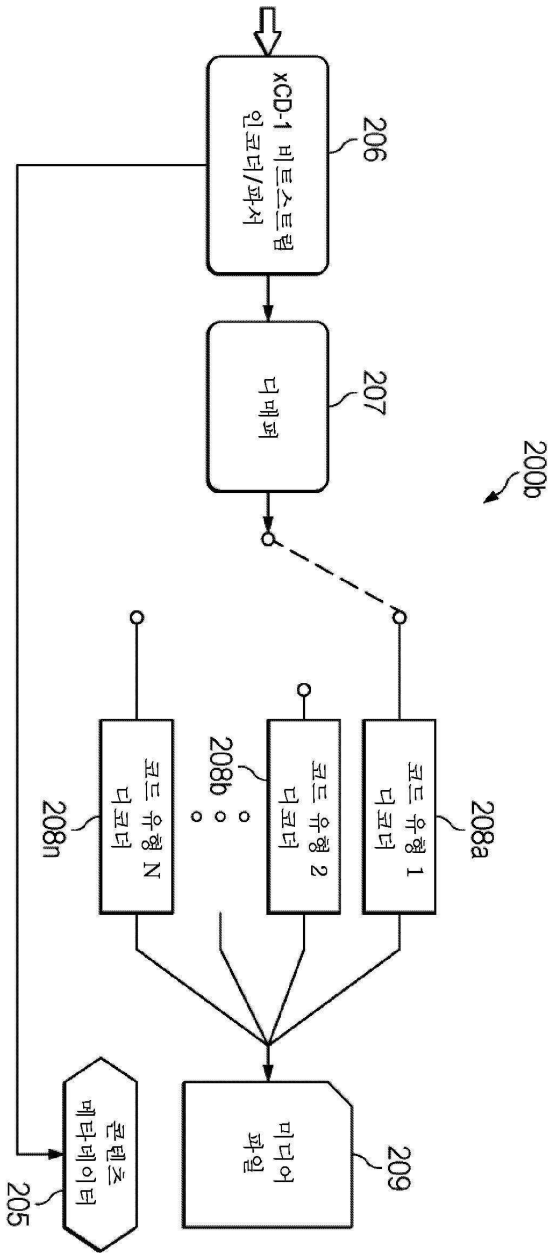
단계 4: 각각의 심볼(x_1)을 함께 연접시켜 블록을 재형성함



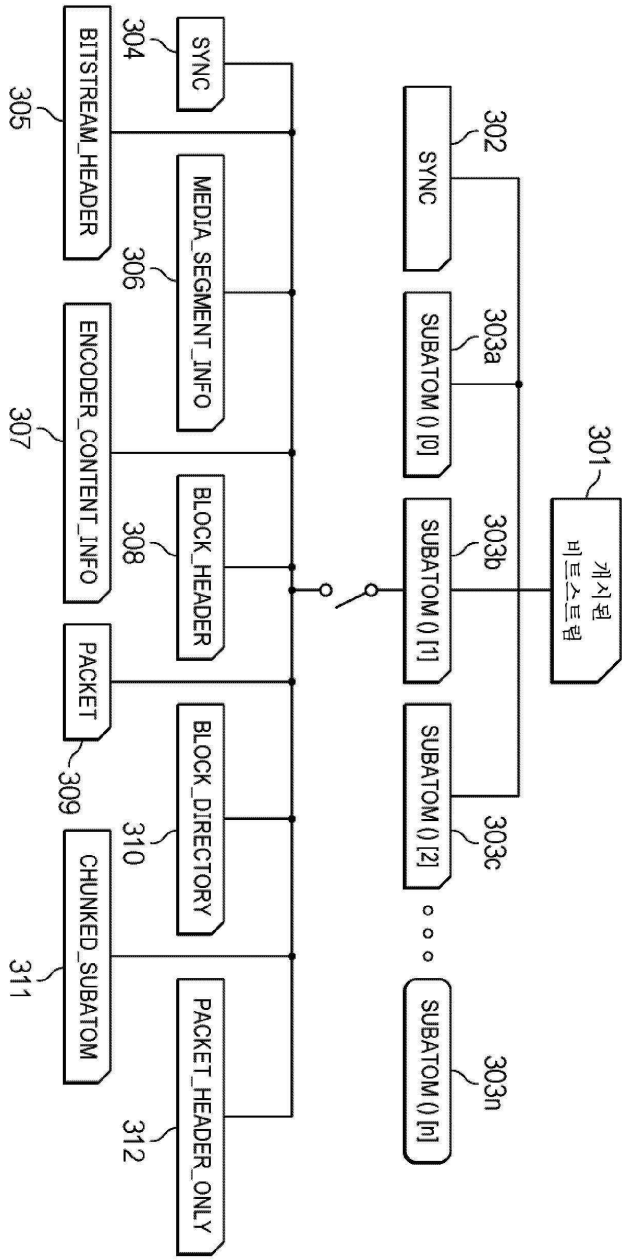
도면2a


200a
205
콘텐츠 메타데이터
201
미디어 파일
202n
코드 유형 N 인코더
202b
코드 유형 2 인코더
202a
코드 유형 1 인코더
203
매퍼
204
xCD-1 비트스트림 인코더

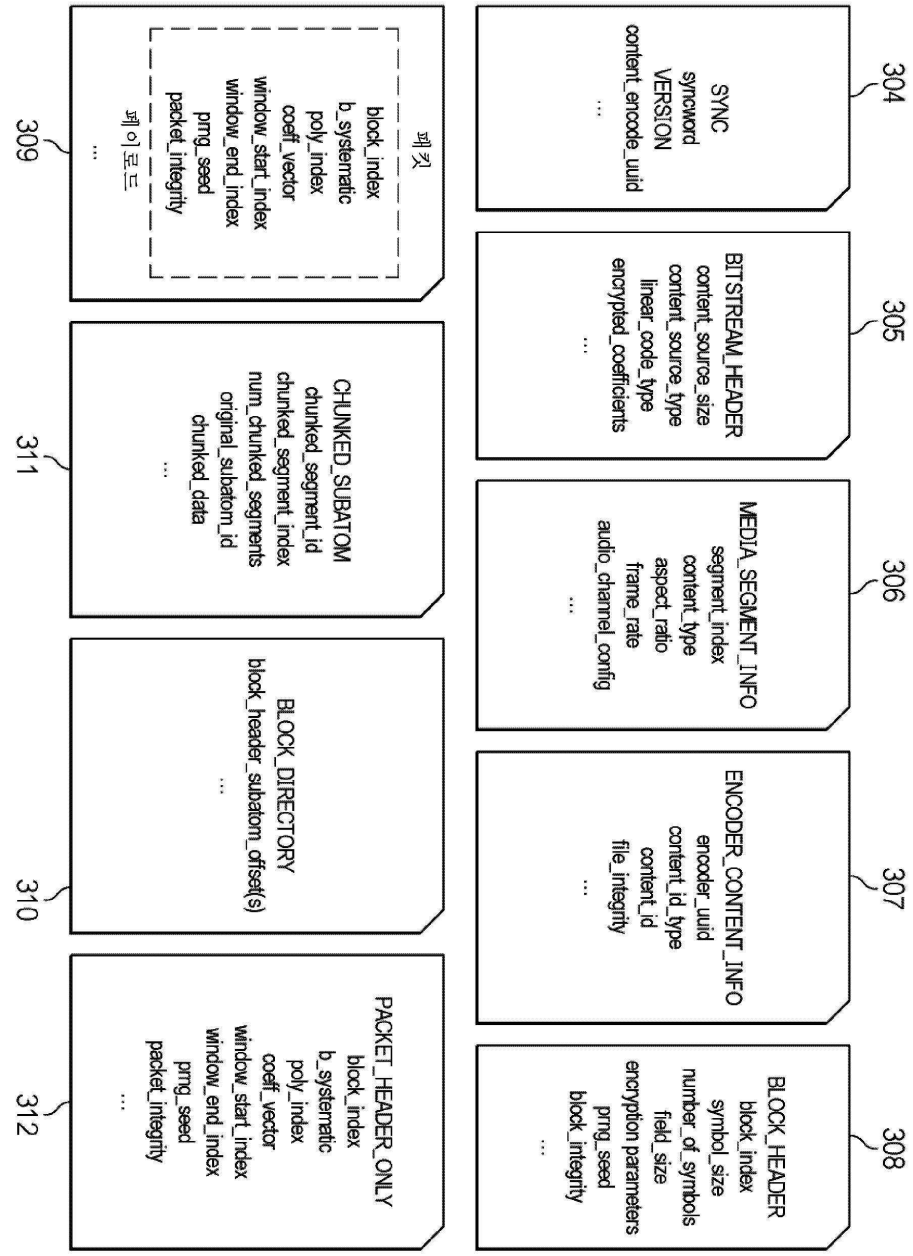
도면2b


xCD-1 비트스트림
인코더/파서
206
디매퍼
207
200b
코드 유형 N
디코더
208n
208b
코드 유형 2
디코더
코드 유형 1
디코더
208a
콘텐츠
메타데이터
205
미디어
파일
209

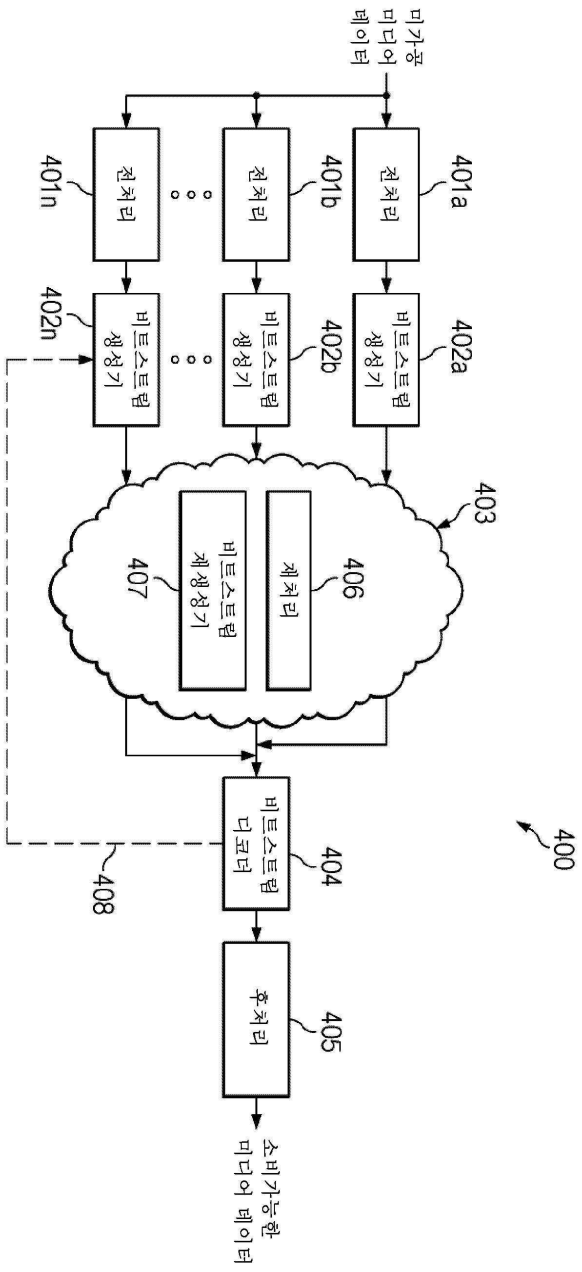
도면3a


301
캐시된
비트스트림
302
SYNC
303a
SUBATOM () [0]
303b
SUBATOM () [1]
303c
SUBATOM () [2]
303n
SUBATOM () [n]
304
SYNC
305
BITSTREAM_HEADER
306
MEDIA_SEGMENT_INFO
307
ENCODER_CONTENT_INFO
308
BLOCK_HEADER
309
PACKET
310
BLOCK_DIRECTORY
311
CHUNKED_SUBATOM
312
PACKET_HEADER_ONLY

