２０６は、コンテンツ・ストリームから１または複数のキー・パッケージを抽出する。暗号解読エンジン２０６は、キー・パッケージを暗号解読するためにキー・ストア／マネージャ２１０へスクリーナ・キーを要求し、それにより、暗号化されたコンテンツ・ユニットに関連する１または複数のコンテンツ・キーを抽出することが可能にされる。暗号解読エンジン２０６は、１または複数のコンテンツ・キーを使用して、暗号化されたコンテンツ・ユニットを暗号解読することができる。暗号解読エンジン２０６は更に、暗号解読されたコンテンツ・ユニットをＣＯＤＥＣ２０８へ提供することができる。

【００４１】

コピー防止保護装置２２２（オプション）は、解凍されたコンテンツをデスクランブラ装置などへ提供する前に、その解凍されたコンテンツをスクランブリング、ダーティイング、あるいは暗号化することによって、解凍されたコンテンツの追加の保護を可能にする。従って、コピー防止保護装置２２２は、コンテンツがセキュア・プレーヤ２００を出た後のコンテンツの或るレベルの保護を可能にする。

【００４２】

タンパ・エージェント２２０は、セキュア・プレーヤ２００内の構成要素を監視し、セキュア・プレーヤ２００自体を含む何れかのコンポーネントが無許可の活動で改変されているかどうか、または無許可の活動に関連しているかどうかを判定するように、動作可能とされる。一実施形態では、タンパ検出／応答保護装置２２０は、少なくともＦＩＰＳセキュリティ・レベル３で動作する。

【００４３】

タンパ・エージェント２２０は、その監視結果に基づいて応答を提供する。そのような応答は、セキュア・コンテンツを消去または使用不能にするよう指示すること、セキュア・プレーヤ２００の動作をロックすること、セキュア・プレーヤの公開／秘密鍵、スクリーナ・キー、コンテンツ・キーなどを消去すること、ならびに検出した無許可の活動を報告することを含むことができる。

【００４４】

図３に、セキュア・コンテンツを提供するコンテンツ・ストリームの一実施形態を示す。コンテンツ・ストリーム３００は、コンテンツの適切なストリームの一例に過ぎず、本発明の使用または機能の範囲に関して何らかの制限を示唆するものではない。従って、本発明の範囲から逸脱することなく、他の知られた構造および構成を使用することができる。

【００４５】

図に示すように、コンテンツ・ユニット・ストリーム３００は、コンテンツ・ユニット３０１、３０３～３０６、およびキー・パッケージ３０２を含む。１つのキー・パッケージ（３０２）のみを図示しているが、コンテンツ・ストリーム３００は実質的には任意の数のキー・パッケージを含むことができることに留意されたい。

【００４６】

コンテンツ・ユニット３０１、および３０３～３０６は、様々なコンテンツ・フォーマットを含むことができる。例えば、コンテンツは、ＭＰＥＧ（Ｍｏｖｉｎｇ　Ｐｉｃｔｕｒｅ　Ｅｘｐｅｒｔｓ　Ｇｒｏｕｐ）フォーマットを使用してフォーマットすることができる。コンテンツ・ユニット３０１、３０３～３０６は、ＭＰＥＧコンテンツ・フォーマットに限定されず、本発明の範囲または精神から逸脱することなくＪＰＥＧフォーマット、ＭＰ３フォーマットなどを含むその他のコンテンツフォーマットを使用することができる。しかし、一例として、例示を容易にするために本明細書ではＭＰＥＧフォーマットを使用する。

【００４７】

簡潔に述べると、ＭＰＥＧは、デジタル・ブロードキャスト（放送）・コンテンツ用の符号化および圧縮の標準である。ＭＰＥＧは、ビデオ・ブロードキャスト・コンテンツのテレビジョン品質伝送用の圧縮サポートを提供する。更に、ＭＰＥＧは、圧縮されたオーディオ、制御、更にはユーザ・ブロードキャスト・コンテンツを提供する。

【００４８】
ＭＰＥＧコンテンツ・ストリームは、複数個のエレメンタリ・ストリーム（ＥＳ）アクセス・ユニットの固定（または可変サイズ）のブロックまたはフレームを典型的には含むパケット化エレメンタリ・ストリーム（ＰＥＳ）を含む。ＥＳは、典型的には、ＭＰＥＧコンテンツ・ストリームの基本構成要素であり、デジタル制御データ、デジタル・オーディオ、デジタル・ビデオ、およびその他のデジタル・コンテンツ（同期または非同期）を含む。実質的に同じタイムベースに対して参照される１群の密に結合したＰＥＳパケットは、ＭＰＥＧプログラム・ストリーム（ＰＳ）を含む。独立したタイムベースを含むことも可能であり、１または複数のコンテンツ・ストリームを組み合わせるための汎用的手法を形成する、ＭＰＥＧトランスポート・ストリーム（ＴＳ）として知られている固定サイズのトランスポート・パケットへと、各ＰＥＳパケットを分解することもできる。更に、ＭＰＥＧフレームは、Ｉフレーム（イントラフレーム（intra-frame））、Ｐフレーム（forward predicted frame）、およびＢフレーム（bi-directional predicted frame）を含む。

【００４９】
コンテンツ・ユニット３０１および３０３～３０６は、それぞれ、様々な基準に基づいてデータのユニットへと区分されるコンテンツ・ストリームの部分を含むことができる。例えば、コンテンツ・ユニット３０１および３０３～３０６は、ビデオ・エレメンタリ・ストリーム（ＥＳ）、オーディオＥＳ、デジタル・データＥＳ、ならびにコンテンツ・ストリームのビデオ、オーディオ、データ・エレメンタリ・ストリームを任意に組み合わせたものから抽出されたデータの部分を含むことができる。例えば、コンテンツ・ユニット３０１および３０３～３０６は、ビデオＥＳの１０秒の部分で構成することができる。更に、コンテンツ・ユニット３０１および３０３～３０６は、コンテンツ・ストリームからの同じ長さ、密度などのコンテンツを含む必要はない。

【００５０】
１または複数のコンテンツ・キーを使用してコンテンツ・ユニット３０１および３０３～３０６を選択的に暗号化することができる。即ち、コンテンツ・ユニット３０１および３０３を暗号化し、コンテンツ・ユニット３０４～３０６を明文のままにすることができる。更に、暗号化は、ビデオ・エレメンタリ・ストリーム（ＥＳ）、オーディオＥＳ、デジタル・データＥＳ、ならびにコンテンツ・ストリーム３００を構成するビデオ、オーディオ、データ・エレメンタリ・ストリームを任意に組み合わせたものの少なくとも一部に、選択的に適用することができる。選択的暗号化は、Ｉフレーム、Ｐフレーム、Ｂフレーム、ならびにＰ、ＢおよびＩフレームを任意に組み合わせたものの少なくとも一部を選択的に暗号化することを更に含むことができる。

【００５１】
キー・パッケージ３０２は、コンテンツ・ユニットを暗号化するのに使用される１または複数のコンテンツ・キーと、コンテンツ・ストリーム３００に関連するコンテンツ識別子とを含むことができる。キー・パッケージ３０２はまた、コンテンツ・ストリーム３００に関連するアクセス制約、資格などを含むことができる。キー・パッケージ３０２は、何れのコンテンツ・キーがコンテンツ・ストリーム３００の何れのコンテンツ・ユニット（３０１、３０２～３０６）と関連するかを示す同期情報を更に含むことができる。

【００５２】
キー・パッケージ３０２は、対象のセキュア・プレーヤの公開／秘密鍵を使用して暗号化することができる。一実施形態では、対象のセキュア・プレーヤの公開／秘密鍵がＥＩＰＳレベル４装置で生成される。しかし、本発明はそのように限定されるものではなく、対象のセキュア・プレーヤの公開／秘密鍵を生成するために、それより低いセキュリティレベルを使用することもできる。一実施形態では、キー・パッケージ３０２は明文のままにされ、コンテンツ・キー（１又は複数）だけが対象のセキュア・プレーヤの公開／秘密鍵で暗号化される。

【００５３】

一般化されたオペレーション

次に、本発明の諸態様のオペレーションを図４～８と関連して説明する。

図４に、セキュア・コンテンツを提供および閲覧するエンド・ツー・エンド・プロセスについての一実施形態を全般的に示す流れ図を示す。プロセス４００は、例えば、図１の動作環境１００内で実施することができる。

【００５４】

プロセス４００は、スタート・ブロックの後、ブロック４０２で始まる。ブロック４０２は図５に関連して以下でより詳細に説明する。しかし、簡潔に述べると、ブロック４０２では、コンテンツ・ストリーム内の少なくとも１つのコンテンツ・ユニットを選択的に暗号化することによって、セキュア・コンテンツが作成される。選択されたコンテンツ・ユニットは、少なくとも１つのコンテンツ・キーを使用して暗号化される。使用するコンテンツ・キー（１又は複数）を、追加の情報と共に更に暗号化し、コンテンツ・ストリーム内に埋め込むことができる。変更されたコンテンツ・ストリームは、図１のコンテンツ媒体１１２などのようなコンテンツ媒体に転送される。

【００５５】

処理はブロック４０４に進み、そこで、コンテンツ媒体およびキー・パッケージが、図１のユーザ１１４などのようなユーザへ配布される。コンテンツ媒体は、メールなどを含む様々な機構を使用して配布することができる。スクリーナ・キー・モジュールは、メモリ・スティック、スマートカード、ＤＶＤ，ディスク，テープなどを含むことができる。スクリーナ・キー・モジュールは、コンテンツ媒体のために使用されるのとは異なる配布機構によってユーザに配布することができる。スクリーナ・キー・モジュールは、例えば、上述のハード媒体を使用することにより、また、ネットワークを介する伝送により、また、メールにより、また、様々な他の配布機構により、配布することができる。

【００５６】

処理はブロック４０６に進む。ブロック４０６は、図６と関連して以下でより詳細に説明する。しかし、簡潔に述べると、ブロック４０６では、コンテンツ媒体上にあるコンテンツ・ストリームを暗号解読および閲覧するために、セキュア・プレーヤがスクリーナ・キー・モジュールおよび識別モジュールと共に使用される。ブロック４０６での動作が完了すると、処理は別の動作の処理へと戻る。

【００５７】

図５に、セキュア・コンテンツを生成するプロセスについての一実施形態を全般的に示す流れ図を示す。プロセス５００は、例えば、図１の動作環境１００内で実施することができる。

【００５８】

プロセス５００は、スタートブロックの後で、ブロック５０２で始まり、そこで、コンテンツ・ユニットのストリームが作成される。コンテンツ所有者、プロデューサなどは、図３と関連して上述したように、様々な基準に基づいてコンテンツ・ストリームをデータのユニットへと分割することによって、コンテンツ・ユニットのストリームを作成することができる。

【００５９】

処理はブロック５０４に進み、ここで、少なくとも１つのコンテンツ・キーが生成される。限定はしないが、次世代暗号化標準（ＡＥＳ）、ＲＳＡ、ＲＣ６、ＩＤＥＡ、ＤＥＳ，ＲＣ２、ＲＣ５、スキップジャック、およびその他の任意の対象暗号化アルゴリズムを含むいくつかの暗号化／暗号解読対称機構のいずれかを使用して、コンテンツ・キーを生成することができる。更に、そのような暗号化アルゴリズムは、適切である場合には、暗号ブロック・チェーン・モード、暗号フィードバック・モード、ＣＢＣ暗号文スチーリング（cipher text stealing）（ＣＴＳ）、ＣＦＢ、ＯＦＢ、カウンタ・モード、および／またはその他の任意のブロック・モードを使用することができる。一実施形態では、コンテンツ・キーは、少なくとも１２８ビットＡＥＳ暗号化／暗号解読アルゴリズムを使用して生成される。しかし、コンテンツ・キーの生成は対称鍵機構に限定されず、本発明の範

囲から逸脱することなく非対称機構を使用することもできる。

【００６０】

処理はブロック５０６に進み、そこで、少なくとも１つのコンテンツ・キーが使用されて、コンテンツ・ストリーム中のコンテンツ・ユニットが選択的に暗号化される。選択的暗号化は、暗号化のためにコンテンツ・ストリーム中のコンテンツ・ユニットをランダムに選択することや、コンテンツ・ストリーム中のＮ番目ごとのコンテンツ・ユニットを選択することなどを含むことができる。選択的暗号化はまた、コンテンツ・ユニット中のビデオ・エレメンタリ・ストリーム（ＥＳ）、オーディオＥＳ、デジタル・データＥＳ、ならびにビデオ、オーディオ、およびデータのエレメンタリ・ストリームを任意に組み合わせたもののうちの少なくとも一部などのような、コンテンツ・ユニットの少なくとも一部を選択的に暗号化することを含むことができる。選択的暗号化は、更に、コンテンツ・ユニットのＩフレーム、Ｐフレーム、Ｂフレーム、ならびにＰ、Ｂ、Ｉフレームを任意に組み合わせたものなどのような、コンテンツ・ユニット中のフレームを暗号化することを含むことができる。

【００６１】

更に、選択的暗号化は、選択したコンテンツ・ユニットを暗号化するために使用されるコンテンツ・キーを変更することを更に含むことができる。例えば、一実施形態では、１組のコンテンツ・キーが、例えば１０秒毎などのような或る基準で交代（循環）されて、コンテンツ・ユニットが暗号化される。