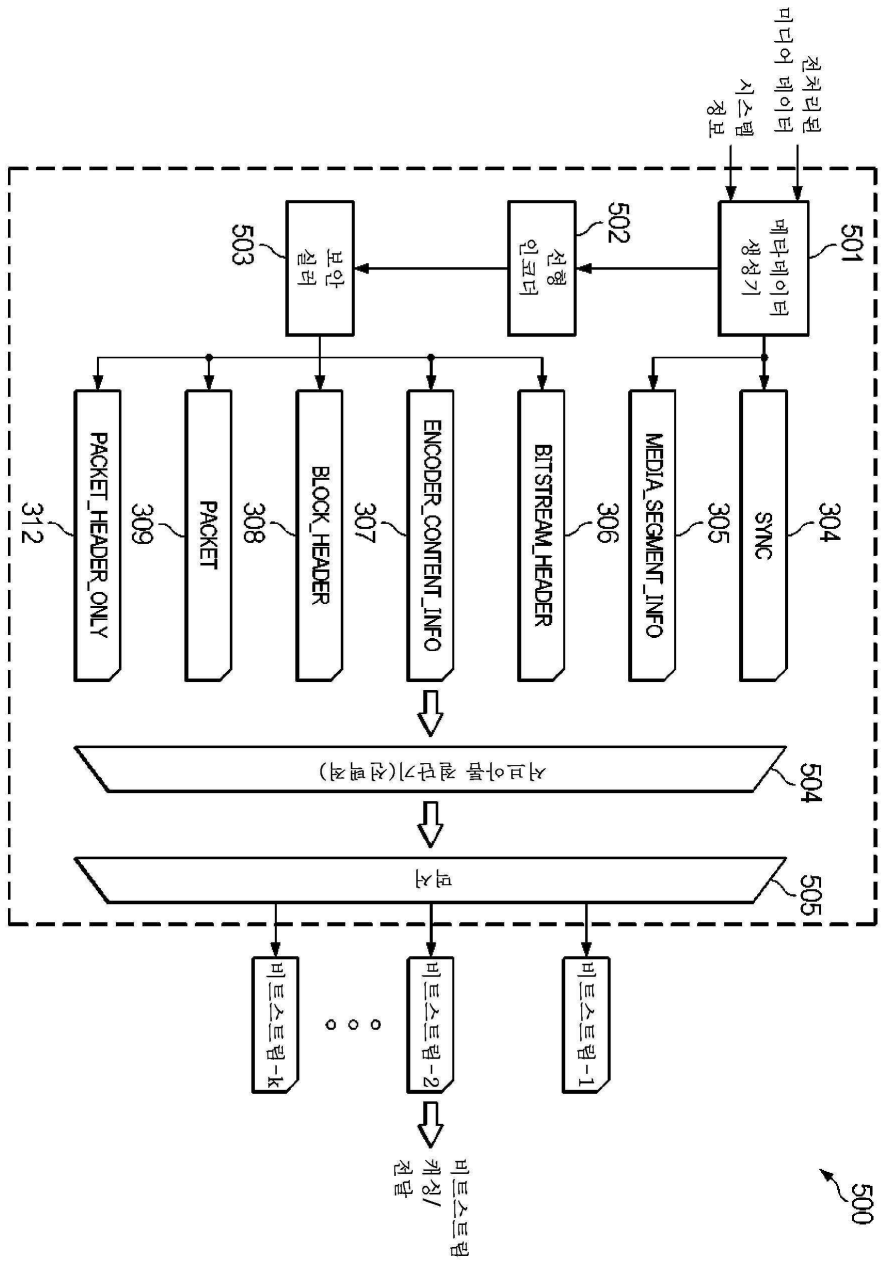
도면3b


304
SYNC
syncword
VERSION
content_encode_uuid
...
305
BITSTREAM_HEADER
content_source_size
content_source_type
linear_code_type
encrypted_coefficients
...
306
MEDIA_SEGMENT_INFO
segment_index
content_type
aspect_ratio
frame_rate
audio_channel_config
...
307
ENCODER_CONTENT_INFO
encoder_uuid
content_id_type
content_id
file_integrity
...
308
BLOCK_HEADER
block_index
symbol_size
number_of_symbols
field_size
encryption parameters
prng_seed
block_integrity
...
309
패킷
block_index
b_systematic
poly_index
coeff_vector
window_start_index
window_end_index
prng_seed
packet_integrity
페이로드
...
311
CHUNKED_SUBATOM
chunked_segment_id
chunked_segment_index
num_chunked_segments
original_subatom_id
chunked_data
...
310
BLOCK_DIRECTORY
block_header_subatom_offset(s)
...
312
PACKET_HEADER_ONLY
block_index
b_systematic
poly_index
coeff_vector
window_start_index
window_end_index
prng_seed
packet_integrity
...

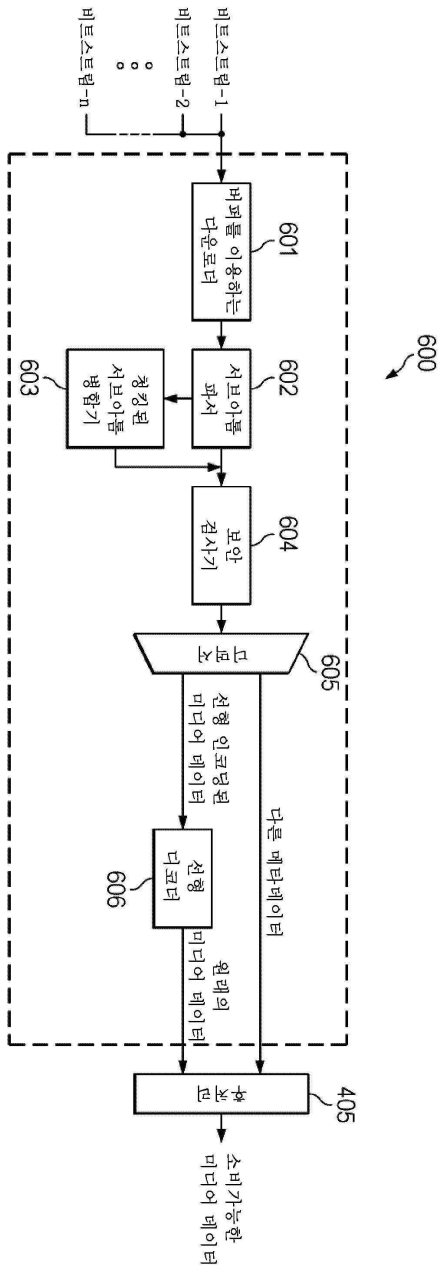
도면4


미가공 미디어 데이터
전처리
401a
401b
401n
비트스트림 생성기
402a
402b
402n
403
비트스트림 재생성기
407
재처리
406
비트스트림 디코더
404
408
400
후처리
405
소비가능한 미디어 데이터

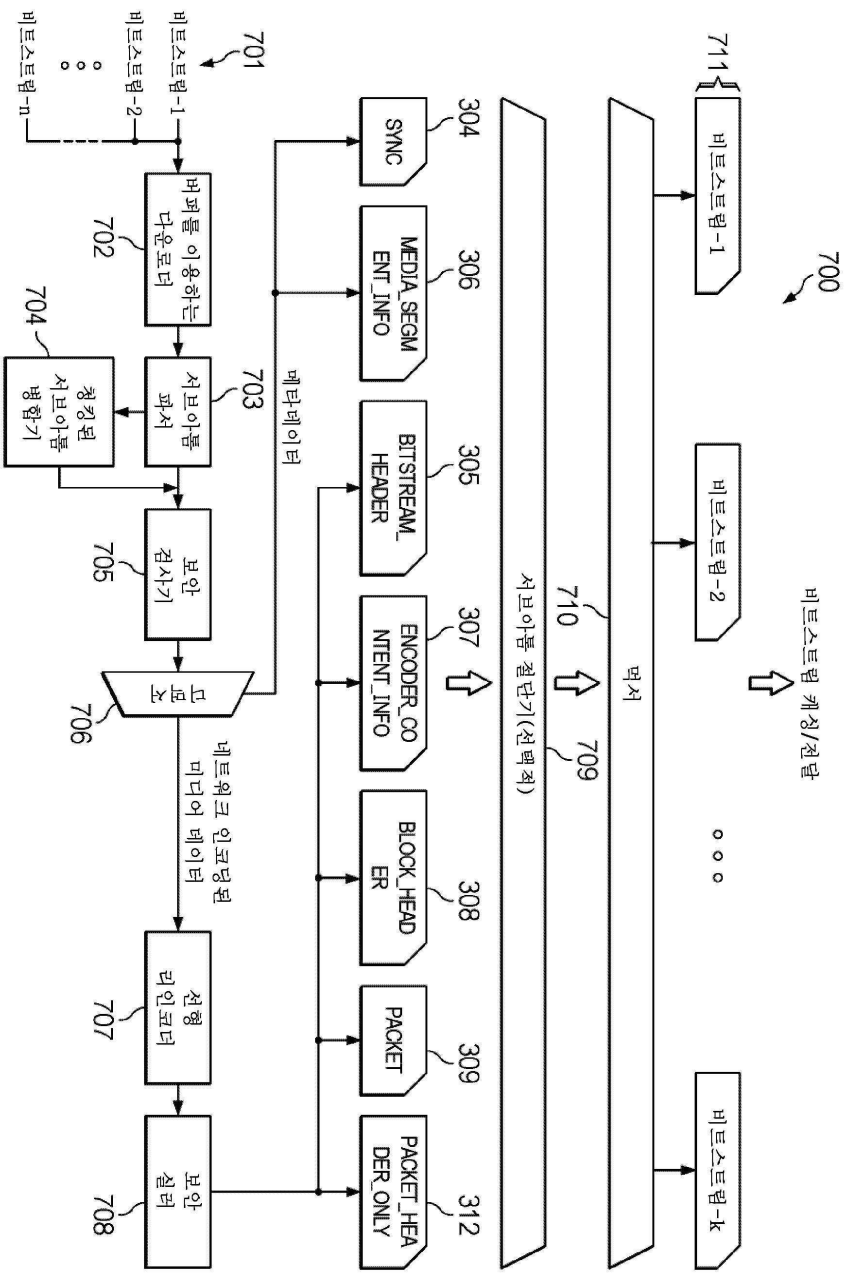
도면5


전처리된 미디어 데이터
시스템 정보
메타데이터 생성기
501
선형 인코더
502
보안 실러
503
SYNC
304
MEDIA_SEGMENT_INFO
305
BITSTREAM_HEADER
306
ENCODER_CONTENT_INFO
307
BLOCK_HEADER
308
PACKET
309
PACKET_HEADER_ONLY
312
서브바봄 절단기(선택적)
504
패킹
505
비트스트림-k
비트스트림-2
비트스트림-1
비트스트림 캐싱/전달
500