【００６２】

次に処理はブロック５０８に進み、そこで、ブロック５０８では、スクリーナ・キー・モジュールが作成される。スクリーナ・キー・モジュールは、コンテンツに関連するコンテンツ識別子およびスクリーナ・キーを含むことができる。スクリーナ・キー・モジュールはまた、コンテンツに関連する資格、権利なども含むことができる。更に、スクリーナ・キーは、対象のセキュア・プレーヤに結びつけられる公開／秘密鍵を使用して暗号化することができる。

【００６３】

処理はブロック５１０に進み、そこで、ブロック５１０ではキー・パッケージが作成される。キー・パッケージは、図３と関連して上述したものと実質的に同様に、少なくとも１つのコンテンツ・キー、コンテンツに関連するコンテンツ識別子、アクセス制約、資格などを含むことができる。一実施形態では、少なくとも１つのコンテンツ・キーが、スクリーナ・キーを使用して暗号化される。別の実施形態では、キー・パッケージが暗号化される。更に別の実施形態では、コンテンツに関連するコンテンツ識別子が暗号化されていない状態とされる。

【００６４】

処理はブロック５１２に進み、ブロック５１２では、キー・パッケージがコンテンツ・ユニット・ストリームに埋め込まれる。処理はブロック５１４へ進み、ブロック５１４では、変更されたコンテンツ・ユニットが、ＤＶＤ、高精細度ＤＶＤなどのようなコンテンツ媒体に書き込まれる。ブロック５１４が完了すると、処理は、他の動作を実施するように戻る。

【００６５】

図５のいくつかのブロックは、本発明の範囲または精神から逸脱することなく、異なるシーケンス、シーケンスの組み合わせなどで実施できることを理解されたい。例えば、ブロック５０６をブロック５０４の前に実施することができ、また、ブロック５０４と組み合わせて実施することもできる。

【００６６】

図６Ａ～６Ｂに、セキュア・コンテンツを頒布するプロセスについての一実施形態を全般的に示す流れ図を示す。プロセス６００は、例えば図２のセキュア・プレーヤ２００内で実施することができる。

【００６７】

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0396
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

プロセス６００は、スタートブロックの後にブロック６０２で始まり、ブロック６０２では、コンテンツ・ユニットがコンテンツ媒体から読み取られる。処理は判断ブロック６０４へ進み、判断ブロック６０４では、読み取ったコンテンツ・ユニットが暗号化されているか否かの判定が行われる。読み取ったコンテンツ・ユニットが暗号化されていると判定された場合、処理はブロック６１０へと分岐し、そうでない場合、処理はブロック６０６へと分岐する。

【００６８】

ブロック６１０では、キー・パッケージがコンテンツ媒体上のコンテンツ・ストリームから抽出される。処理はブロック６１４へ進み、ブロック６１４では、コンテンツ識別子がキー・パッケージから抽出される。一実施形態では、コンテンツ識別子は既に「明文」である。処理はブロック６１６へ進み、ブロック６１６では、コンテンツ識別子が索引として使用されて、セキュア・コンテンツに関連するスクリーナ・キーが突き止められる。

【００６９】

処理は判断ブロック６１８へ進み、判断ブロック６１８では、コンテンツ識別子に関連するスクリーナ・キーが既存のスクリーナ・キーの既存のデータベース、ファイル、ディレクトリなどに位置するかどうかの判定が行われる。スクリーナが位置していない場合、処理はブロック６２０へと分岐し、そうでない場合、処理は判断ブロック６２６へと分岐する。

【００７０】

ブロック６２０では、スクリーナ・キー・モジュールへのアクセスを許可するためのユーザＩＤを得るためのリクエスト（要求）が行われる。この要求は、ユーザ名／パスワードの入力、バイオメトリック入力などを要求することができる。一実施形態では、上記の要求は、識別（ＩＤ）キーを含むスマートカードをユーザが挿入することを要求する。処理は判断ブロック６２２へ進み、判断ブロック６２２では、受け取ったＩＤが有効であるか否かの判定が行われる。受け取ったＩＤが有効である場合、処理はブロック６２４へと分岐する。そうでない場合、処理は戻って他の動作を実施する。一実施形態では、そのような他の動作は、限定するものではないが、ユーザが有効なＩＤの再入力を試みることを可能にすること、コンテンツ媒体を消去すること、ユーザがコンテンツ媒体にアクセスするのをロックすることなどを含むことができる。

【００７１】

ブロック６２４では、有効なユーザ識別を使用して、スクリーナ・キー・モジュール上に位置するスクリーナ・キーへのアクセスを可能にする。処理は判断ブロック６２６へ進む。

【００７２】

判断ブロック６２６では、確認されたユーザがコンテンツ・ユニットに対する適切なアクセス権、資格などを有するか否かの判定が行われる。有する場合、処理はブロック６２８へと分岐し、そうでない場合、処理は戻って、判断ブロック６２２での上述の動作などのような他の動作を実施する。

【００７３】

ブロック６２８では、対象のセキュア・プレーヤに結びつけられた秘密鍵を使用してスクリーナ・キーが暗号解読される。処理はブロック６３０へと進み、ブロック６３０では、暗号解読されたスクリーナ・キーを使用してコンテンツ・キーが暗号解読される。一実施形態では、暗号解読されたスクリーナ・キーを使用して、キー・パッケージが暗号解読され、コンテンツ・キーが抽出される。処理はブロック６３２へと進み、ブロック６３２では、暗号解読されたコンテンツ・キーを使用して、暗号化されたコンテンツ・ユニットが暗号解読される。処理はブロック６０６へ進む。

【００７４】

ブロック６０６では、ＣＯＤＥＣが使用され、現コンテンツ・ユニットが圧縮解除（解凍）される。次いで処理はブロック６０８へ進み、ブロック６０８では、解凍されたコンテンツが、ユーザが楽しむための装置、例えばテレビジョンなどへ供給される。一実施形

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0397
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

態では、ブロック６０８で、解凍されたコンテンツが更にコピー・プロテクト処理される。即ち、解凍されたコンテンツを転送する前に、その解凍されたコンテンツをオプションのコピー防止保護装置に通すことができる。ブロック６０８での動作が完了すると、プロセスは従って他の動作を実施する。

【００７５】

流れ図の各ブロック、および流れ図中のブロックの組み合わせを、コンピュータ・プログラム命令で実施できることが理解されよう。こうしたプログラム命令をプロセッサに提供してマシンを生成することができ、そのプロセッサ上で実行される命令が、流れ図のブロックで指定される動作を実施する手段を生成する。コンピュータ・プログラム命令をプロセッサで実行して、一連の動作ステップをプロセッサで実施させ、コンピュータで実施されるプロセスを生成することができ、そのプロセッサ上で実行される命令が、流れ図のブロックで指定される動作を実施するステップを提供する。

【００７６】

従って流れ図のブロックは、指定の動作を実施する手段の組合せと、指定の動作を実施するステップの組合せと、指定の動作を実施するプログラム命令手段とをサポートする。また、流れ図の各ブロック、および流れ図のブロックの組合せを、指定の動作またはステップを実施する専用目的のハードウェア・ベースのシステムや、専用目的のハードウェアとコンピュータ命令との組合せで実施できることも理解されたい。

【００７７】

上記の仕様、例、およびデータは、本発明の構成の製造および使用の完全な説明を提供する。本発明の精神および範囲から逸脱することなく本発明の多くの実施形態を作成することができるので、本発明は特許請求の範囲に存する。

【図面の簡単な説明】

【００７８】

【図１】図１は、本発明を実施することのできる例示的環境を示す。

【図２】図２は、セキュア・コンテンツの閲覧を可能にする例示的装置のブロック図を示す。

【図３】図３は、セキュア・コンテンツを提供するコンテンツ・ストリームの一実施形態を示す。

【図４】図４は、セキュア・コンテンツを提供および閲覧するエンドツーエンド・プロセスについての一実施形態を全般的に示す流れ図を示す。

【図５】図５は、セキュア・コンテンツを生成するプロセスについての一実施形態を全般的に示す流れ図を示す。

【図６Ａ】図６Ａおよび図６Ｂは、本発明によるセキュア・コンテンツを閲覧するプロセスについての一実施形態を全般的に示す流れ図を示す。

【図６Ｂ】図６Ａおよび図６Ｂは、本発明によるセキュア・コンテンツを閲覧するプロセスについての一実施形態を全般的に示す流れ図を示す。

【図１】

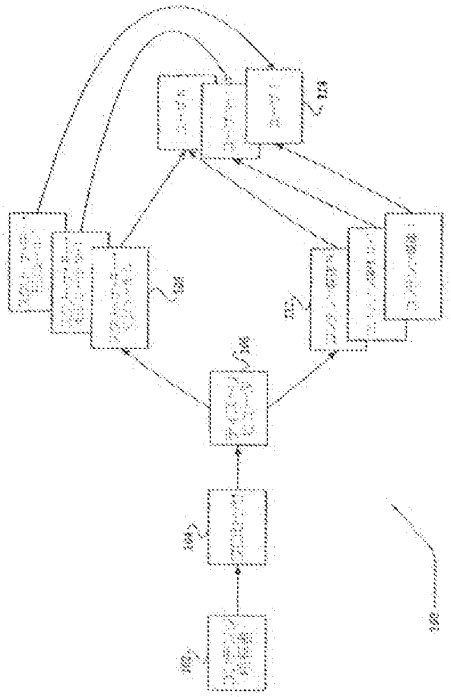


Fig. 1.

【図２】

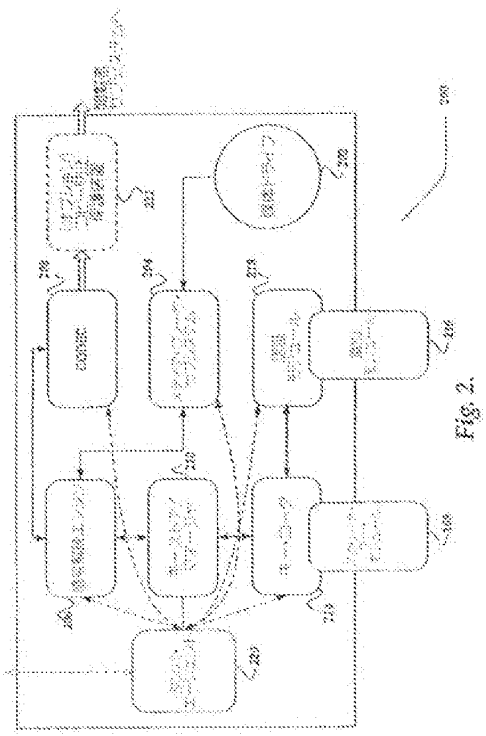


Fig. 2.

【図３】

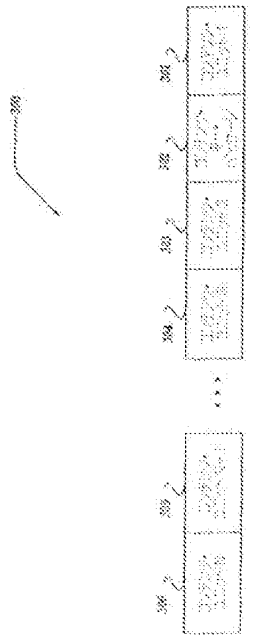


Fig. 3.

【図４】

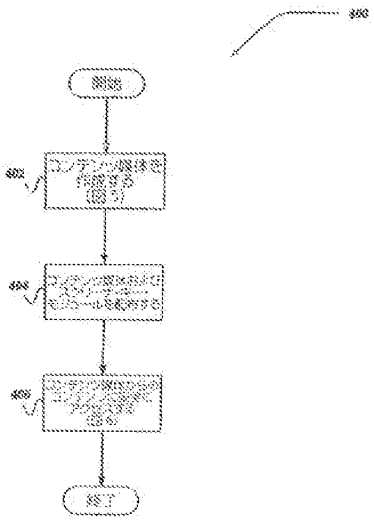


Fig. 4.

【図５】

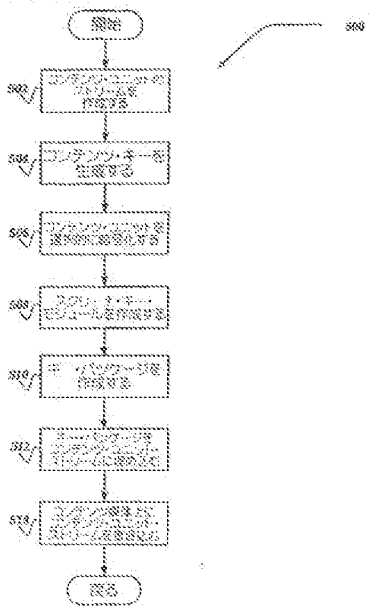


Fig. 5.