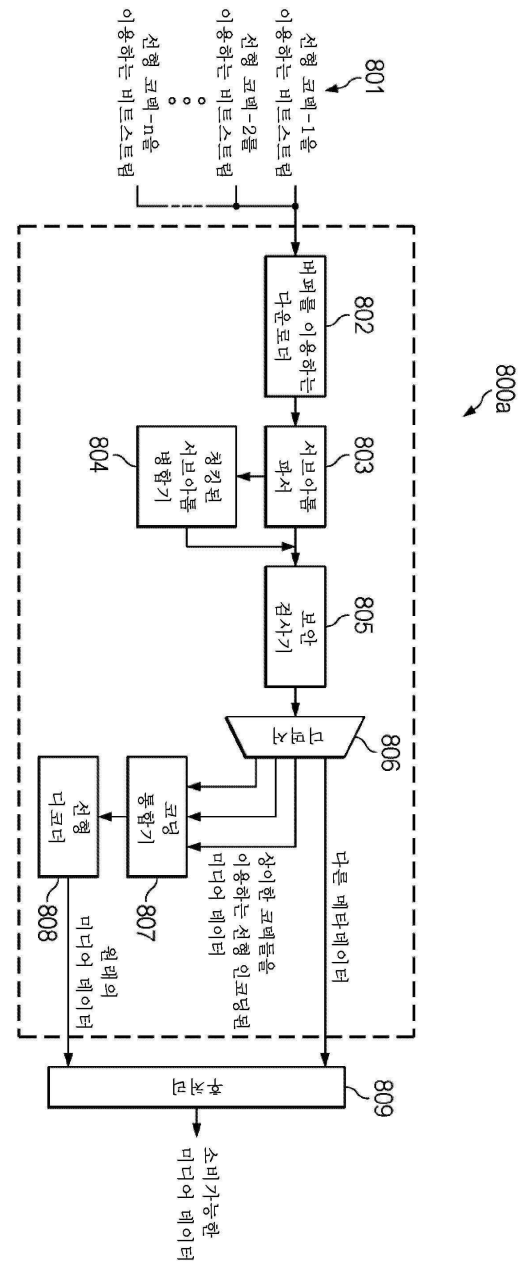
도면6


비트스트림-1
비트스트림-2
비트스트림-n
버퍼를 이용하는 다운로더
601
서브아틀 파서
602
첫 컨텍 서브아틀 병합기
603
600
보안 검사기
604
디코딩
605
선형 인코딩된 미디어 데이터
다른 메타데이터
선형 디코더
606
원래의 미디어 데이터
합성
405
소비가능한 미디어 데이터

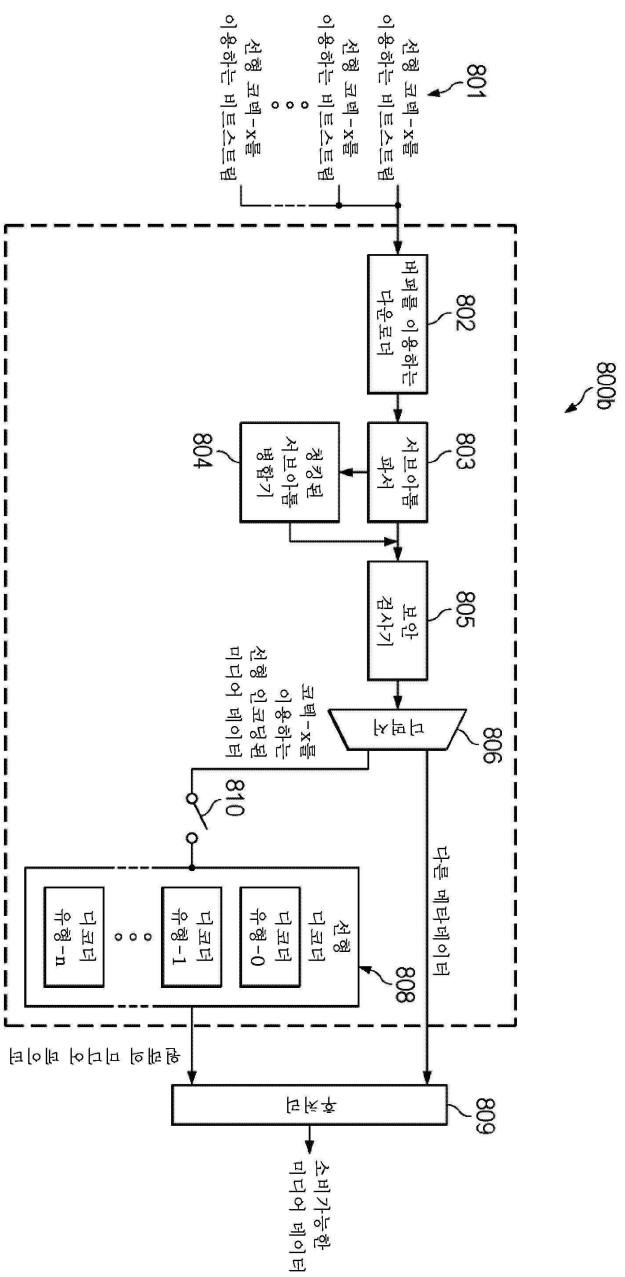
도면7


700
711
비트스트림-1
비트스트림-2
비트스트림-k
비트스트림 캐싱/전달
먹서
710
709
서브아톰 절단기(선택적)
304
SYNC
306
MEDIA_SEGMENT_INFO
305
BITSTREAM_HEADER
307
ENCODER_CONTENT_INFO
308
BLOCK_HEADER
309
PACKET
312
PACKET_HEADER_ONLY
701
비트스트림-1
비트스트림-2
비트스트림-n
702
버퍼를 이용하는 다운로더
703
서브아톰 파서
704
청킹된 서브아톰 병합기
705
보안 검사기
706
디멀티플렉서
707
선형 리인코더
708
보안 실러
메타데이터
네트워크 인코딩된 미디어 데이터

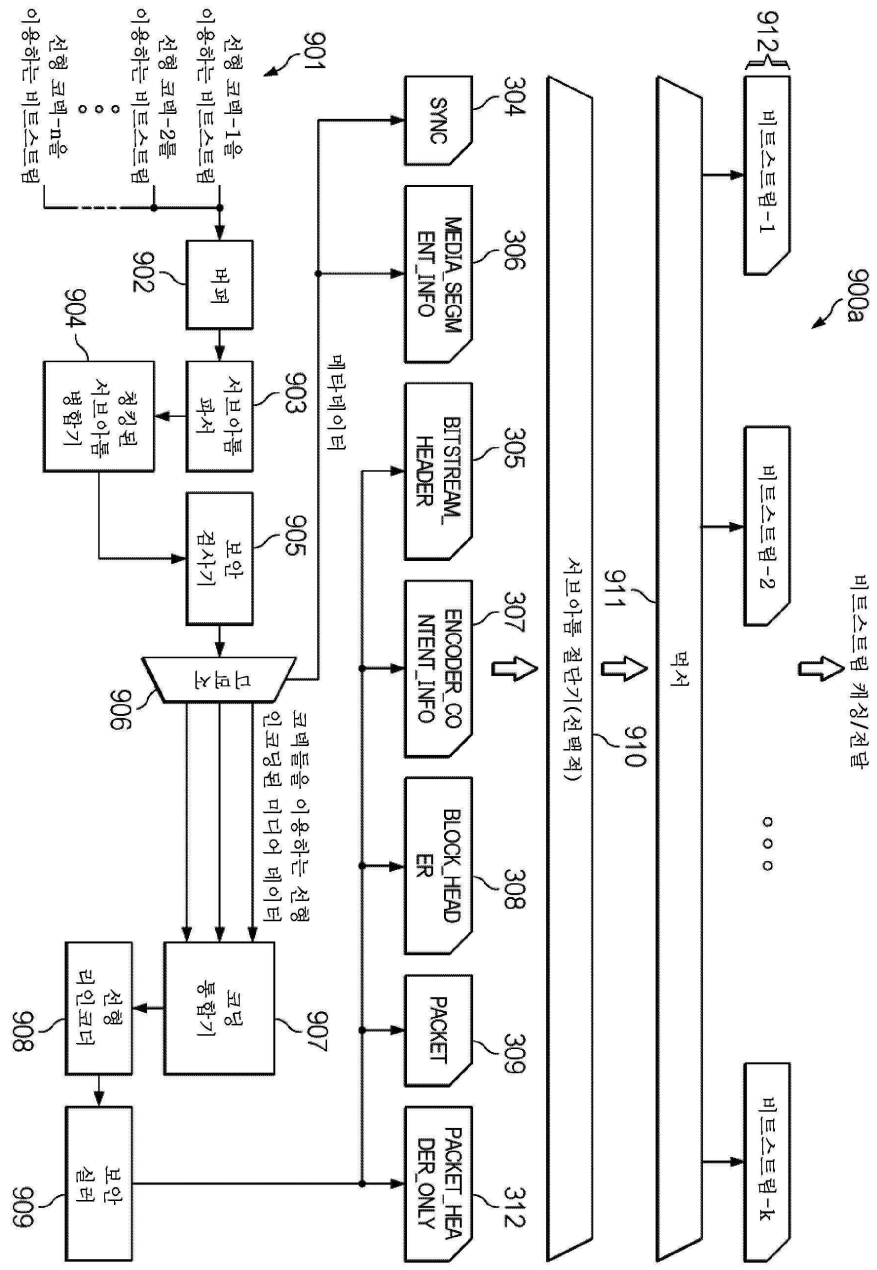
도면8a


선형 코덱-1을 이용하는 비트스트림
선형 코덱-2를 이용하는 비트스트림
선형 코덱-n을 이용하는 비트스트림
801
800a
버퍼를 이용하는 다운로더
802
서브아틈 파서
803
청킹된 서브아틈 병합기
804
보안 검사기
805
디멀티플렉서
806
코덱 불확기
807
선형 디코더
808
다른 메타데이터
상이한 코덱들을 이용하는 선형 인코딩된 미디어 데이터
원래의 미디어 데이터
멀티플렉서
809
소비가능한 미디어 데이터