【図６Ａ】

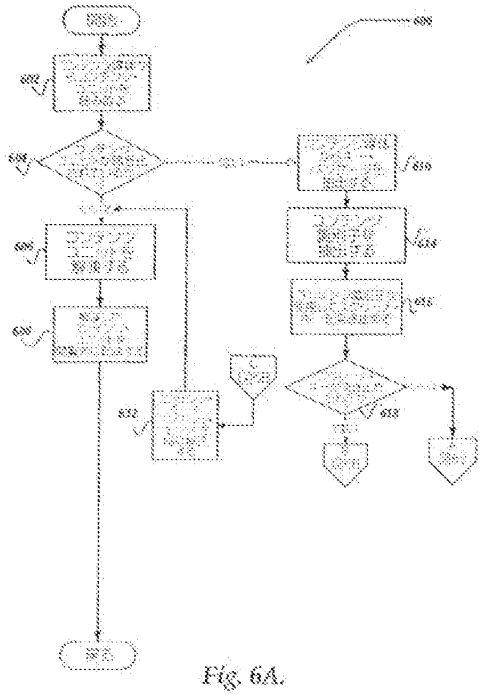


Fig. 6A.

【図６Ｂ】

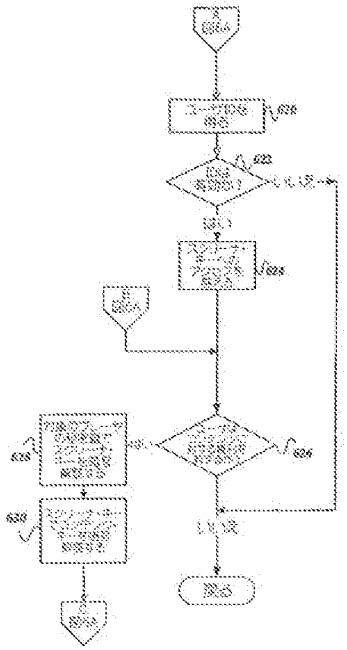


Fig. 6B.

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US04/40210

A CLASSIFICATION OF SUBJECT MATTER
IPC(7): : G06F 13/00; H04L 9/00, H04K 1/00
US CL : 713/193; 380/44; 281/282; 283/285
According to International Patent Classification (IPC) or to both national classification and IPC

B FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 713/193; 380/44; 281/282; 283/285

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
East, West, IEEE

C DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6,160,891 A (ALI-SALOJAN) 12 December 2000 (12.12.2000), abstract; fig.2-5; col.3, lines 17-67; col.4-6, col.7, lines 1-45.	1-16, 18-22, 24-29, 31-37 and 39
Y		17, 23, 30 and 38
Y	US 6,226,618 B1 (DOWNS et al.) 01 May 2001 (01.05.2001), see col.39, lines 50-54; col.64, lines 38-58; col.65.	17, 23, 30 and 38

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 December 2004 (21.12.2004)	07 JAN 2005
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (703)305-3230	Authorized officer Gilberto Barron Telephone No. 571-272-3799

Form PCT/ISA/210 (second sheet) (January 2004)

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

(74)代理人 100096611
弁理士 宮田 博行
(74)代理人 100084283
弁理士 秋元 芳雄
(72)発明者 モーテン、グレン・エイ
アメリカ合衆国ワシントン州９８００６，ベルヴュー，ハンドレッドセヴンティサード・アヴェニュー・サウス・イースト ６０３５

Fターム(参考) 5B017 AA07 BA05 BA07 CA16
5J104 EA17 EA19 FA05 FA07 FA14

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2008 年1 月24 日 (24.01.2008)

PCT

(10) 国際公開番号
WO 2008/010275 A1

(51) 国際特許分類:
H04L 9/08 (2006.01) *H04N 7/167* (2006.01)
G06F 21/24 (2006.01)

(21) 国際出願番号: PCT/JP2006/314296

(22) 国際出願日: 2006 年7 月19 日 (19.07.2006)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) [JP/JP]; 〒5718501 大阪府門真市大字門真１００６番地 Osaka (JP).

(72) 発明者; および
(75) 発明者/出願人 (米国についてのみ): 幡野 浩司 (HATANO, Kouji).

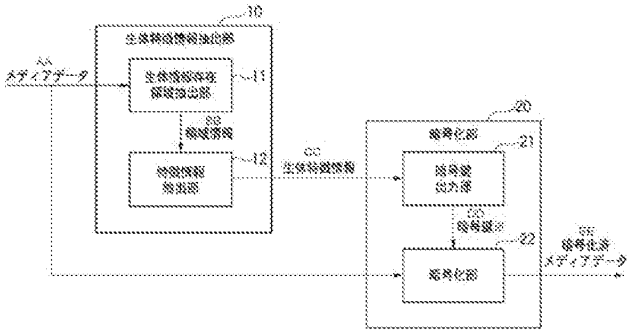
(74) 代理人: 市川 利光, 外(ICHIKAWA, Toshimitsu et al.); 〒1050003 東京都港区西新橋一丁目７番１３号 栄光特許事務所 Tokyo (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(続葉有)

(54) Title: MEDIUM DATA PROCESSING DEVICE AND MEDIUM DATA PROCESSING METHOD

(54) 発明の名称: メディアデータ処理装置及びメディアデータ処理方法



AA. MEDIUM DATA
10. BIOFEATURE INFORMATION EXTRACTING UNIT
11. BIOINFORMATION PRESENCE REGION EXTRACTING SECTION
BB. REGION INFORMATION
12. FEATURE INFORMATION EXTRACTING SECTION
CC. BIOFEATURE INFORMATION

20. ENCRYPTION UNIT
21. ENCRYPTION KEY OUTPUT SECTION
DD. ENCRYPTION KEY
22. ENCRYPTING SECTION
EE. ENCRYPTED MEDIUM DATA

(57) **Abstract:** Protection of privacy is realized when medium data including image information on a person is managed and an increase in management cost and degradation of operability are prevented. A medium data processing device for receiving medium data including image information at least on an image and creating encrypted medium data comprises biofeature extracting means (10) for extracting the feature value of bioinformation at least on a person included in the image information from the received medium data and encrypting means (20) for encrypting the medium data partially or wholly according to the extracted feature value. The feature parameter of the face of a person included in medium data is utilized.

(57) 要約: 人物の画像情報を含むメディアデータを管理する場合に、プライバシーの保護を実現すると共に、管理コストの増大や操作性の悪化を抑制する。 少なくとも１つの画像情報を含むメディアデータを入力し、暗号化されたメディアデータを生成するメディアデータ処理装置であって、入力される前記メディアデータからその画像情報に含まれている少なくとも一

(続葉有)

WO 2008/010275 A1

(84) 指定国（表示のない限り、全ての種類の広域保護が可能）：ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類：
— 国際調査報告書

２文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

人の人物に関する生体情報の特徴量を抽出する生体特徴抽出手段１０と、生体特徴抽出手段１０が抽出した特徴量に基づいて前記メディアデータを部分的にもしくは全体的に暗号化する暗号化処理手段２０とを設けた。メディアデータ中の人物の顔の特徴パラメータを利用する。

明 細 書

メディアデータ処理装置及びメディアデータ処理方法

技術分野

[0001] 本発明は、少なくとも1つの画像情報を含むメディアデータを入力し、暗号化されたメディアデータを生成するメディアデータ処理装置に関する。

背景技術

[0002] 最近では、例えばカメラ付き携帯電話端末や、デジタルカメラや、デジタルビデオカメラなどを用いることにより様々な被写体を撮影し、静止画や動画や音声情報などを含むメディアデータを簡単に生成することができる。また、得られたメディアデータを電子メールを利用して配布したり、インターネット上に存在するサーバを介して配布することも容易に行うことができる。

[0003] ところで、人物を含む被写体を撮影して得られたメディアデータの場合には、個人情報を含むのでプライバシーの問題が発生する。例えば、写真に写った特定の人物は、自分以外の他人にはその写真を見られたくないという願望を持つ場合も多い。従って、インターネット上や第三者に写真等を含むメディアデータが流出すると、特定の人物の写真が第三者に見られてしまうことになり、プライバシーが侵害される。

[0004] しかし、例えば撮影者から写真に写った特定の人物に対して写真を含むメディアデータを正規な手続きで転送する場合であっても、データ転送にインターネット上のサーバや電子メールを利用する場合には、メディアデータがインターネットを介して第三者に流出する可能性がある。従って、このようなメディアデータをインターネットを介して転送するような場合には、メディアデータを暗号化してから転送し、データが第三者に流出した場合でも第三者が写真を閲覧できないようにするのが望ましい。

[0005] 例えば、特許文献1に開示された技術では、自動写真撮影装置において、撮影毎にクライアントIDを生成し、生成されたクライアントIDを撮影された人物(顧客)に渡すようにしている。この場合、撮影された人物は受け取ったクライアントIDを利用してサーバにアクセスし、該当する写真を閲覧することができるが、クライアントIDを知らない第三者は写真を閲覧できないので、プライバシーを保護できる。

[0006] また、特許文献2に開示された技術では、撮影して得られた画像中の人の顔に基づいて特定した特定の人物に対して自動的に画像をメールで送信することを提案している。

特許文献1:特開2004－54672号公報

特許文献2:特開2005－267146号公報

発明の開示

発明が解決しようとする課題

[0007] しかしながら、特許文献1の技術を採用する場合には、撮影毎に生成されるクライアントIDの情報を、サーバ側及び顧客側の双方でそれぞれ確実に管理しなければならない。このため、サーバ側では膨大な量のクライアントIDの情報を管理しなければならず、設備コストや管理コストの増大が避けられない。また、顧客側では受け取ったクライアントIDの情報を覚えておかないと写真の閲覧ができなくなるので、覚えるのが煩わしい。クライアントIDの情報を紙に書き取ったり、パソコン上などにメモしておくことも可能であるが、メモを紛失する可能性もあるので、正規の顧客が自分の写真を閲覧できなくなる事態も想定される。

[0008] また、メディアデータを電子メールを利用して転送するような場合に、仮にインターネット上から第三者にデータが流出しなかった場合であっても、正規の受信者の端末(例えばカメラ付き携帯電話端末)の画面を第三者がのぞき込む場合もあるし、正規の受信者が端末を紛失したり端末が盗難にあったような場合には、写真を第三者に見られるのは避けられない。

[0009] 本発明は、人物の画像情報を含むメディアデータを管理する場合に、プライバシーの保護を実現すると共に、管理コストの増大や操作性の悪化を抑制することが可能なメディアデータ処理装置及びメディアデータ処理方法を提供することを目的とする。

課題を解決するための手段

[0010] 1番目の発明(請求項1)は、少なくとも1つの画像情報を含むメディアデータを入力し、暗号化されたメディアデータを生成するメディアデータ処理装置であって、入力される前記メディアデータからその画像情報に含まれている少なくとも一人の人物に関

する生体情報の特徴量を抽出する生体特徴抽出手段と、前記生体特徴抽出手段が抽出した特徴量に基づいて前記メディアデータを部分的にもしくは全体的に暗号化する暗号化処理手段とを設けたことを特徴とする。

[0011] 1番目の発明では、前記メディアデータを暗号化する際には、前記メディアデータの画像情報に含まれている人物に関する生体情報の特徴量を利用して暗号化が実施される。そのため、暗号化されたメディアデータから元のメディアデータを復元する際には、前記メディアデータの画像に含まれている人物に関する生体情報の特徴量が必要になる。従って、前記メディアデータを閲覧すべき正規の人物であれば、自分自身の生体情報があるので暗号化されたメディアデータから元のメディアデータを復元するのは容易であるが、該当する生体情報を持たない第三者が元のメディアデータを復元するのは困難である。しかも、生体情報以外にはIDやパスワードのような特別な情報を必要としないので、利用者はIDやパスワードのような情報を覚えたりメモして保存する必要もなく、簡単な操作で確実に元のメディアデータを復元できる。

[0012] 2番目の発明(請求項2)は、1番目の発明において、前記生体特徴抽出手段が抽出した特徴量からそれに対応付けられた特定の人物に事前に割り当てられた公開鍵の情報を取得する個人情報取得手段を更に設け、前記暗号化処理手段は、事前に準備したもしくは生成した共通鍵の情報を用いて前記メディアデータを暗号化することにより第1の暗号化データを生成し、前記個人情報取得手段が取得した公開鍵の情報を用いて前記共通鍵の情報を暗号化することにより第2の暗号化データを生成し、前記第1の暗号化データと第2の暗号化データとの両方を含む暗号化データを出力することを特徴とする。

[0013] 2番目の発明では、第1の暗号化データは前記共通鍵の情報により暗号化されているので、前記共通鍵の情報を利用して復号することができる。また、第2の暗号化データは特定の人物の公開鍵の情報で暗号化されているので、メディアデータを閲覧しようとする人物が正規の利用者である場合には、自分の公開鍵に対応する秘密鍵(自分自身だけが所有している)を利用して前記第2の暗号化データを復号し、前記共通鍵の情報を取得することができる。従って、取得した前記共通鍵の情報を利用して第1の暗号化データを復号し、元のメディアデータを復元できる。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0407
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0014] 例えば、メディアデータの画像中に複数の人物が写っているような場合には、その画像を閲覧すべき正規の利用者が複数になる可能性がある。このような場合、情報量の大きいメディアデータを正規の利用者毎にそれぞれ異なる公開鍵で暗号化すると、膨大な情報量の暗号化データを蓄積しなければならない。従って、複数の利用者が共通に利用可能な形式でメディアデータを暗号化するために共通鍵を用いるのが望ましい。そこで、例えば画像を閲覧すべき正規の利用者のそれぞれについて、独立した公開鍵を用いて第2の暗号化データを生成し、第2の暗号化データと第1の暗号化データとを合成しておけば、生体情報によって正規の利用者であると確認された人物(一人又は複数)だけが、第2の暗号化データから共通鍵を取得し、第1の暗号化データを前記共通鍵で復号し、メディアデータを閲覧可能になる。