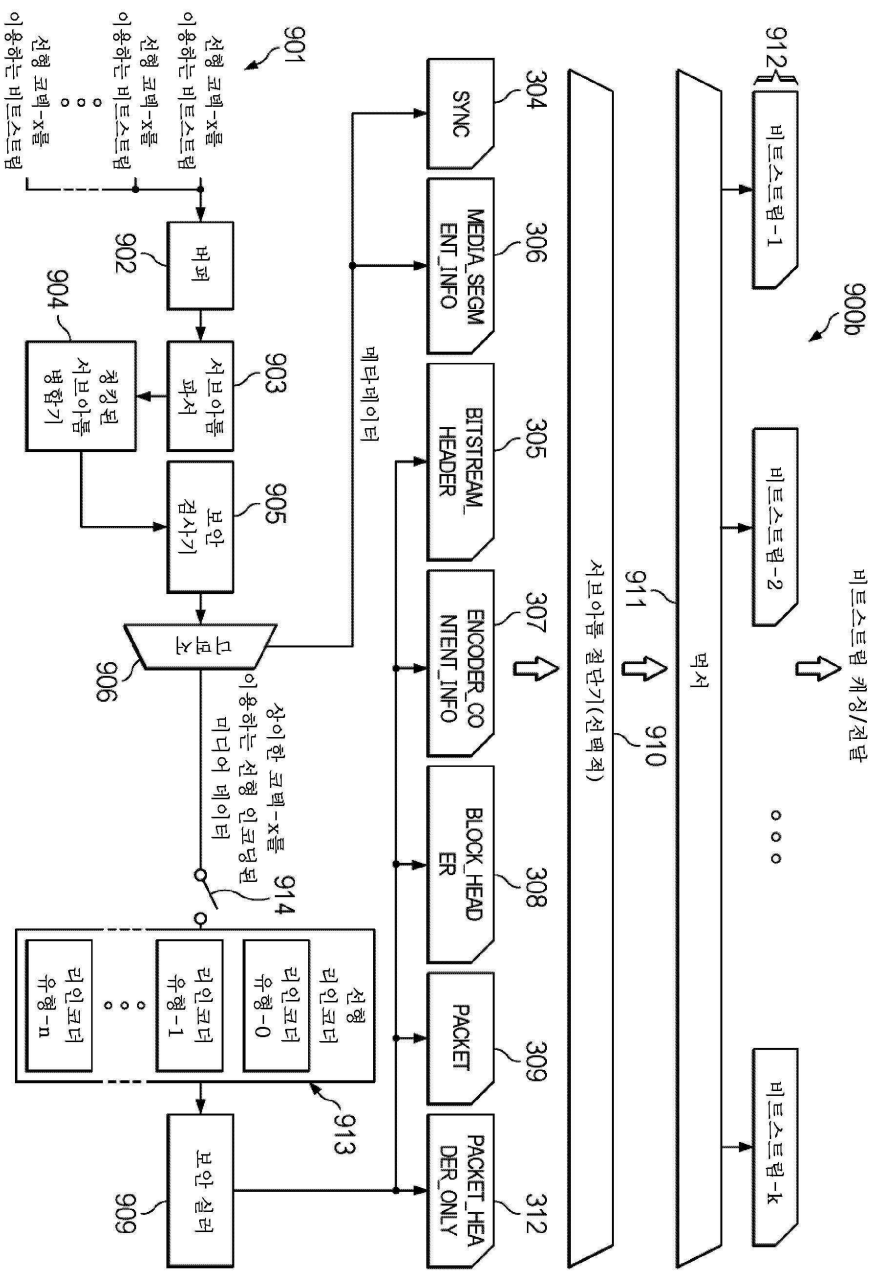
도면8b


800b
801
선형 코덱-x를 이용하는 비트스트림
선형 코덱-x를 이용하는 비트스트림
선형 코덱-x를 이용하는 비트스트림
802
버퍼를 이용하는 다운로더
803
서브아틀 파서
804
청킹된 서브아틀 병합기
805
보안 검사기
806
디멀티플렉서
선형 인코딩된 미디어 데이터
코덱-x를 이용하는
810
808
선형 디코더
디코더 유형-0
디코더 유형-1
디코더 유형-n
다른 메타데이터
렌더링 미디어 데이터
809
렌더러
소비가능한 미디어 데이터

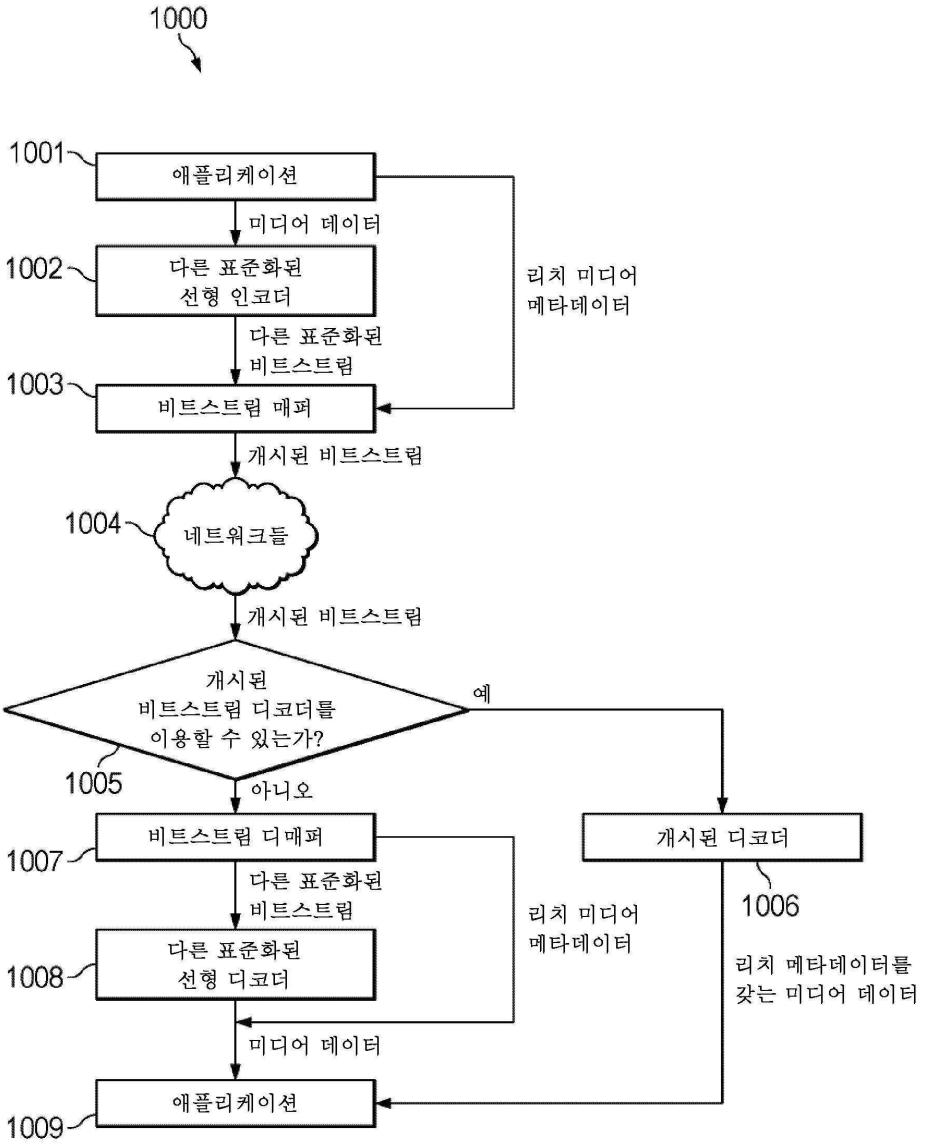
도면9a


900a
912
비트스트림-1
비트스트림-2
비트스트림-k
비트스트림 캐싱/전달
먹서
911
서브아틈 절단기(선택적)
910
901
선형 코덱-1을 이용하는 비트스트림
선형 코덱-2를 이용하는 비트스트림
선형 코덱-n을 이용하는 비트스트림
버퍼
902
서브아틈 파서
903
청킹된 서브아틈 병합기
904
보안 검사기
905
디먹서
906
메타데이터
코덱들을 이용하는 선형 인코딩된 미디어 데이터
코딩 통합기
907
선형 리인코더
908
보안 실러
909
SYNC
304
MEDIA_SEGMENT_INFO
306
BITSTREAM_HEADER
305
ENCODER_CONTENT_INFO
307
BLOCK_HEADER
308
PACKET
309
PACKET_HEADER_ONLY
312

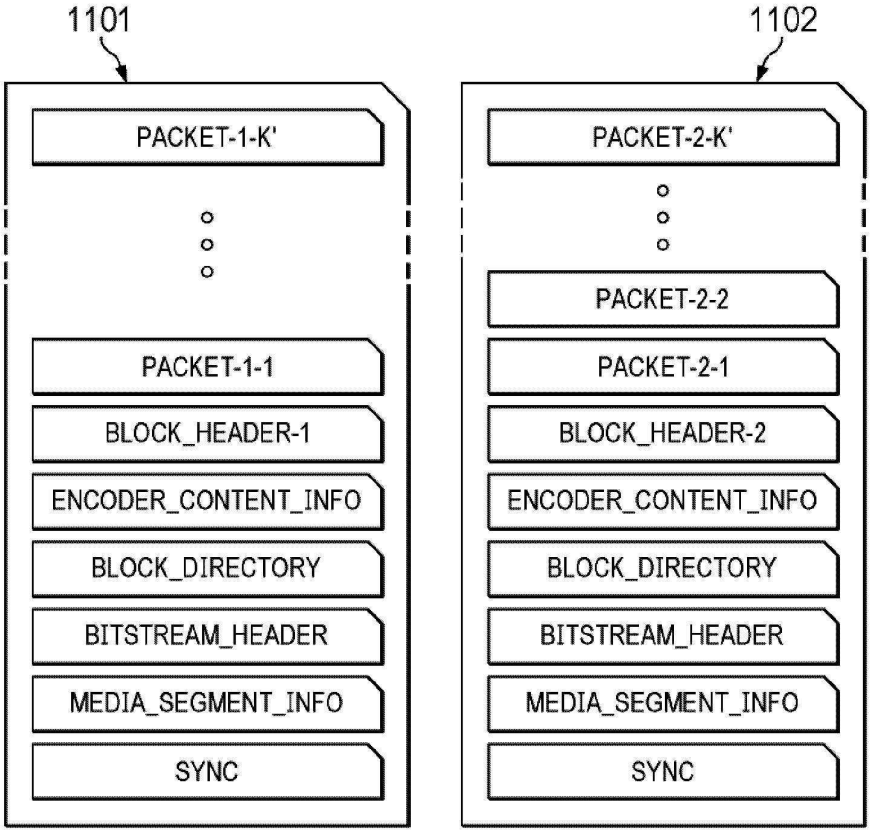
도면9b


900b
912
비트스트림-1
비트스트림-2
비트스트림-k
먹서
911
서브아톰 절단기(선택적)
910
비트스트림 캐싱/전달
304
SYNC
306
MEDIA_SEGMENT_INFO
305
BITSTREAM_HEADER
307
ENCODER_CONTENT_INFO
308
BLOCK_HEADER
309
PACKET
312
PACKET_HEADER_ONLY
메타데이터
901
선형 코덱-x를 이용하는 비트스트림
선형 코덱-x를 이용하는 비트스트림
선형 코덱-x를 이용하는 비트스트림
902
버퍼
903
서브아톰 파서
904
쳥킹 뤄 서브아톰 병합기
905
보안 검사기
906
스위치
상이한 코덱-x를 이용하는 선형 인코딩 뤄 미디어 데이터
914
913
선형 리인코더
리인코더 유형-0
리인코더 유형-1
리인코더 유형-n
909
보안 실러

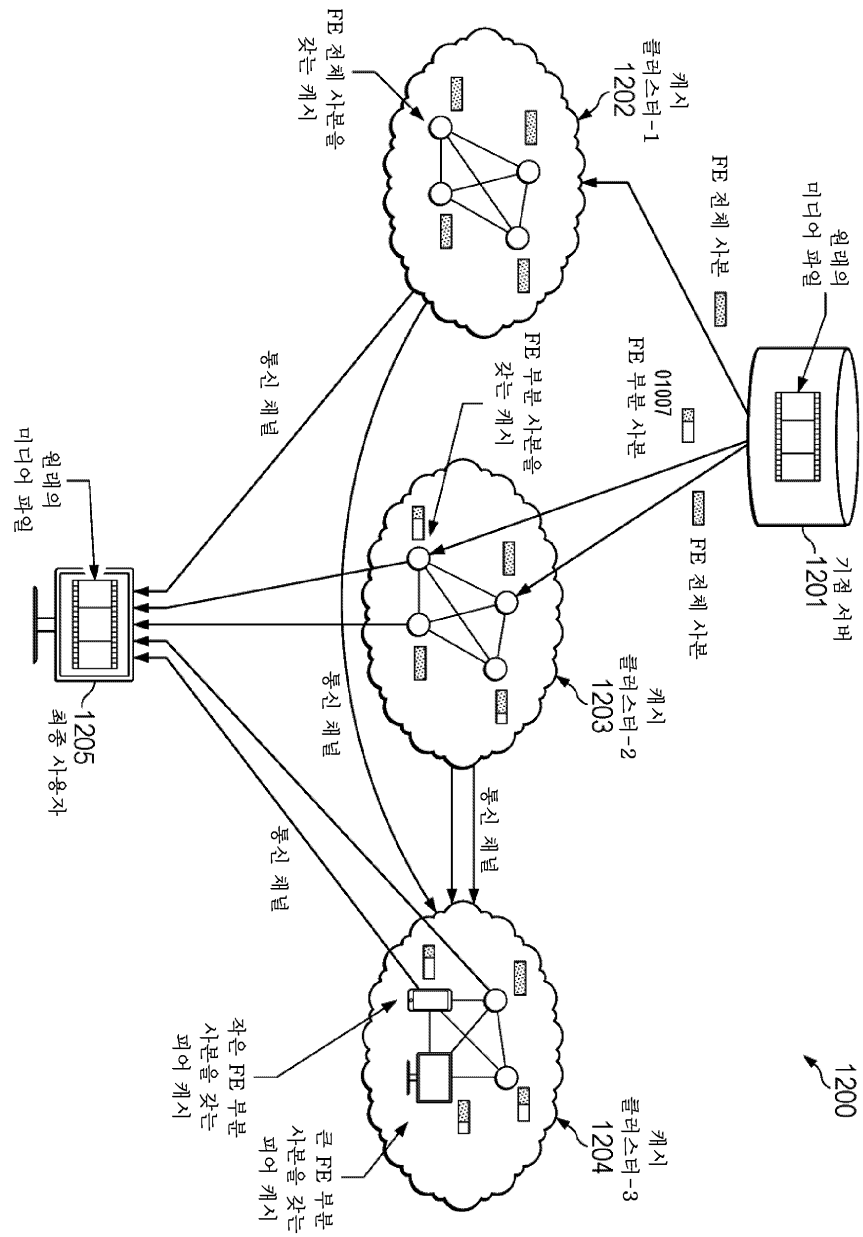
도면10


1000
1001
애플리케이션
미디어 데이터
1002
다른 표준화된
선형 인코더
다른 표준화된
비트스트림
1003
비트스트림 매퍼
개시된 비트스트림
1004
네트워크들
개시된 비트스트림
개시된
비트스트림 디코더를
이용할 수 있는가?
1005
예
아니오
리치 미디어
메타데이터
비트스트림 디매퍼
1007
다른 표준화된
비트스트림
다른 표준화된
선형 디코더
1008
미디어 데이터
애플리케이션
1009
개시된 디코더
1006
리치 미디어
메타데이터
리치 메타데이터를
갖는 미디어 데이터

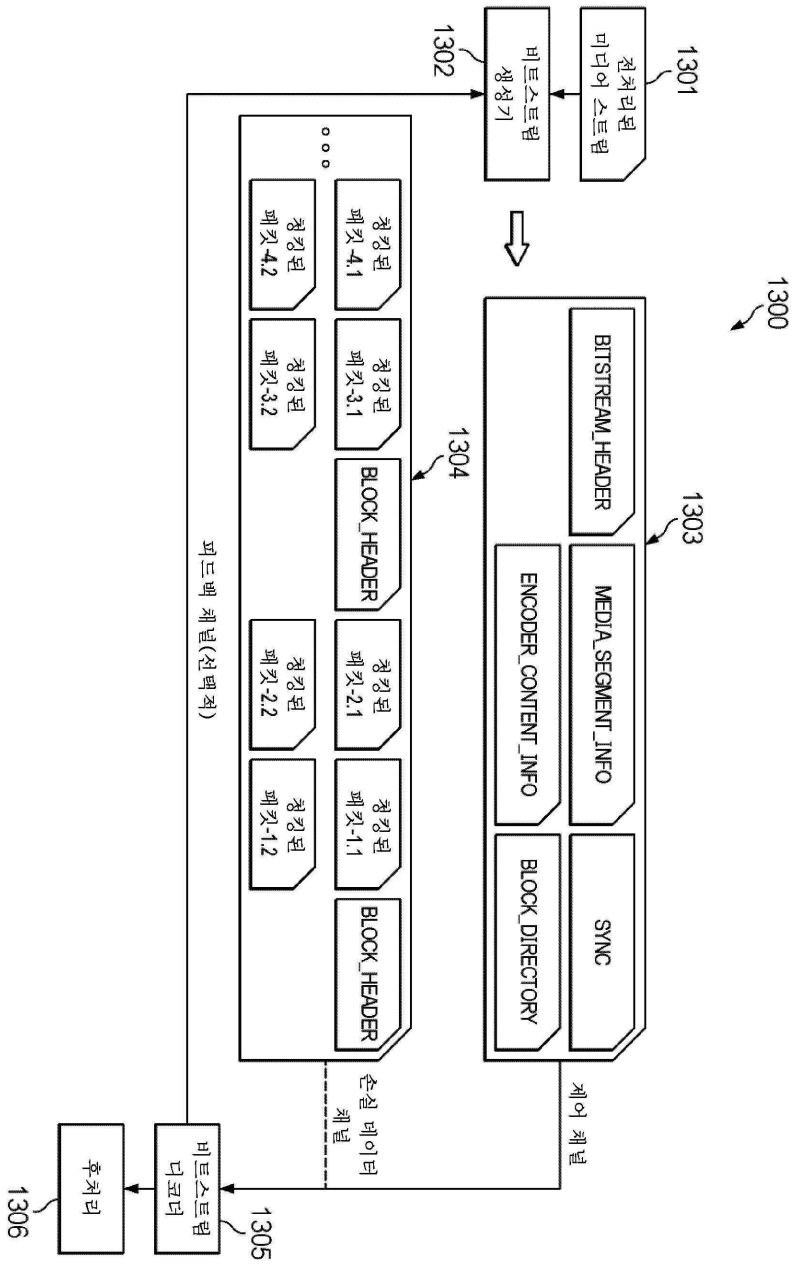
도면11


1101
PACKET-1-K'
PACKET-1-1
BLOCK_HEADER-1
ENCODER_CONTENT_INFO
BLOCK_DIRECTORY
BITSTREAM_HEADER
MEDIA_SEGMENT_INFO
SYNC
1102
PACKET-2-K'
PACKET-2-2
PACKET-2-1
BLOCK_HEADER-2
ENCODER_CONTENT_INFO
BLOCK_DIRECTORY
BITSTREAM_HEADER
MEDIA_SEGMENT_INFO
SYNC