[0015] 3番目の発明(請求項3)は、2番目の発明において、前記個人情報取得手段は取得した公開鍵に対応する特定の人物に事前に割り当てられたメールアドレス情報を取得し、前記暗号化処理手段は、生成した暗号化データを前記個人情報取得手段が取得したメールアドレス情報を宛先として電子メールで送信することを特徴とする。

[0016] 3番目の発明では、メディアデータに個人情報が含まれている正規の利用者のみに対して、暗号化されたメディアデータを電子メールで自動的に送信することができる。

[0017] 4番目の発明(請求項4)は、1番目の発明において、前記暗号化処理手段は、事前に準備したもしくは生成した第1の共通鍵の情報を用いて前記メディアデータを暗号化することにより第1の暗号化データを生成し、前記生体特徴抽出手段が抽出した特徴量をパラメータとして算出されるハッシュ値を第2の共通鍵として生成し、前記第2の共通鍵の情報を用いて前記第1の共通鍵の情報を暗号化することにより第2の暗号化データを生成し、前記第1の暗号化データと第2の暗号化データとの両方を含む暗号化データを出力することを特徴とする。

[0018] 4番目の発明では、第1の暗号化データは前記第1の共通鍵の情報により暗号化されているので、前記第1の共通鍵の情報を利用して復号することができる。また、第2の暗号化データは正規の利用者である人物に関する生体情報の特徴量から生成された第2の共通鍵の情報を用いて暗号化されているので、メディアデータを閲覧しよ

うとする人物が正規の利用者である場合には、自分の生体情報から前記第2の共通鍵と同等の情報(多少の誤差はある)を再び生成できるので、生成した第2の共通鍵を利用して前記第2の暗号化データを復号し、前記第1の共通鍵の情報を取得することができる。従って、取得した前記第1の共通鍵の情報を利用して第1の暗号化データを復号し、元のメディアデータを復元できる。

[0019] 5番目の発明(請求項5)は、1番目の発明において、前記生体特徴抽出手段は、前記メディアデータの画像に含まれている人物の顔に相当する領域の情報から前記特徴量を抽出することを特徴とする。

[0020] 5番目の発明では、画像に含まれている人物の顔に相当する領域の情報から前記特徴量を抽出するので、様々な人物を区別することが可能である。例えば、顔の中の眉毛、目、鼻などの要素について、相対的な距離などを調べることにより、人物を区別するのに必要な特徴量を抽出できる。

[0021] 6番目の発明(請求項6)は、5番目の発明において、前記生体特徴抽出手段は、前記メディアデータの1つの画像に複数の人物が含まれている場合には、複数の人物のそれぞれについて独立した特徴量を抽出することを特徴とする。

[0022] 6番目の発明では、複数の人物のそれぞれについて独立した特徴量を抽出するので、1つの画像に複数の人物が含まれている場合であっても、複数の人物のそれぞれについて、正規の利用者か否かを区別可能な暗号化データを生成できる。

[0023] 7番目の発明(請求項7)は、5番目の発明において、前記暗号化処理手段は、前記メディアデータの画像の中で、特徴量を抽出した人物の少なくとも一部を含む領域に限定して暗号化を実施し、画像中の前記領域を暗号化されたデータで置換することを特徴とする。

[0024] 7番目の発明では、画像中の人物の一部あるいは全ての領域だけが暗号化されたデータで置換されるので、暗号化を最小限にとどめることができ、処理の所要時間を短縮することが可能になる。また、プライバシーと無関係な画像領域については、正規の利用者でなくても閲覧することが可能になる。

[0025] 8番目の発明(請求項8)は、7番目の発明において、暗号化された前記メディアデータを復号処理する復号処理手段を更に設け、前記復号処理手段は、撮影により得

られた最新画像に含まれる特徴量から共通鍵を生成し、前記共通鍵を利用して、前記メディアデータに含まれる暗号化された領域のデータを復元することを特徴とする。

[0026] 8番目の発明では、メディアデータを閲覧しようとする人物から、顔などの特徴量を例えばリアルタイムで取得し、その人物が正規の利用者であるか否かを区別する場合を想定している。すなわち、正規の利用者がメディアデータを閲覧しようとする場合には、自分自身の生体情報の特徴量に基づいて正しい共通鍵を生成できるので、暗号化されたメディアデータを復号し閲覧することが可能になる。

[0027] 9番目の発明(請求項9)は、8番目の発明において、前記メディアデータの画像に複数の人物が含まれている場合には、人物毎に暗号化された複数の領域が含まれる画像の中で、最新画像から抽出した特徴量に対応する1つの領域だけについて画像データを復元することを特徴とする。

[0028] 9番目の発明では、閲覧対象の画像の中に複数の人物が含まれている場合に、閲覧しようとする一人の正規の利用者についてのみ、該当する人物を暗号化した領域を復号し、閲覧することが可能になる。

[0029] 10番目の発明(請求項10)は、8番目の発明において、撮影により得られた最新画像に複数の人物が含まれている場合には、復元可能な領域についても画像データの復元を中止することを特徴とする。

[0030] 10番目の発明では、閲覧しようとする一人の正規の利用者の他に第三者が端末の画面等を同時にのぞき込んでいるような場合には、正規の利用者が存在する場合であっても画像データの復元を中止することができる。

[0031] 11番目の発明(請求項11)は、少なくとも1つの画像情報を含むメディアデータを入力し、暗号化されたメディアデータを生成するメディアデータ処理方法であって、入力される前記メディアデータからその画像情報に含まれている少なくとも一人の人物に関する生体情報の特徴量を抽出し、抽出された前記特徴量に基づいて前記メディアデータを部分的にもしくは全体的に暗号化することを特徴とする。

[0032] 11番目の発明では、1番目の発明の装置を利用する場合と同様の結果が得られる。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0410
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

発明の効果

[0033] 本発明によれば、人物の画像情報を含むメディアデータを管理する場合に、プライバシーの保護を実現すると共に、管理コストの増大や操作性の悪化を抑制することが可能になる。すなわち、閲覧等を許可すべき正規の利用者か否かを区別するために人物の生体情報を利用するので、確実にプライバシーを保護することができ、しかもIDやパスワードなどを管理する必要がないので、メディアデータを管理するためのコスト増大を抑制できるし、利用者はIDやパスワードなどを覚える必要がないので操作性も改善される。

図面の簡単な説明

[0034] [図1]第1の実施の形態におけるメディアデータ処理装置の基本的な構成を示すブロック図である。

[図2]第1の実施の形態におけるメディアデータ処理装置の具体的な構成例を示すブロック図である。

[図3]図2に示すメディアデータ処理装置の主要動作を示すフローチャートである。

[図4]図2に示す装置から送信された電子メールを受信する端末の構成例を示すブロック図である。

[図5]図4に示す端末の主要動作を示すフローチャートである。

[図6]図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図7]図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図8]図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図9]図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図10]第2の実施の形態におけるメディアデータ処理装置の具体的な構成例を示すブロック図である。

[図11]図10に示すメディアデータ処理装置の主要動作を示すフローチャートである。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0411
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[図12]図10に示すメディアデータ処理装置によって暗号化された画像データを表示する端末の構成例を示すブロック図である。

[図13]図12に示す端末の主要動作を示すフローチャートである。

[図14]図10に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図15]第3の実施の形態におけるメディアデータ処理装置の具体的な構成例を示すブロック図である。

[図16]図15に示すメディアデータ処理装置の主要動作を示すフローチャートである。

[図17]図15に示すメディアデータ処理装置によって暗号化された画像データを表示する端末の主要動作を示すフローチャートである。

[図18]図15に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図19]図15に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図20]図15に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[図21]図2に示すメディアデータ処理装置における処理の内容を示す模式図である。

[図22]図4に示す端末における処理の内容を示す模式図である。

符号の説明

[0035] 5、6 撮影部

10、15 生体特徴情報抽出部

11 生体情報存在領域抽出部

12 特徴情報抽出部

20, 20B, 20C 暗号化部

21、21B 暗号鍵出力部

22、22B, 22C 暗号化処理部

23 アドレス帳保持部

25 暗号化部

30 電子メール送信部

35 蓄積部

36 画像表示指示入力部

40 電子メール受信部

50、50B 復号部

51 復号処理部

52 秘密鍵保持部

60 メールアドレス保持部

65 表示部

発明を実施するための最良の形態

[0036] （第1の実施の形態）

本発明のメディアデータ処理装置及びメディアデータ処理方法に関する1つの実施の形態について、図1～図9、図21及び図22を参照しながら以下に説明する。

[0037] 図1は第1の実施の形態におけるメディアデータ処理装置の基本的な構成を示すブロック図である。図2は第1の実施の形態におけるメディアデータ処理装置の具体的な構成例を示すブロック図である。図3は図2に示すメディアデータ処理装置の主要動作を示すフローチャートである。図4は図2に示す装置から送信された電子メールを受信する端末の構成例を示すブロック図である。図5は図4に示す端末の主要動作を示すフローチャートである。

[0038] 図6は図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。図7は図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。図8は図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。図9は図2に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。図21は図2に示すメディアデータ処理装置における処理の内容を示す模式図である。図22は図4に示す端末における処理の内容を示す模式図である。

[0039] まず、メディアデータ処理装置の基本的な構成及び動作について図1を参照しなが

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0413
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ら説明する。図1に示すように、このメディアデータ処理装置は生体特徴情報抽出部10と暗号化部20とを備えている。また、生体特徴情報抽出部10には生体情報存在領域抽出部11及び特徴情報抽出部12が備わっており、暗号化部20には暗号鍵出力部21及び暗号化処理部22が備わっている。

[0040] なお、生体特徴情報抽出部10及び暗号化部20のそれぞれの機能については、専用のハードウェア(電子回路)で構成することもできるし、制御用のコンピュータが実行するプログラムにより実現することもできる。

[0041] 生体特徴情報抽出部10及び暗号化部20には所定のメディアデータが処理対象データとして入力される。現実のメディアデータとしては、例えば写真のような静止画像だけを含む画像データや、動画像と音声情報とを組み合わせたビデオデータなどが想定される。

[0042] 生体特徴情報抽出部10に備わっている生体情報存在領域抽出部11は、メディアデータの中から、生体情報の存在する領域を抽出する。例えば、画像中に含まれている人物の顔の領域や、音声データに含まれている人物の音声の領域などが生体情報存在領域抽出部11によって抽出される。

[0043] 特徴情報抽出部12は、生体情報存在領域抽出部11が抽出した領域のデータから、特定人物の特徴を表す生体特徴情報を抽出する。例えば、画像中に映っている人物の顔のデータから、眉毛、目、鼻などの要素に関する相対的な大きさや距離を算出することにより生体特徴情報を抽出することが可能である。

[0044] 一方、暗号化部20の暗号鍵出力部21は、特徴情報抽出部12が抽出した生体特徴情報に応じた暗号鍵を出力する。また、暗号化処理部22は処理対象のメディアデータを暗号鍵出力部21が出力する暗号鍵を用いて暗号化し、暗号化済み画像データを出力する。

[0045] 従って、このメディアデータ処理装置から出力される暗号化済み画像データを閲覧しようとする場合には、特定の個人を表す生体特徴情報が必要になるので、例えばメディアデータに含まれている画像に映っている人物本人であれば暗号化済み画像データを復号するのは容易であるが、第三者がこれを復号するのは極めて困難である。また、生体特徴情報に基づいて暗号化を行うので、ID情報やパスワードは不要

であり、画像に映っている人物本人のような正規の利用者であれば、簡単な操作で暗号化済み画像データを閲覧できる。

[0046] 次に、さらに具体的な構成例について説明する。図2に示すメディアデータ処理装置は、例えばカメラ付き携帯電話端末のような端末に本発明を適用する場合を想定している。また、図2の例ではメディアデータとしてカメラの撮影によって得られた写真の画像データを利用する場合を想定している。

[0047] 図2に示すメディアデータ処理装置は、撮影部5と、生体特徴情報抽出部10と、暗号化部20と、電子メール送信部30とを備えている。撮影部5は利用者の入力操作に従って人物などを含む被写体を撮影し静止画などの画像データを生成する。撮影部5の撮影により得られた画像データが生体特徴情報抽出部10及び暗号化部20に入力される。

[0048] 生体特徴情報抽出部10は、図1と同様に生体情報存在領域抽出部11及び特徴情報抽出部12を備えている。暗号化部20は、暗号鍵出力部21、暗号化処理部22の他にアドレス帳保持部23を備えている。

[0049] 図2に示す暗号鍵出力部21は、特徴情報抽出部12から出力される生体特徴情報に対応付けられた人物の個人情報をアドレス帳保持部23から取り出し、前記人物に割り当てられた公開鍵及びメールアドレスの情報を暗号化処理部22に与える。なお、アドレス帳保持部23の代わりに、個人情報をネットワークを経由して取得するように構成を変更しても良い。

[0050] また、図2に示す暗号化処理部22は、撮影部5から出力される画像データを、暗号鍵出力部21の出力する公開鍵及びメールアドレスを用いて暗号化し、暗号化済み画像データを出力する。電子メール送信部30は、暗号化処理部22が出力する暗号化済み画像データを電子メールとして特定の宛先に自動的に送信する。

[0051] 次に、図2に示すメディアデータ処理装置の具体的な処理の内容について、図3を参照しながら説明する。

撮影部5の出力する画像データDが生体情報存在領域抽出部11に入力されると、生体情報存在領域抽出部11はステップS11に進み、画像データDの中で生体情報を含むi番目の領域Riを抽出する。1つの画像データDに複数の人物が写っている場

合には、それぞれの人物に対応する領域を抽出する。

[0052] 例えば、図6に示す画像データDには5人の人物が写っているので、それぞれの人物の顔が存在する矩形領域R1、R2、R3、R4、R5を生体情報存在領域抽出部11が特定し抽出する。各領域を特定する情報としては、図6に示すように矩形の左上位置のXY座標及び右下位置のXY座標を用いればよい。

[0053] 特徴情報抽出部12は、生体情報存在領域抽出部11から領域情報が入力されると、ステップS12の処理に進み、画像データ中のi番目の領域Riに含まれている生体特徴情報Piを抽出する。

[0054] 例えば、図7に示すようにi番目の領域Riの画像に対応する人物の顔に含まれている眉毛の端点F1、F2と、目の端点F3、F4と、鼻の下端F5の位置座標をそれぞれ求め、(F1－F2間の距離)と(F3－F4間の距離)との比率αを求め、(F2－F5間の距離)と(F4－F5間の距離)との比率βを求め、前記比率α及び比率βをパラメータとする生体特徴情報P1を決定することができる。この生体特徴情報P1は特定の人物、すなわちi番目の領域Riに写っている人と他人とを区別するのに利用できる。

[0055] 暗号鍵出力部21は、特徴情報抽出部12から生体特徴情報Piが入力されるとステップS13の処理に進み、生体特徴情報Piに対応する人物に割り当てられた公開鍵Kpi及びメールアドレスAiをアドレス帳保持部23上で検索する。

[0056] アドレス帳保持部23上には、例えば図8に示すように多数の人物のそれぞれに対応付けた個人情報が予め登録されているので、特徴情報抽出部12が出力する生体特徴情報Piと対応する特定の人物の個人情報を取得することができる。実際には、アドレス帳保持部23に保持されている各人物の生体特徴情報と処理対象の画像データから特徴情報抽出部12によって抽出される生体特徴情報Piとの間には誤差が生じる可能性がある。従って、ステップS13では生体特徴情報Piとの差(距離)が所定値(生じうる誤差程度)以内の生体特徴情報をアドレス帳保持部23上で検索すればよい。

[0057] ステップS13で検索に成功した場合には、暗号鍵出力部21はステップS14の処理において、該当する人物に割り当てられた公開鍵Kpi及びメールアドレスAiの情報を暗号化処理部22に与える。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0416
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0058] 一方、暗号化処理部22においては、処理対象の画像データDが撮影部5から入力されると、ステップS15の処理に進む。ステップS15では、暗号化に用いるメディア鍵(共通鍵)Kmを生成する。実際には、例えば乱数発生処理により生成した疑似乱数を用いて図8に示すようなメディア鍵Kmを生成すればよい。このメディア鍵Kmは処理対象の画像毎に生成される鍵であり、画像に含まれる人物とは無関係な共通鍵である。

[0059] ステップS16では、暗号化処理部22はステップS15で生成したメディア鍵Kmを用いて画像データDを暗号化処理する。これにより、暗号化された画像データ(C(Km、D)で表す)が生成される。

[0060] また、特定の人物に割り当てられた公開鍵Kpi及びメールアドレスAiの情報が暗号鍵出力部21から入力されると、暗号化処理部22はステップS17の処理に進む。ステップS17では、ステップS15で生成したメディア鍵Kmを、暗号鍵出力部21から入力された公開鍵Kpiを用いて暗号化し、例えば図8に示すような暗号化されたメディア鍵(C(Kpi、Km)で表す)を生成する。

[0061] なお、1フレームの画像データの中に複数の人物が写っている場合には、人物の数だけステップS12, S13, S14, S17の処理を繰り返し実行する。従って、画像データの中に複数の人物が写っている場合には、暗号化されたメディア鍵C(Kpi、Km)が人数分だけ順次に生成される。

[0062] ステップS18では、ステップS17で生成された暗号化されたメディア鍵C(Kpi、Km)と、該当する人物のメールアドレスAiと、ステップS16で生成した暗号化された画像データC(Km, D)とを結合することにより、暗号化済み画像データを生成する。ステップS18で生成される暗号化済み画像データの具体例が図9に示されている。

[0063] 図9に示す暗号化済み画像データは、ヘッダ部と、鍵情報部と、暗号化データ部とで構成されている。暗号化済み画像データのヘッダ部には、鍵情報部のサイズを表す情報が格納され、鍵情報部には複数エントリ(複数の人物に相当)のそれぞれについて、サイズと、メールアドレスAiと、暗号化されたメディア鍵C(Kpi、Km)とが格納され、暗号化データ部にはステップS16で生成した暗号化された画像データC(Km, D)が格納される。

[0064] このような暗号化済み画像データが電子メール送信部30に入力されるので、電子メール送信部30は入力される暗号化済み画像データの内容から送信先のメールアドレスAiを特定し、この暗号化済み画像データを電子メールで自動的に送信することができる。つまり、暗号化処理した画像データの中に写っている人物に割り当てられたメールアドレスを宛先とする電子メールが自動的に送信される。

[0065] 次に、電子メール送信部30が送信した電子メールを受信する受信側の端末の構成及び動作について説明する。受信側の端末の構成例が図4に示されている。受信側の端末についても、携帯電話端末などを想定することができる。図4を参照すると、この端末には電子メール受信部40と、復号部50と、メールアドレス保持部60と、表示部65とが設けてあり、復号部50には復号処理部51と秘密鍵保持部52とが備わっている。

[0066] 送信側の端末から送信された電子メールは、所定の通信ネットワークを経由して受信側の端末の電子メール受信部40に入力される。電子メール受信部40は、受信した電子メールに含まれている暗号化済み画像データ(図9参照)を出力する。

[0067] メールアドレス保持部60は、受信者に事前に割り当てられたメールアドレスのデータを保持している。また、秘密鍵保持部52は受信者が事前に生成した秘密鍵Ksiを保持している。この秘密鍵Ksiは受信者だけが所有している。また、秘密鍵Ksiとペアになる公開鍵Kpiが秘密鍵Ksiと同時に生成されており、公開鍵Kpiは様々な人に事前に配布されている。従って、受信者の公開鍵Kpiは送信側の端末に設けられた前述のアドレス帳保持部23にも存在している。

[0068] 復号部50の復号処理部51は、電子メール受信部40から入力される暗号化済み画像データを、受信者のメールアドレスと秘密鍵Ksiとを用いて復号し、暗号化前の画像データDを復元する。表示部65は、復号部50によって復号された画像データを、例えば図6に示すような画像として画面に表示する。

[0069] 次に、復号処理部51における処理の具体的な内容について、図5を参照しながら説明する。

[0070] ステップS21では、メールアドレス保持部60から入力した受信者本人のメールアドレスAiを用いて、暗号化済み画像データ中の鍵情報部から、受信者に対応付けられ

た暗号化されたメディア鍵C(Kpi、Km)を検索する。すなわち、受信者本人のメールアドレスAiと一致するメールアドレスを探し、該当するメールアドレスと並んで配置されている暗号化されたメディア鍵C(Kpi、Km)を暗号化済み画像データの中から探す。

[0071] 目的の暗号化されたメディア鍵C(Kpi、Km)が見つかった場合には、ステップS22からステップS23に進み、見つからない場合はステップS26に進む。

[0072] ステップS23では、ステップS21で暗号化済み画像データから抽出した暗号化されたメディア鍵C(Kpi、Km)を、秘密鍵保持部52から取り出した受信者本人の秘密鍵Ksiを用いて復号処理する。これにより、メディア鍵Kmが得られる。

[0073] ステップS24では、暗号化済み画像データに含まれている暗号化データ部の内容、すなわち暗号化された画像データC(Km、D)を取り出し、ステップS23で取得したメディア鍵Kmを用いて(C(Km、D))を復号処理する。これにより、暗号化前の画像データDが復元される。

[0074] ステップS25では、ステップS24で復元された画像データDを表示部65に出力する。

[0075] ステップS26では、復号不能であることを表すメッセージ等を表示部65で表示するように指示する。

[0076] 理解を容易にするため、上述のメディアデータ処理装置に関する処理の流れについて説明を補足する。撮影装置側、すなわち図2に示したメディアデータ処理装置においては図21に示すような処理が実施される。

(1)カメラを用いて人物を含む画像を撮影する。

(2)処理対象の画像から各人物の顔領域を抽出する。

(3)各人物の顔領域の特徴パラメータを事前に用意したアドレス帳上のデータと照合することにより撮影された人物を特定する。

(4)画像データを暗号化するための暗号鍵、すなわちメディア鍵Kmを生成する。

(5)暗号鍵で画像データを暗号化する。

(6)暗号鍵自体が第三者に流出しないように、撮影された人物の公開鍵を用いて暗号鍵を暗号化する。

(7)暗号化された画像データと暗号化された暗号鍵とを連結して送信用のデータを生成する。

(8)生成した暗号化データを電子メール等で配布する。

[0077] 一方、閲覧装置側、すなわち図4に示した受信側の端末においては、図22に示すような処理が実施される。

(1)受信した暗号化済み画像データから受信者本人のメールアドレスに対応する鍵を抽出する。

(2)抽出した鍵は暗号化されているので、受信者本人の秘密鍵Ksiを用いて復号し、暗号鍵、すなわちメディア鍵Kmを復元する。

(3)得られたメディア鍵Kmを用いて画像データを復号する。

(4)復号した画像データを表示して閲覧する。

従って、画像データを閲覧できるのは、受信者が画像データの中に写っている場合だけに限られる。また、送信側では画像データの中に写っている人物の生体情報の特徴に基づいて暗号化を実施するので、送信側及び受信側のそれぞれにおいて、IDやパスワードのような特別な情報を管理する必要がない。

[0078] (第2の実施の形態)

本発明のメディアデータ処理装置及びメディアデータ処理方法に関するもう1つの実施の形態について, 図10～図14を参照しながら以下に説明する。

[0079] 図10は第2の実施の形態におけるメディアデータ処理装置の具体的な構成例を示すブロック図である。図11は図10に示すメディアデータ処理装置の主要動作を示すフローチャートである。図12は図10に示すメディアデータ処理装置によって暗号化された画像データを表示する端末の構成例を示すブロック図である。図13は図12に示す端末の主要動作を示すフローチャートである。図14は図10に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[0080] この形態は第1の実施の形態の変形例である。また, 図10～図14において第1の実施の形態と対応する要素は同一の符号や番号を付けて示してある。以下の説明においては、第1の実施の形態と異なる構成要素や動作を主体として説明を行う。

[0081] 図10に示すメディアデータ処理装置は, 第1の実施の形態と同様に、カメラ付き携

帯電話端末のような端末に本発明を適用する場合を想定しているが、電子メールのようなデータ送信機能は必要としない。

[0082] 図10に示すメディアデータ処理装置は、撮影部5と、生体特徴情報抽出部10と、暗号化部20Bと、蓄積部35とを備えている。蓄積部35としては、例えばSDカードのように脱着が可能で持ち運びが可能な記憶媒体を想定している。また、撮影部5及び生体特徴情報抽出部10の構成や動作については第1の実施の形態と同様である。

[0083] 暗号化部20Bは、暗号鍵出力部21B及び暗号化処理部22Bを備えている。暗号鍵出力部21Bは、特徴情報抽出部12から入力される生体特徴情報に基づいて共通鍵を生成し、暗号化処理部22Bは、撮影部5の出力する画像データを暗号化すると共に、暗号鍵出力部21Bの生成した共通鍵を利用して暗号化を実施する。具体的な処理の内容については図11に示されている。

[0084] 図11において、各ステップS11、S12、S15、S16については第1の実施の形態の場合と同様である。

[0085] ステップS13Bでは、暗号鍵出力部21Bは特徴情報抽出部12から入力される生体特徴情報Piに一方向性関数をかけることにより、生体特徴情報Piのハッシュ値h(Pi)を求める。そしてハッシュ値h(Pi)を共通鍵Kciとして出力する(図14参照)。

[0086] ステップS17Bでは、暗号化処理部22Bはステップ S15で生成したメディア鍵(第2の共通鍵)Kmを、暗号鍵出力部21Bから入力された共通鍵Kciを用いて暗号化する。暗号化されたメディア鍵Kmを(C(Kci、Km))として表してある(図14参照)。

[0087] ステップS18Bでは、ステップS17Bで得られた(C(Kci、Km))とステップS16で得られた暗号化画像データ(C(Km, D))とを結合することにより、暗号化済み画像データを生成する。この暗号化済み画像データが蓄積部35に保存される。