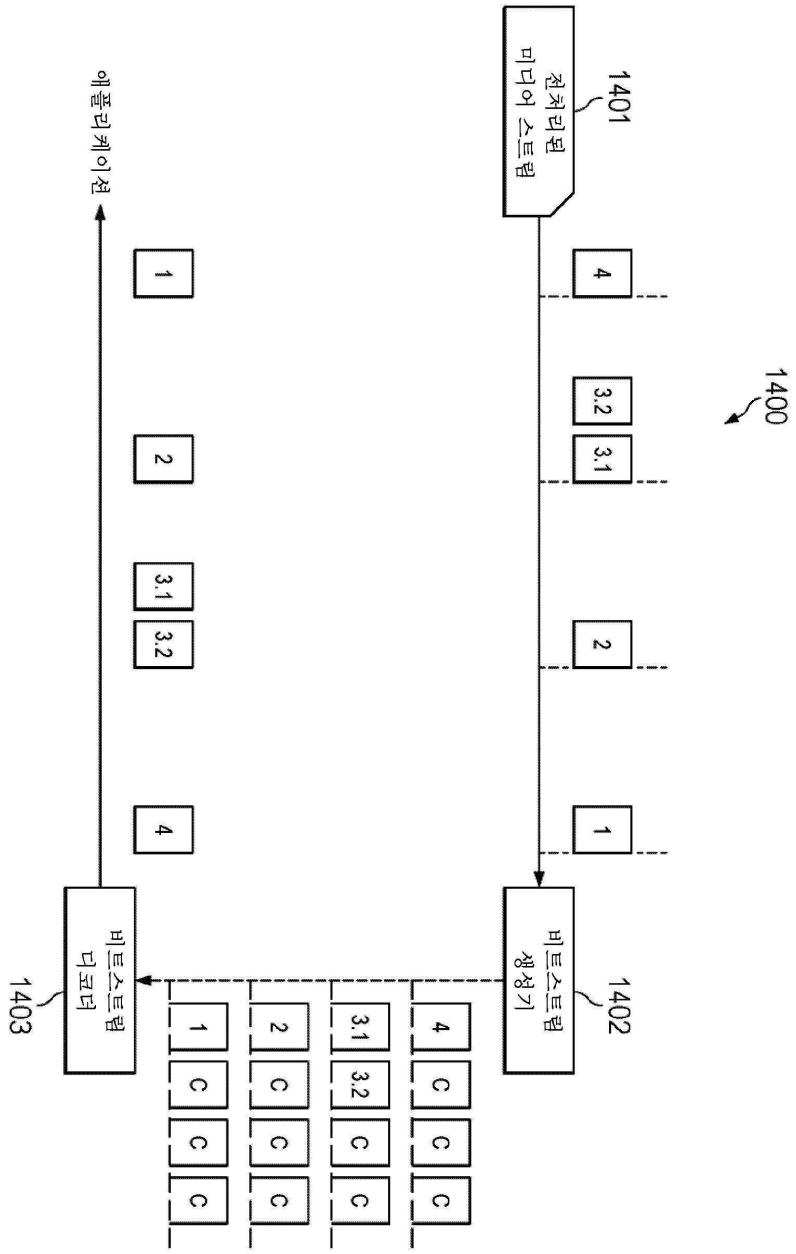
도면12


1200
원래의
미디어 파일
기점 서버
1201
FE 전체 사본
FE 전체 사본
01007
FE 부분 사본
FE 부분 사본을
갖는 캐시
캐시
클러스터-1
1202
FE 전체 사본을
갖는 캐시
캐시
클러스터-2
1203
캐시
클러스터-3
1204
통신 채널
통신 채널
통신 채널
통신 채널
원래의
미디어 파일
최종 사용자
1205
작은 FE 부분
사본을 갖는
피어 캐시
큰 FE 부분
사본을 갖는
피어 캐시

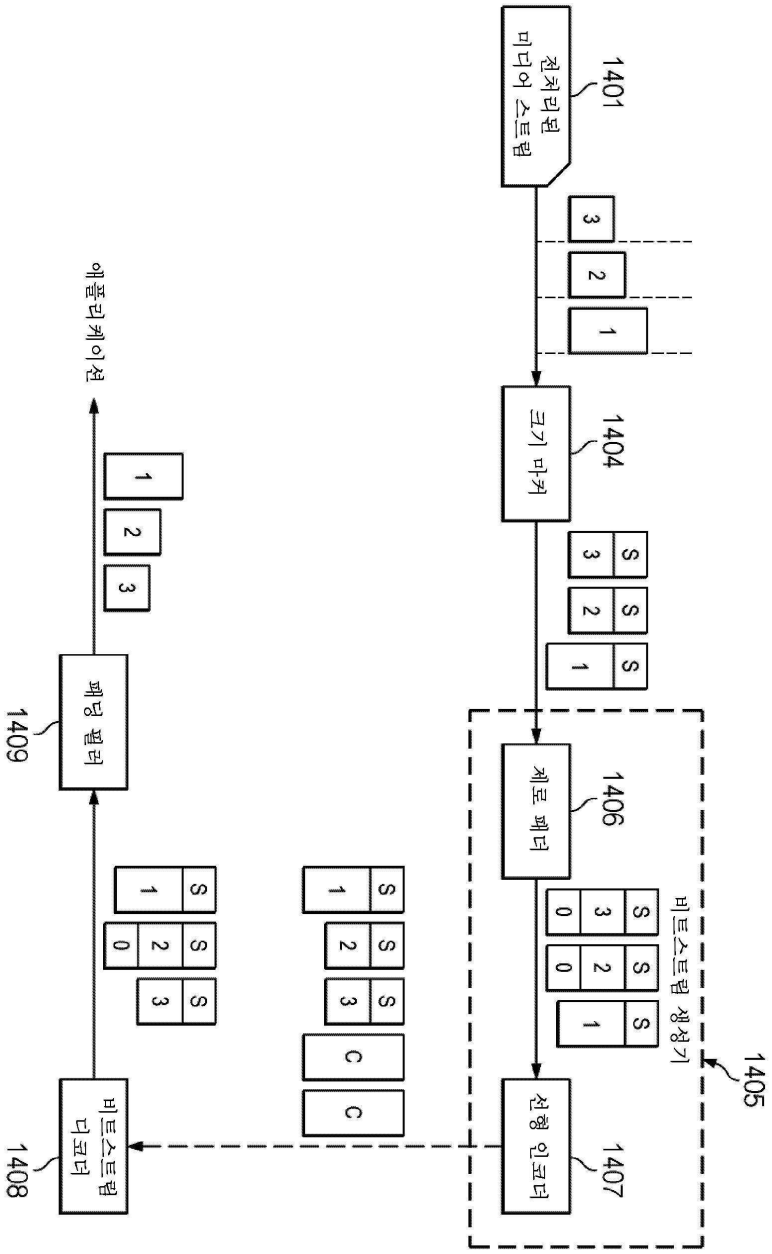
도면13


1301
전처리된 미디어 스트림
1302
비트스트림 생성기
1300
BITSTREAM_HEADER
MEDIA_SEGMENT_INFO
SYNC
ENCODER_CONTENT_INFO
BLOCK_DIRECTORY
1303
1304
BLOCK_HEADER
청킹된 패킷-1.1
청킹된 패킷-2.1
청킹된 패킷-3.1
청킹된 패킷-4.1
BLOCK_HEADER
청킹된 패킷-1.2
청킹된 패킷-2.2
청킹된 패킷-3.2
청킹된 패킷-4.2
제어 채널
손실 데이터 채널
피드백 채널(선택적)
1305
비트스트림 디코더
1306
후처리

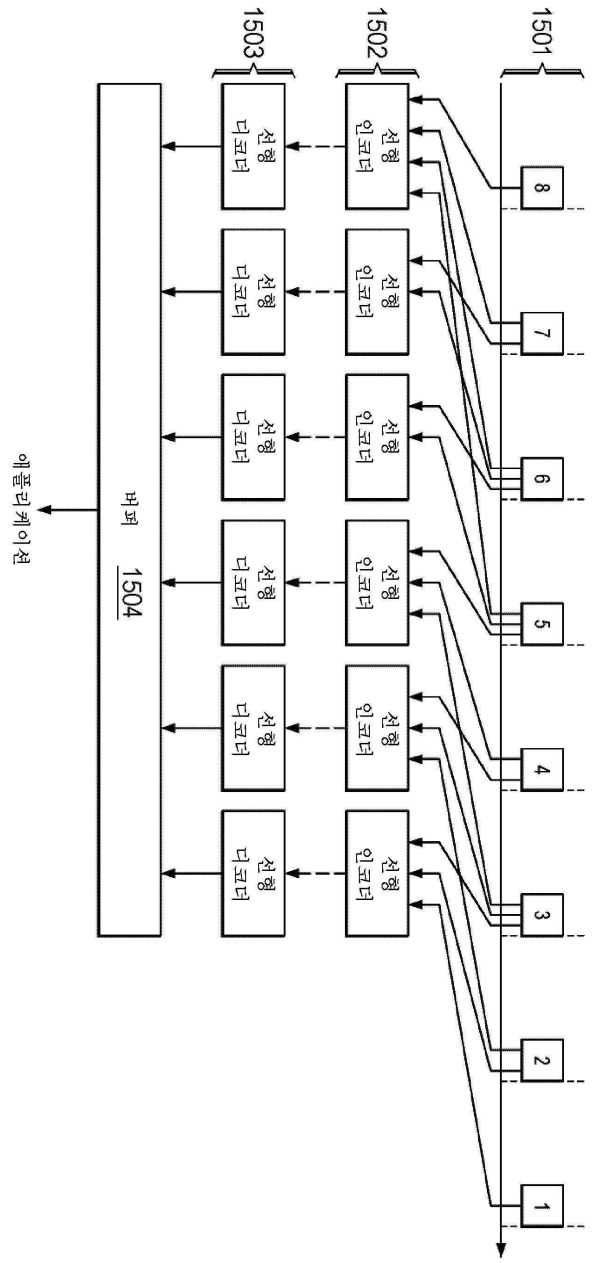
도면14a


전처리된
미디어 스트림
1401
1400
4
3.2
3.1
2
1
비트스트림
생성기
1402
비트스트림
디코더
1403
애플리케이션
1
2
3.1
3.2
4
C

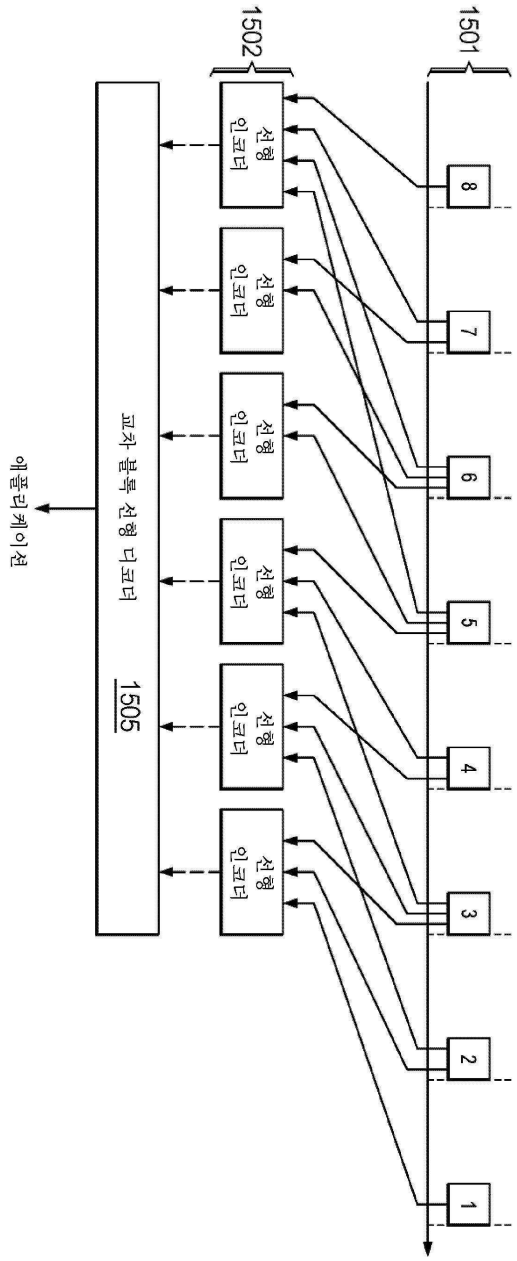
도면14b


전처리된 미디어 스트림
1401
크기 마커
1404
제로 패더
1406
선형 인코더
1407
비트스트림 생성기
1405
패딩 풀러
1409
비트스트림 디코더
1408
애플리케이션

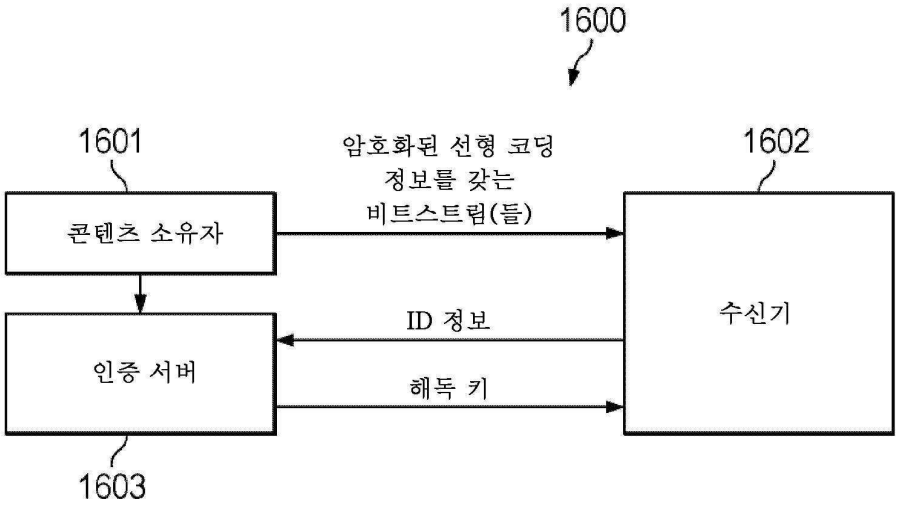
도면15a



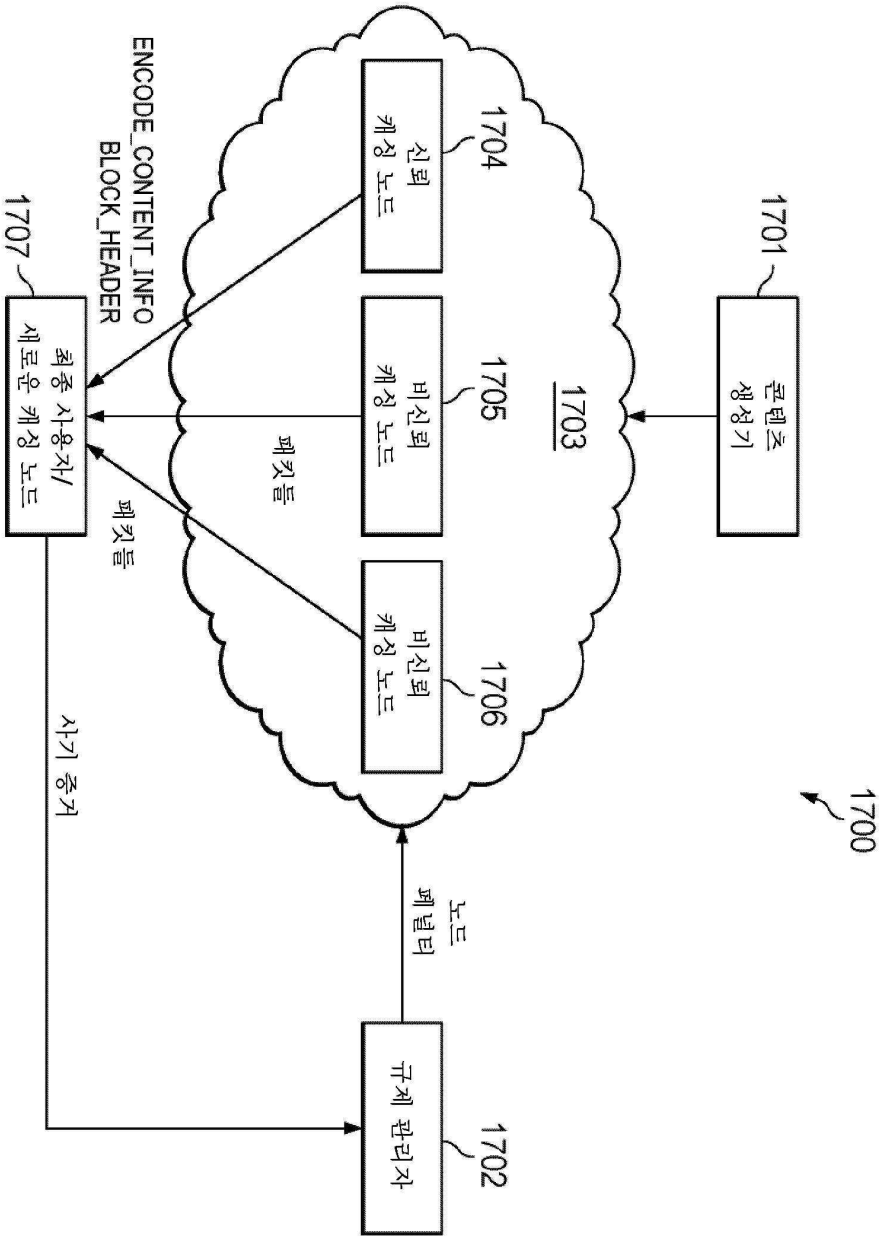
도면15b


1502
1501
선형 인코더
선형 인코더
선형 인코더
선형 인코더
선형 인코더
선형 인코더
8
7
6
5
4
3
2
1
교차 블록 선형 디코더
1505
애플리케이션

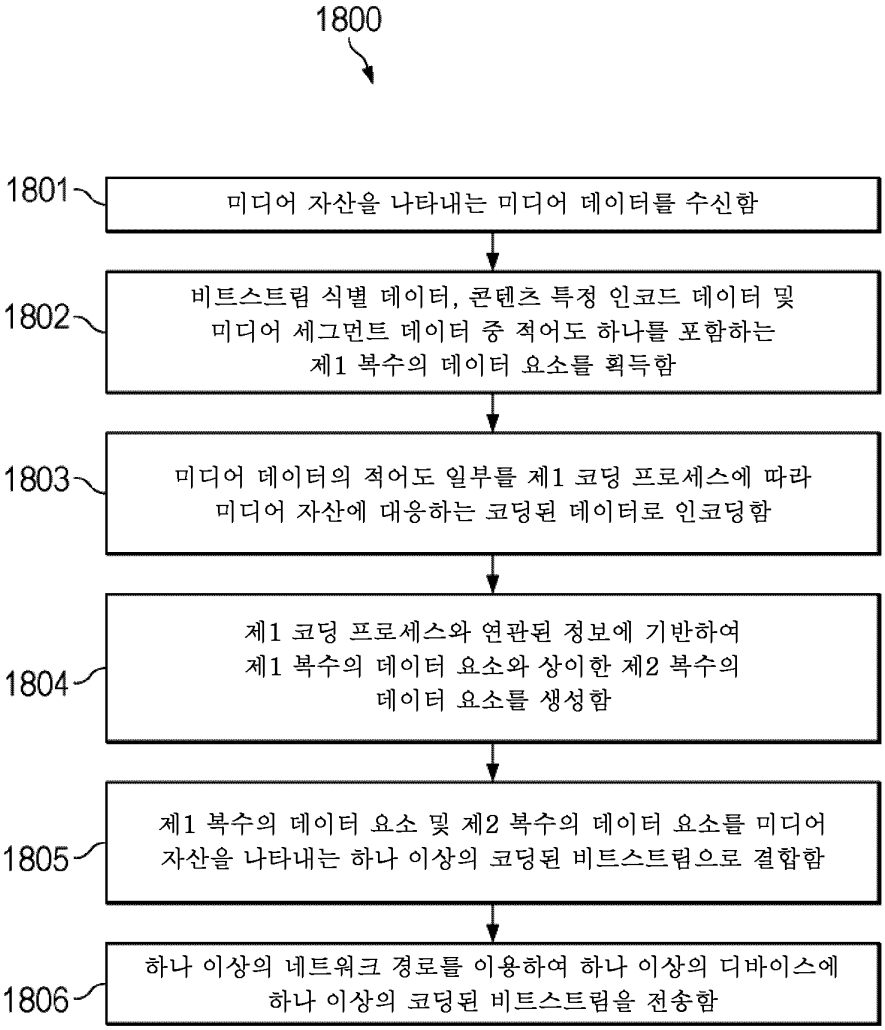
도면16


1600
1601
콘텐츠 소유자
암호화된 선형 코딩
정보를 갖는
비트스트림(들)
1602
수신기
ID 정보
인증 서버
해독 키
1603

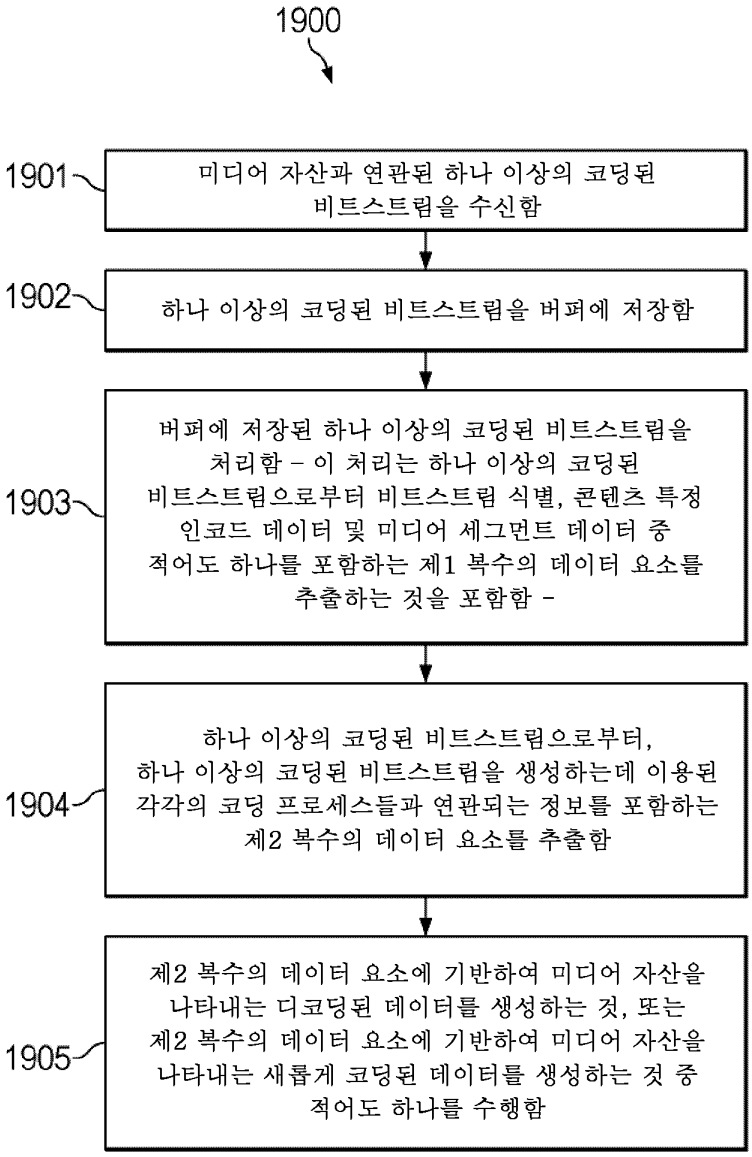
도면17


1700
1701
콘텐츠 생성기
1703
1704
신뢰 캐싱 노드
1705
비신뢰 캐싱 노드
1706
비신뢰 캐싱 노드
1707
최종 사용자/ 새로운 캐싱 노드
ENCODE_CONTENT_INFO
BLOCK_HEADER
패킷들
패킷들
사기 증거
노드 페널티
1702
규제 관리자