[0088] 図11の処理によって生成される暗号化済み画像データの具体例が図14に示されている。図14に示す暗号化済み画像データは、ヘッダ部と、鍵情報部と、暗号化データ部とで構成されている。暗号化済み画像データのヘッダ部には、鍵情報部のサイズを表す情報が格納され、鍵情報部には複数の人物のそれぞれについて、暗号化されたメディア鍵C(Kpi、Km)が格納され、暗号化データ部には暗号化された画像データC(Km, D)が格納される。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0421
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0089] 蓄積部35を端末から取り外して持ち運ぶことにより、複数の利用者や複数の端末の間で暗号化済み画像データの受け渡しを行うことができる。受け取った暗号化済み画像データを閲覧しようとする場合に用いる端末の構成例が図12に示されている。図12に示す端末としては、例えばカメラ付き携帯電話端末を利用することが考えられる。

[0090] 図12に示す端末は、撮影部6と、生体特徴情報抽出部15と、暗号化部25と、蓄積部35と、画像表示指示入力部36と、復号部50Bと、表示部65とを備えている。

[0091] 撮影部6は、例えばカメラ付き携帯電話端末に備わっているカメラのように、端末の利用者自身の顔の画像を撮影することが可能な撮像装置である。

[0092] 生体特徴情報抽出部15は、図10に示したメディアデータ処理装置の生体特徴情報抽出部10と同様に、生体情報存在領域抽出部11及び特徴情報抽出部12を備えている。生体特徴情報抽出部15は、利用者の顔を含む画像データを撮影部6から入力して、前述の生体特徴情報抽出部10と同様の処理を実施する。

[0093] 暗号化部25は、暗号鍵出力部21Bを内蔵している。暗号鍵出力部21Bの動作は前述の暗号鍵出力部21の動作と同等である。

[0094] 蓄積部35は、例えばSDカードのように端末から取り外して持ち運ぶことが可能な記憶媒体であり、図10に示したメディアデータ処理装置によって生成された暗号化済み画像データを保持している。

[0095] 画像表示指示入力部36は、利用者が操作可能なキースイッチのような入力装置であり、利用者からの指示を入力するために利用される。復号部50Bは、蓄積部35に保持されている暗号化済み画像データを入力し、暗号化部25が生成する共通鍵を利用して復号処理し、暗号化前の画像データを復元する。表示部65は、復号部50Bが復元した画像データを画面上に表示する。

[0096] 図12に示す端末の主要な処理の内容が図13に示されている。図13に示す処理について以下に説明する。

ステップS31で画像表示指示入力部36を利用者が操作すると、画像表示指示入力部36は画像表示指示を蓄積部35に与える。ステップS32で画像表示指示が入力されると、蓄積部35はそれ自身が保持している暗号化済み画像データを復号部50

Bに出力する。

[0097] 一方、画像表示指示入力部36から出力される画像表示指示に従って、撮影部6は最新の画像(利用者の顔が被写体として含まれる)を撮影し、生体特徴情報抽出部15は最新の画像に基づいて生体特徴情報を生成する。そして、暗号化部25内の暗号鍵出力部21Bは、生体特徴情報抽出部15から出力される生体特徴情報Pjに基づいて共通鍵Kcjを生成する。

[0098] すなわち、図11に示すステップS13Bと同様に、生体特徴情報Pjに一方向性関数をかけることにより、生体特徴情報Pjのハッシュ値h(Pj)を求める。そしてハッシュ値h(Pj)を共通鍵Kcjとして出力する。つまり、蓄積部35に保持されている暗号化された画像を閲覧しようとしている利用者の顔の特徴パラメータに基づいて共通鍵Kcjを生成する。

[0099] 復号部50Bは、ステップS34で暗号鍵出力部21Bから共通鍵Kcjが入力されると、ステップS35の処理に進む。ステップS35では、復号部50Bは暗号化済み画像データから暗号化されたメディア鍵(C(Kci、Km))を取り出し、(C(Kci、Km))を共通鍵Kcjを用いて復号処理する。これによりメディア鍵Km'が得られる。

[0100] すなわち、閲覧対象の画像に映っている人物と、その画像を閲覧しようとしている利用者とが同一人物である場合には、共通鍵Kciを生成する際に用いた生体情報の特徴パラメータと、共通鍵Kcjを生成する際に用いた生体情報の特徴パラメータとが実質的に同一であるため、共通鍵Kciと共通鍵Kcjとは同じ値になる。従って、共通鍵Kcjを用いて(C(Kci、Km))を復号処理することにより得られたメディア鍵Km'は、共通鍵Kciを用いて復号処理することにより得られるメディア鍵Kmと同等の値になる。

[0101] ただし、特徴情報抽出部12が抽出する生体情報の特徴パラメータに誤差が含まれると

、同一人物の生体情報から生成される共通鍵Kciと共通鍵Kcjとが同じ値にならないことになるので、暗号鍵出力部21Bのステップ S13Bの処理においては、与えられた特徴パラメータを有効桁数内に丸めるなど特徴パラメータに含まれる誤差を除去した後に一方向性関数をかけて共通鍵Kciおよび共通鍵Kcjを生成するようにすることが

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0423
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

望ましい。

[0102] 次のステップS36では、復号部50Bは暗号化済み画像データから暗号化画像データ(C(Km、D))を取り出し、ステップS35で得られたメディア鍵Km'を用いて復号処理を実施し、暗号化前の画像データD'を生成する。この画像データD'は撮影された画像データDと同等である。

[0103] 図13に示す例では、ステップS36で復元された画像データD'が正しく復号されているか否かを調べるために、次のステップS37で画像データD'のファイルの先頭領域に予め定めたデータ(ファイル形式固有の文字列など)が存在するか否かを確認し、存在することを確認した場合に次のステップS38に進み、画像データD'を画像として表示する。なお、ステップS37の処理については省略しても良い。

[0104] 従って、画像データを閲覧できるのは、閲覧用の端末の撮影部6で撮影される利用者が、閲覧対象の画像データの中に写っている人物と一致する場合だけに限られる。また、画像データを暗号化する際に、その画像の中に写っている人物の生体情報の特徴に基づいて暗号化を実施するので、画像を暗号化する端末と画像を閲覧する端末とのそれぞれにおいて、IDやパスワードのような特別な情報を管理する必要がない。また、事前に利用者の公開鍵を準備する必要もない。

[0105] (第3の実施の形態)

本発明のメディアデータ処理装置及びメディアデータ処理方法に関するもう1つの実施の形態について、図15～図20を参照しながら以下に説明する。

[0106] 図15は第3の実施の形態におけるメディアデータ処理装置の具体的な構成例を示すブロック図である。図16は図15に示すメディアデータ処理装置の主要動作を示すフローチャートである。図17は図15に示すメディアデータ処理装置によって暗号化された画像データを表示する端末の主要動作を示すフローチャートである。図18は図15に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。図19は図15に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。図20は図15に示すメディアデータ処理装置に関するデータの具体例を示す模式図である。

[0107] この形態は第2の実施の形態の変形例である。また、図15～図20において第2の

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0424
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

実施の形態と対応する要素は同一の符号や番号を付けて示してある。以下の説明においては、第2の実施の形態と異なる構成要素や動作を主体として説明を行う。

[0108] 図15に示すメディアデータ処理装置は、第2の実施の形態と同様に、カメラ付き携帯電話端末のような端末に本発明を適用する場合を想定している。

[0109] 図10に示すメディアデータ処理装置は、撮影部5と、生体特徴情報抽出部10と、暗号化部20Cと、蓄積部35とを備えている。蓄積部35としては、例えばSDカードのように脱着が可能で持ち運びが可能な記憶媒体を想定している。また、撮影部5及び生体特徴情報抽出部10の構成や動作については第1の実施の形態と同様である。

[0110] 暗号化部20Cは、暗号鍵出力部21B及び暗号化処理部22Cを備えている。暗号鍵出力部21Bは、特徴情報抽出部12から入力される生体特徴情報に基づいて共通鍵を生成し、暗号化処理部22Cは、撮影部5の出力する画像データを部分的に暗号化すると共に、暗号鍵出力部21Bの生成した共通鍵を利用して暗号化を実施する。具体的な処理の内容については図16に示されている。

[0111] 第2の実施の形態と同様に、ステップS11では処理対象の画像データの中で生体情報を含む領域Riを検出する。例えば、人物毎に顔の領域Riを検出する。また、ステップS12では、ステップS11で検出された領域Ri毎に、生体特徴情報Piを抽出する。

[0112] ステップS13Bでは、暗号鍵出力部21Bは特徴情報抽出部12から入力される生体特徴情報Piに一方向性関数をかけることにより、生体特徴情報Piのハッシュ値h(Pi)を求める。そしてハッシュ値h(Pi)を共通鍵Kciとして出力する(図18参照)。

[0113] 生体情報存在領域抽出部11がステップS11で検出した領域Riの情報(矩形の位置と大きさを表す座標情報)は、暗号化処理部22Cにも入力される。暗号化処理部22Cは、生体情報存在領域抽出部11から領域Riの情報が入力されるとステップS15Bの処理に進む。

[0114] ステップS15Bでは、検出された各領域Riを暗号化するためのメディア鍵(共通鍵)Kmiを検出された領域Riの数だけ生成する。

[0115] ステップS41では、処理対象の画像データの中でステップS11で検出されたi番目の領域Riに相当する部分画像データを、ステップSS15Bで生成されたi番目のメディ

ア鍵Kmiを用いて暗号化処理する。処理の結果をi番目の暗号化部分画像データC(Kmi、Ri)とする。なお、(Di－1)は処理前の画像データ全体を表す。

[0116] ステップS42では、処理対象の画像データの中でステップS11で検出されたi番目の領域Riに相当する部分画像データを、ステップS41で生成された暗号化部分画像データC(Kmi、Ri)で置換し、その結果を処理後画像データDiとして出力する。

[0117] つまり、処理対象の1フレームの画像データの中で、各人物の顔に相当する領域Riのデータだけをメディア鍵Kmiを用いて暗号化し、暗号化前の1フレームの画像データの中で各領域Riだけが部分的に暗号化された処理後画像データDiを生成する。

[0118] 次のステップS43では、ステップS15Bで生成された各メディア鍵Kmiを、暗号鍵出力部21Bから入力されるi番目の共通鍵Kciを用いて暗号化し、暗号化されたメディア鍵C(Kci、Kmi)を生成する。

[0119] 処理対象の1フレームの画像データの中に複数の人物が写っている場合には、各ステップS12、S13B、S15B、S41、S42、S43は人物の数だけ繰り返し実行される。

[0120] ステップS44では、ステップS43で生成された暗号化されたメディア鍵C(Kci、Kmi)と、ステップS11で検出された領域Riの情報と、ステップS41で生成された各領域の暗号化部分画像データC(Kmi、Ri)とを結合した暗号化済み画像データを生成する。この暗号化済み画像データが蓄積部35に保存される。

[0121] 暗号化済み画像データの具体的な構成例が図18に示されている。図18を参照すると、この暗号化済み画像データはヘッダ部と、鍵情報部と、暗号化データ部とで構成されている。

[0122] ヘッダ部は、鍵情報部のサイズを表す情報を格納する。鍵情報部は、1フレームの画像データに含まれている人物の数に相当する複数の記憶領域を備えており、各記憶領域はi番目の領域Riの座標情報と、領域Riに対応付けられた暗号化されたメディア鍵C(Kci、Kmi)とをそれぞれ格納する。暗号化データ部は、1フレームの画像データの中で各領域Riのデータだけが部分的に暗号化された処理後画像データDnを格納する。

[0123] 処理後画像データDnをそのまま再生すると、例えば図19に示すような画像が表示される。すなわち、画像に写っている各人物の顔以外は正常に見ることができるが、各人物の顔の領域はそれぞれ暗号化されているので、例えばモザイクがかかっている場合と同様に見ることができない。

[0124] 第2の実施の形態と同様に、蓄積部35を端末から取り外して持ち運ぶことにより、複数の利用者や複数の端末の間で暗号化済み画像データの受け渡しを行うことができる。受け取った暗号化済み画像データを閲覧しようとする場合には、前述の図12に示されたような構成の端末を用いればよい。具体的な端末としては、例えばカメラ付き携帯電話端末を利用することが考えられる。但し、暗号化済み画像データを再生するために端末が実行する処理の内容は図17に示すように変更されている。図17に示す処理について以下に説明する。

[0125] ステップS31で画像表示指示入力部36を利用者が操作すると、画像表示指示入力部36は画像表示指示を蓄積部35に与える。ステップS32で画像表示指示が入力されると、蓄積部35はそれ自身が保持している暗号化済み画像データを復号部50Bに出力する。

[0126] 一方、画像表示指示入力部36から出力される画像表示指示に従って、撮影部6は最新の画像(利用者の顔が被写体として含まれる)を撮影し、生体特徴情報抽出部15は最新の画像に基づいて生体特徴情報を生成する。そして、暗号化部25内の暗号鍵出力部21Bは、生体特徴情報抽出部15から出力される生体特徴情報Pjに基づいて共通鍵Kcjを生成する。

[0127] すなわち、図16に示すステップS13Bと同様に、生体特徴情報Pjに一方向性関数をかけることにより、生体特徴情報Pjのハッシュ値h(Pj)を求める。そしてハッシュ値h(Pj)を共通鍵Kcjとして出力する。つまり、蓄積部35に保持されている暗号化された画像を閲覧しようとしている利用者の顔の特徴パラメータに基づいて共通鍵Kcjを生成する。

[0128] 復号部50Bは、暗号化済み画像データに含まれている暗号化データ部のデータDnを画像データD'としてステップS51で抽出する。

[0129] また、暗号鍵出力部21Bから共通鍵Kcjが入力されると、復号部50Bはステップ S5

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0427
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

2の処理に進む。ステップS52では、復号部50Bは暗号化済み画像データから暗号化されたそれぞれのメディア鍵(C(Kci、Km))を取り出し、(C(Kci、Km))を共通鍵Kcjを用いて復号処理する。これによりi番目のメディア鍵Kmi'が得られる。

[0130] 次のステップS53では、復号部50Bはステップ S51で抽出された画像データD'のi番目の領域Riについて、ステップS52で得られたメディア鍵Kmi'を用いて復号処理を実施し、暗号化前の画像データD'を生成する。

[0131] 但し、暗号化された画像を閲覧しようとしている利用者が一人だけの場合には、暗号鍵出力部21Bから出力される共通鍵Kcjも1つだけであるので、ステップS52では1つのメディア鍵Kmi'だけしか正しく復号できない。そのため、ステップS53で正しく復号される部分画像データも1つの領域に対応する画像データだけである。

[0132] 従って、例えば図19に示すように5人の人物が写っている画像を暗号化した暗号化済み画像データを閲覧する場合に、その画像データに写っている一人の人物が利用者としてこの画像を閲覧しようとすると、例えば図20に示すように、利用者自身に対応する部分画像(顔の領域)だけが正しく復号されて表示され、他の4人の顔の情報は復号できないため見ることができない。

[0133] なお、第3の実施の形態では画像を閲覧しようとする人と一致する人物の画像だけを復号して閲覧可能にする場合を想定しているが、閲覧対象の画像に含まれている複数の人物のいずれか一人と閲覧する利用者とが一致する場合には、全ての人物の画像を復号して閲覧できるようにすることも可能である。

[0134] また、第3の実施の形態では画像を閲覧しようとする人が利用者一人だけの場合を想定しているが、現実には利用者の側方から第三者が端末の画面と利用者と共にのぞき込む場合も想定される。従って、例えば閲覧対象の画像に含まれている複数の人物のいずれとも生体情報の特徴パラメータが一致しない人物が撮影部6で撮影した画像から検出されている時には、正規の利用者が画像を閲覧しようとしている場合であっても、画像の復号、あるいは表示を中止するように制御するのが望ましい。

[0135] また、第3の実施の形態では暗号化処理部が生体情報が存在する領域全体を暗号化するようにしているが、たとえば目の部分だけを暗号化するなど、生体情報が含まれる領域の一部のみに暗号化処理を施すようにしてもよい。生体情報が含まれる

領域は、特徴情報抽出部が抽出した特徴点の情報に基づいて特定することができる。たとえば、特徴情報抽出部12が抽出した右眉の特徴点（たとえば図7のF1、F2）、右目の特徴点（同図7のF3、F4）、左眉の特徴点（例示せず）、左目の特徴点（例示せず）の各座標値を取り囲む領域Ri'は人物の目の付近の領域を表すので、暗号化処理部22Cが図16のステップS41～ステップS44において領域Riの替わりに領域Ri'に対して暗号化処理を行うことにより、人物の目の部分だけを暗号化した暗号化済メディアデータを得ることができる。

[0136] また、前述の各実施の形態では、処理対象の画像から生体情報の特徴パラメータを抽出する場合を想定しているが、例えばメディアデータとして画像の他に音声データが含まれている場合には、音声データから各人物の声紋を調べて生体情報の特徴パラメータを抽出することもできる。

産業上の利用可能性

[0137] 以上のように、人物を撮影して得られた画像データを含むメディアデータを、例えばメモリカードのような記憶媒体、電子メール、ネットワーク上のサーバなどを利用して受け渡すような場合に本発明を適用することにより、個人情報である人物の画像が第三者に見られるのを防止することができる。しかも、生体情報の特徴パラメータに基づいて暗号化を実施するので、利用者の管理のためにID情報やパスワードを使用する必要がない。従って、暗号化された情報を提供する側においては、低コストで利用者の管理を行うことができ、暗号化された情報を閲覧しようとする利用者においては、ID情報やパスワードなどを覚える必要がないので操作性が良く、自分自身が写っている画像であれば確実に閲覧できる。

請求の範囲

[1] 少なくとも1つの画像情報を含むメディアデータから暗号化されたメディアデータを生成するメディアデータ処理装置であって、

入力される前記メディアデータからその画像情報に含まれている少なくとも一人の人物に関する生体情報の特徴量を抽出する生体特徴抽出手段と、

前記生体特徴抽出手段が抽出した特徴量に基づいて前記メディアデータを部分的にもしくは全体的に暗号化する暗号化処理手段と

を設けたことを特徴とするメディアデータ処理装置。

[2] 請求項1に記載のメディアデータ処理装置において、

前記生体特徴抽出手段が抽出した特徴量からそれに対応付けられた特定の人物に事前に割り当てられた公開鍵の情報を取得する個人情報取得手段を更に設け、

前記暗号化処理手段は、事前に準備したもしくは生成した共通鍵の情報を用いて前記メディアデータを暗号化することにより第1の暗号化データを生成し、前記個人情報取得手段が取得した公開鍵の情報を用いて前記共通鍵の情報を暗号化することにより第2の暗号化データを生成し、前記第1の暗号化データと第2の暗号化データとの両方を含む暗号化データを出力する

ことを特徴とするメディアデータ処理装置。

[3] 請求項2に記載のメディアデータ処理装置において、

前記個人情報取得手段は取得した公開鍵に対応する特定の人物に事前に割り当てられたメールアドレス情報を取得し、

前記暗号化処理手段は、生成した暗号化データを前記個人情報取得手段が取得したメールアドレス情報を宛先として電子メールで送信する

ことを特徴とするメディアデータ処理装置。

[4] 請求項1に記載のメディアデータ処理装置において、

前記暗号化処理手段は、事前に準備したもしくは生成した第1の共通鍵の情報を用いて前記メディアデータを暗号化することにより第1の暗号化データを生成し、前記生体特徴抽出手段が抽出した特徴量をパラメータとして算出されるハッシュ値を第2の共通鍵として生成し、前記第2の共通鍵の情報を用いて前記第1の共通鍵の情

報を暗号化することにより第2の暗号化データを生成し、前記第1の暗号化データと第2の暗号化データとの両方を含む暗号化データを出力する

ことを特徴とするメディアデータ処理装置。

[5] 請求項1に記載のメディアデータ処理装置において、前記生体特徴抽出手段は、前記メディアデータの画像に含まれている人物の顔に相当する領域の情報から前記特徴量を抽出することを特徴とするメディアデータ処理装置。

[6] 請求項5に記載のメディアデータ処理装置において、前記生体特徴抽出手段は、前記メディアデータの1つの画像に複数の人物が含まれている場合には、複数の人物のそれぞれについて独立した特徴量を抽出することを特徴とするメディアデータ処理装置。

[7] 請求項5に記載のメディアデータ処理装置において、

前記暗号化処理手段は、前記メディアデータの画像の中で、特徴量を抽出した人物の少なくとも一部を含む領域に限定して暗号化を実施し、画像中の前記領域を暗号化されたデータで置換する

ことを特徴とするメディアデータ処理装置。

[8] 請求項7に記載のメディアデータ処理装置において、

暗号化された前記メディアデータを復号処理する復号処理手段を更に設け、

前記復号処理手段は、撮影により得られた最新画像に含まれる特徴量から共通鍵を生成し、前記共通鍵を利用して、前記メディアデータに含まれる暗号化された領域のデータを復元する

ことを特徴とするメディアデータ処理装置。

[9] 請求項8に記載のメディアデータ処理装置において、前記メディアデータの画像に複数の人物が含まれている場合には、人物毎に暗号化された複数の領域が含まれる画像の中で、最新画像から抽出した特徴量に対応する1つの領域だけについて画像データを復元することを特徴とするメディアデータ処理装置。

[10] 請求項8に記載のメディアデータ処理装置において、撮影により得られた最新画像に複数の人物が含まれている場合には、復元可能な領域についても画像データの復元を中止することを特徴とするメディアデータ処理装置。

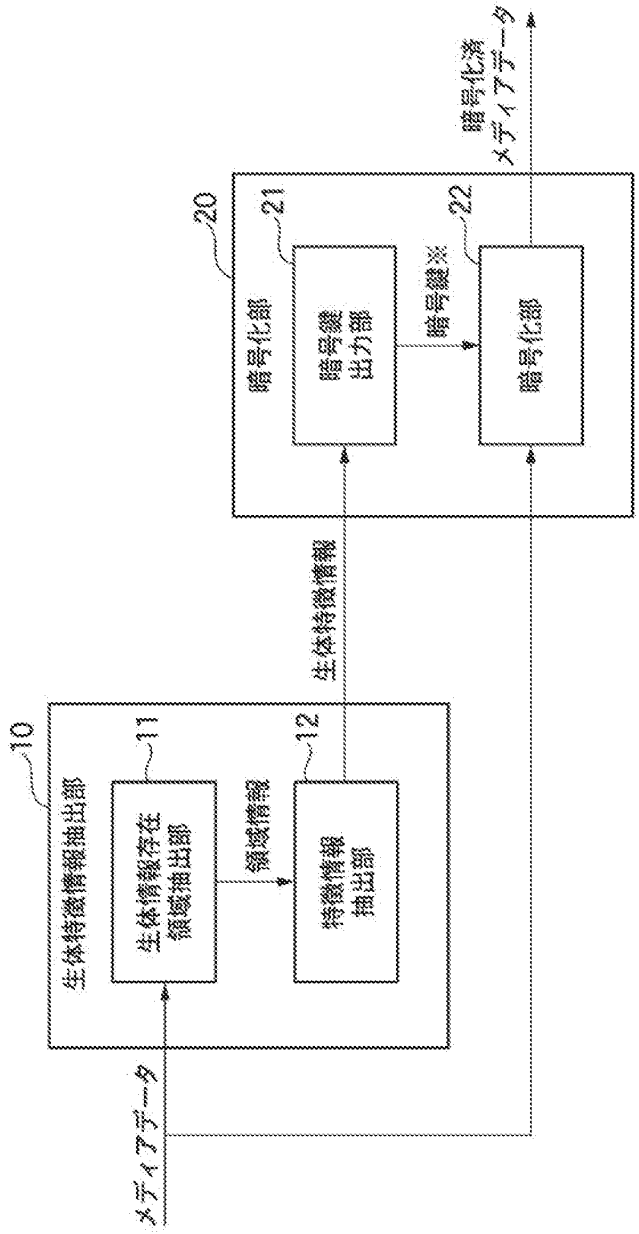
[11] 少なくとも1つの画像情報を含むメディアデータがらから暗号化されたメディアデータを生成するメディアデータ処理方法であって、

入力される前記メディアデータからその画像情報に含まれている少なくとも一人の人物に関する生体情報の特徴量を抽出し、

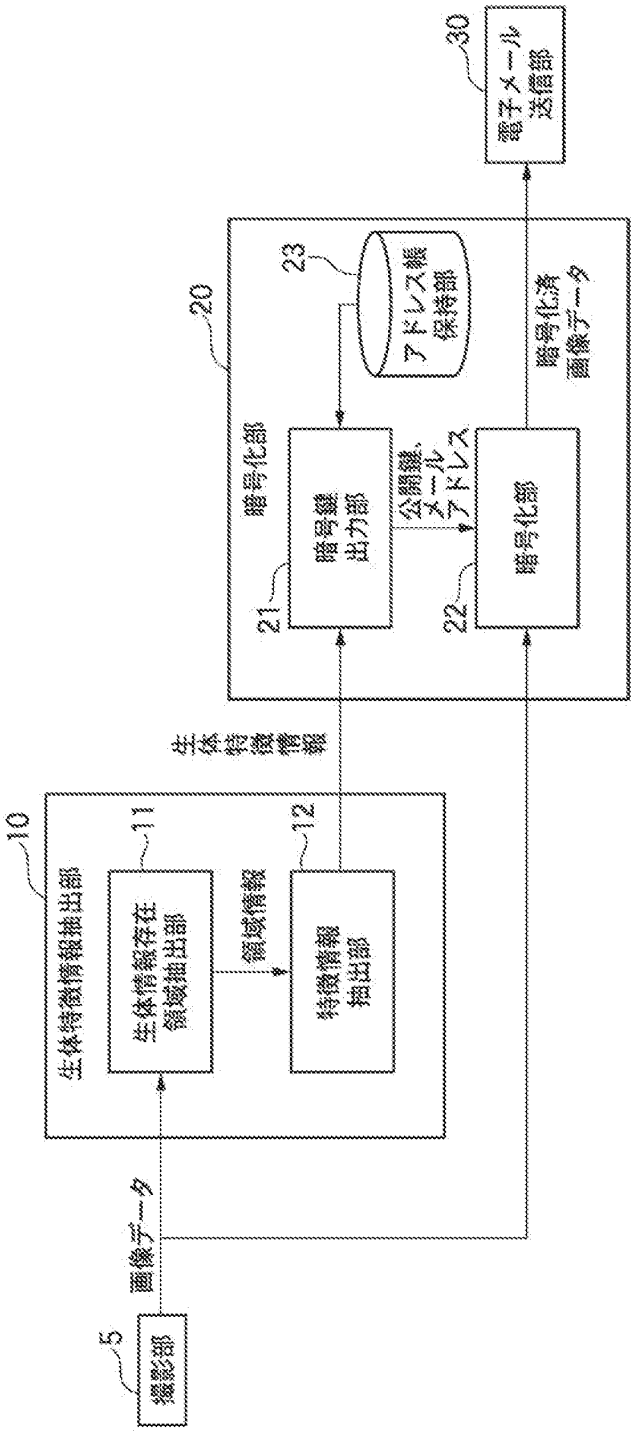
抽出された前記特徴量に基づいて前記メディアデータを部分的にもしくは全体的に暗号化する