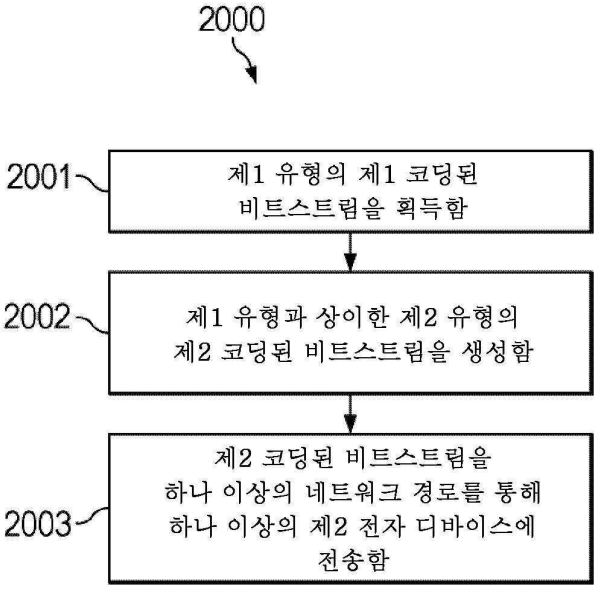
도면18


1800
1801
미디어 자산을 나타내는 미디어 데이터를 수신함
1802
비트스트림 식별 데이터, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 획득함
1803
미디어 데이터의 적어도 일부를 제1 코딩 프로세스에 따라 미디어 자산에 대응하는 코딩된 데이터로 인코딩함
1804
제1 코딩 프로세스와 연관된 정보에 기반하여 제1 복수의 데이터 요소와 상이한 제2 복수의 데이터 요소를 생성함
1805
제1 복수의 데이터 요소 및 제2 복수의 데이터 요소를 미디어 자산을 나타내는 하나 이상의 코딩된 비트스트림으로 결합함
1806
하나 이상의 네트워크 경로를 이용하여 하나 이상의 디바이스에 하나 이상의 코딩된 비트스트림을 전송함

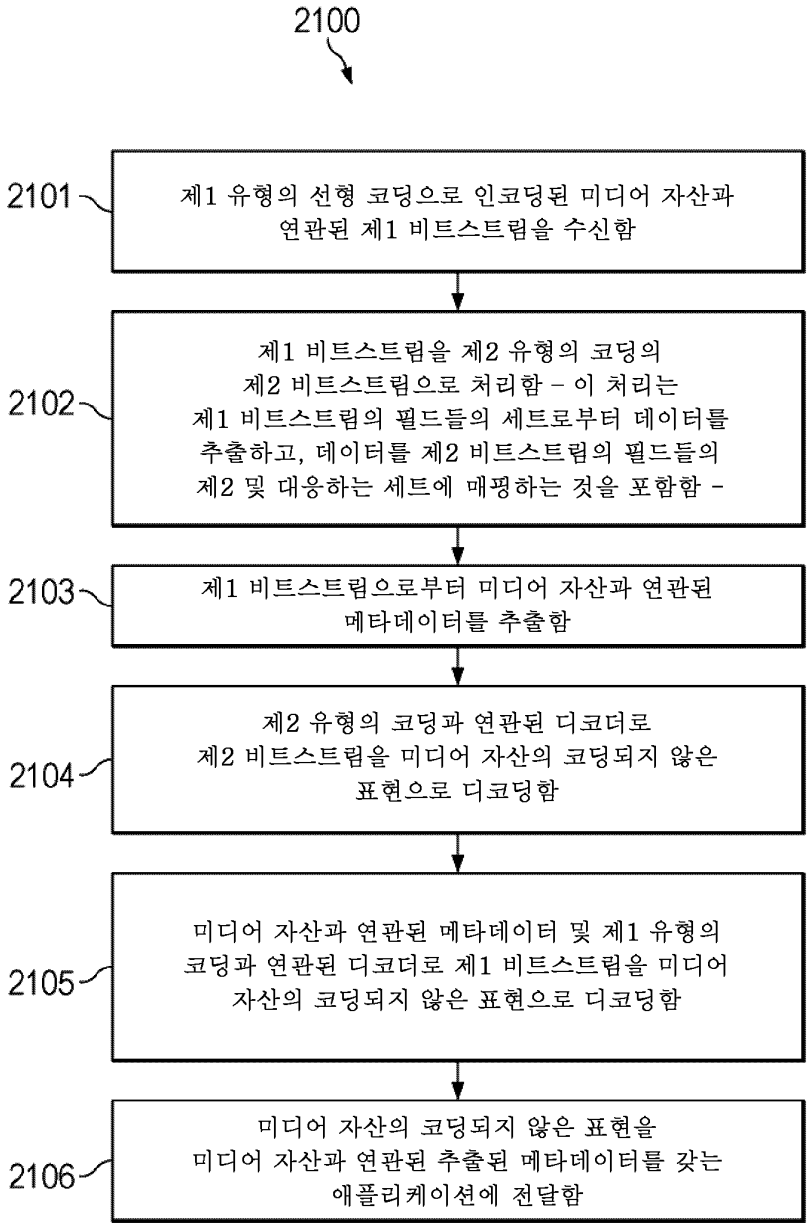
도면19


1900
1901
미디어 자산과 연관된 하나 이상의 코딩된 비트스트림을 수신함
1902
하나 이상의 코딩된 비트스트림을 버퍼에 저장함
1903
버퍼에 저장된 하나 이상의 코딩된 비트스트림을 처리함 - 이 처리는 하나 이상의 코딩된 비트스트림으로부터 비트스트림 식별, 콘텐츠 특정 인코드 데이터 및 미디어 세그먼트 데이터 중 적어도 하나를 포함하는 제1 복수의 데이터 요소를 추출하는 것을 포함함 -
1904
하나 이상의 코딩된 비트스트림으로부터, 하나 이상의 코딩된 비트스트림을 생성하는데 이용된 각각의 코딩 프로세스들과 연관되는 정보를 포함하는 제2 복수의 데이터 요소를 추출함
1905
제2 복수의 데이터 요소에 기반하여 미디어 자산을 나타내는 디코딩된 데이터를 생성하는 것, 또는 제2 복수의 데이터 요소에 기반하여 미디어 자산을 나타내는 새롭게 코딩된 데이터를 생성하는 것 중 적어도 하나를 수행함

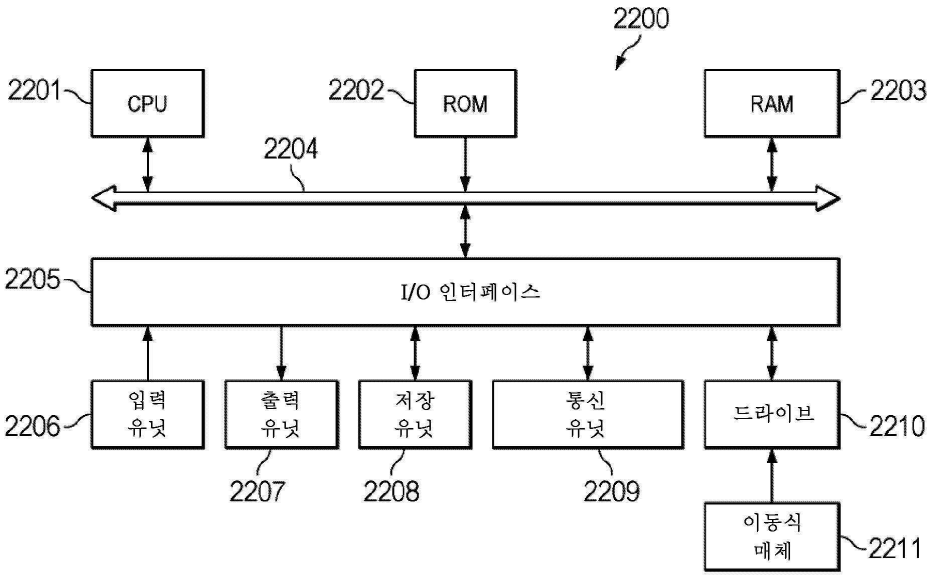
도면20


2000
2001
제1 유형의 제1 코딩된 비트스트림을 획득함
2002
제1 유형과 상이한 제2 유형의 제2 코딩된 비트스트림을 생성함
2003
제2 코딩된 비트스트림을 하나 이상의 네트워크 경로를 통해 하나 이상의 제2 전자 디바이스에 전송함

도면21


2100
2101
제1 유형의 선형 코딩으로 인코딩된 미디어 자산과 연관된 제1 비트스트림을 수신함
2102
제1 비트스트림을 제2 유형의 코딩의 제2 비트스트림으로 처리함 - 이 처리는 제1 비트스트림의 필드들의 세트로부터 데이터를 추출하고, 데이터를 제2 비트스트림의 필드들의 제2 및 대응하는 세트에 매핑하는 것을 포함함 -
2103
제1 비트스트림으로부터 미디어 자산과 연관된 메타데이터를 추출함
2104
제2 유형의 코딩과 연관된 디코더로 제2 비트스트림을 미디어 자산의 코딩되지 않은 표현으로 디코딩함
2105
미디어 자산과 연관된 메타데이터 및 제1 유형의 코딩과 연관된 디코더로 제1 비트스트림을 미디어 자산의 코딩되지 않은 표현으로 디코딩함
2106
미디어 자산의 코딩되지 않은 표현을 미디어 자산과 연관된 추출된 메타데이터를 갖는 애플리케이션에 전달함

도면22


2200
2201
CPU
2202
ROM
RAM
2203
2204
2205
I/O 인터페이스
2206
입력
유닛
출력
유닛
저장
유닛
통신
유닛
드라이브
2210
2207
2208
2209
이동식
매체
2211