ことを特徴とするメディアデータ処理方法。

[図1]

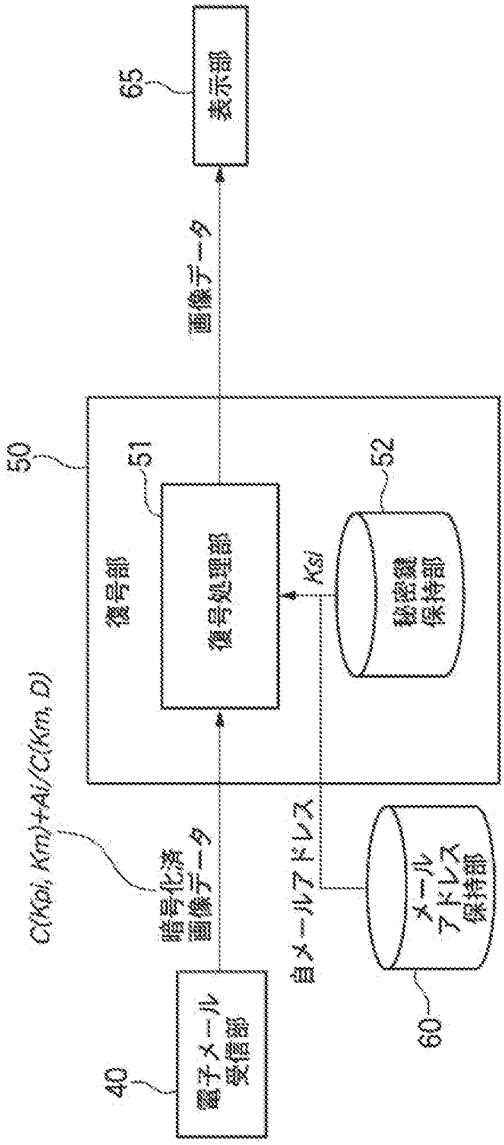

暗号化済
メディアデータ
20
21
22
暗号化部
暗号鍵
出力部
暗号鍵※
暗号化部
生体特徴情報
10
11
12
生体特徴情報抽出部
生体情報存在
領域抽出部
領域情報
特徴情報
抽出部
メディアデータ

[図2]

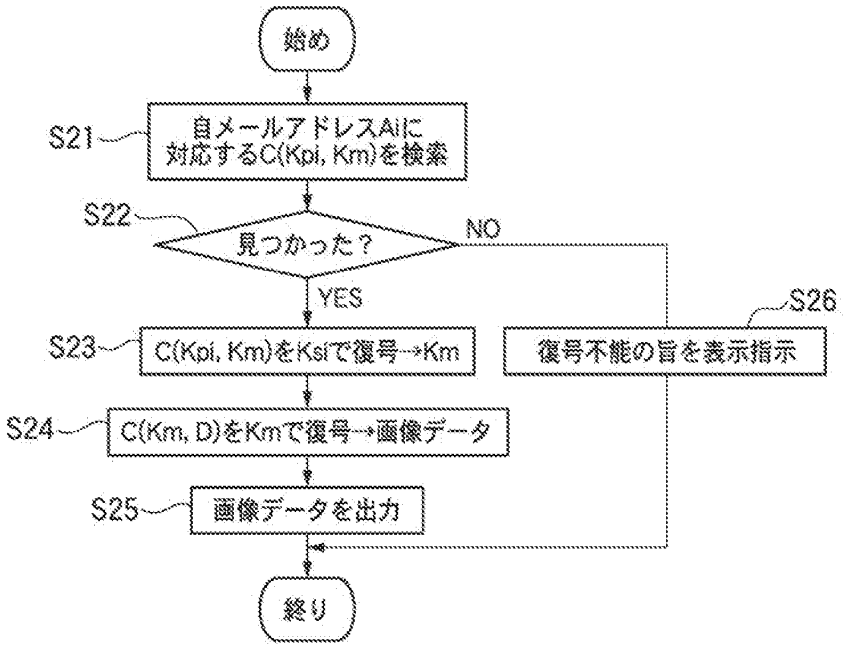

30
電子メール
送信部
20
23
アドレス帳
保持部
暗号化部
暗号鍵
出力部
公開鍵、
メール
アドレス
暗号化部
暗号化済
画像データ
21
22
生体特徴情報
10
生体特徴情報抽出部
11
生体情報存在
領域抽出部
領域情報
12
特徴情報
抽出部
画像データ
5
撮影部

生体情報存在領域抽出部
特徴情報抽出部
暗号鍵出力部
暗号化処理部
画像データD
画像データDのうち生体情報を含む領域Riを抽出
S11
領域情報
画像データDを暗号化するメディア鍵(共通鍵)Kmを生成
S15
画像データDをKmで暗号化→C(Km, D)
S16
領域Riに含まれる生体特徴情報(ベクトル)Piを抽出
S12
生体特徴情報Pi
S13
Piに対応する公開鍵KpiとメールアドレスAiをアドレス帳から検索
S14
見つかった？
YES
Kpi, Ai
すべてのiについて繰り返す
KmをKpiで暗号化→C(Kpi, Km)
S17
C(Kpi, Km)+AiをC(Km, D)へ結合→暗号化済画像データ
S18
暗号化済画像データ

[図4]


65
表示部
画像データ
50
51
52
復号部
復号処理部
Ksi
秘密鍵
保持部
C(Kpi, Km)+Ai/C(Km, D)
暗号化済
画像データ
自メールアドレス
メール
アドレス
保持部
40
電子メール
受信部
60

[図5]


始め
S21
自メールアドレスAiに
対応するC(Kpi, Km)を検索
S22
見つかった？
NO
YES
S23
C(Kpi, Km)をKsiで復号→Km
S24
C(Km, D)をKmで復号→画像データ
S25
画像データを出力
S26
復号不能の旨を表示指示
終り

領域情報

	左上のX, Y座標		右下のX, Y座標	
	X1	Y1	X2	Y2
R1	203	56	298	196
R2	288	36	404	176
R3	106	301	227	420
R4	266	298	388	460
R5	412	254	572	416

（原点=画像データの左上隅）

特徴点情報（各領域の左上が原点）

	X1	Y1
F1	60	161
F2	130	167
F3	69	184
F4	124	193
F5	151	270

距離F1～F2と距離F3～F2の比率α
=sqrt((130-60)^2+(167-161)^2)/sqrt((124-69)^2+(193-184)^2)
=1.26

距離F2～F5と距離F4～F5の比率β
=sqrt((124-151)^2+(193-270)^2)/sqrt((151-124)^2+(270-193)^2)
=1.29

生体特徴情報

P1=(1.26, 1.29)

P1=(1.26, 1.29)との距離が所定値（たとえば0.02）以内のもののうちで最も距離が小さいものを検索結果とする。

名前	メールアドレス	生体特徴情報	公開鍵(Kpi) 1024bit
A	AAA@AAA. AAA	:	:
B	BBB@BBB. BBB	(1.25, 1.30)	mQGiBDxD/tERBADPOQ192X...
C	CCC@CCC. CCC	:	:
D	DDD@DDD. DDD	:	:
E	EEE@EEE. EEE	:	:
F	FFF@FFF. FFF	:	:
G	GGG@GGG. GGG	:	:
H	HHH@HHH. HHH	:	:
		:	:

メディア鍵（128bit；疑似乱数に基づいて生成）

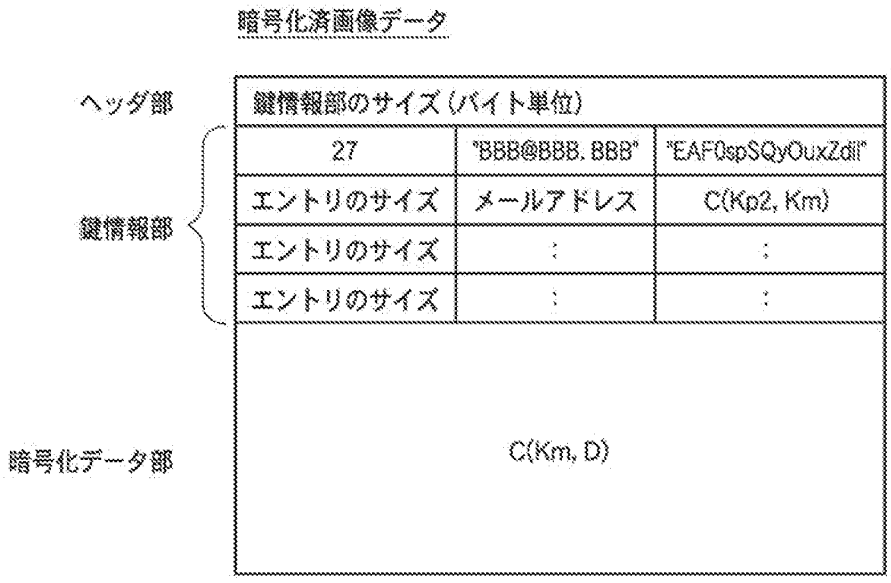
Km = "25U7M3BumJjR6blP"

➡

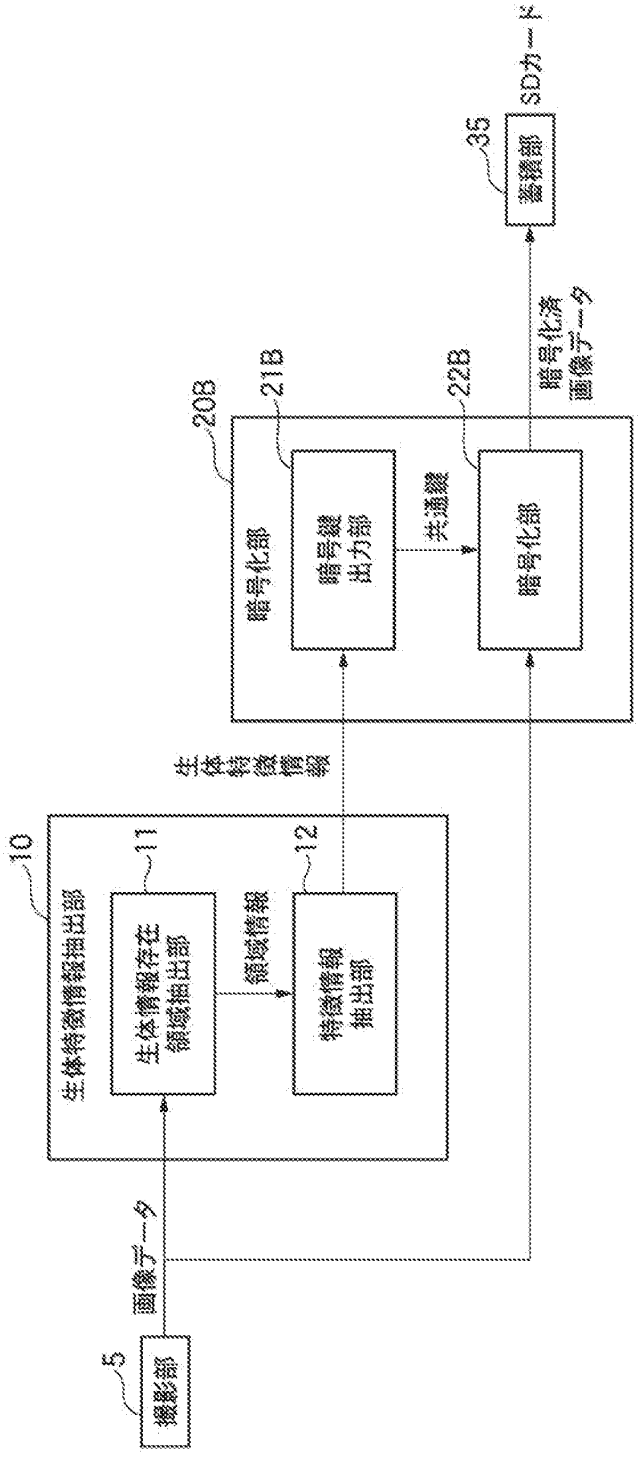
公開鍵Kp1で暗号化したメディア鍵

C(Kp1, Km) ="EAF0spSQyOuxZdil"

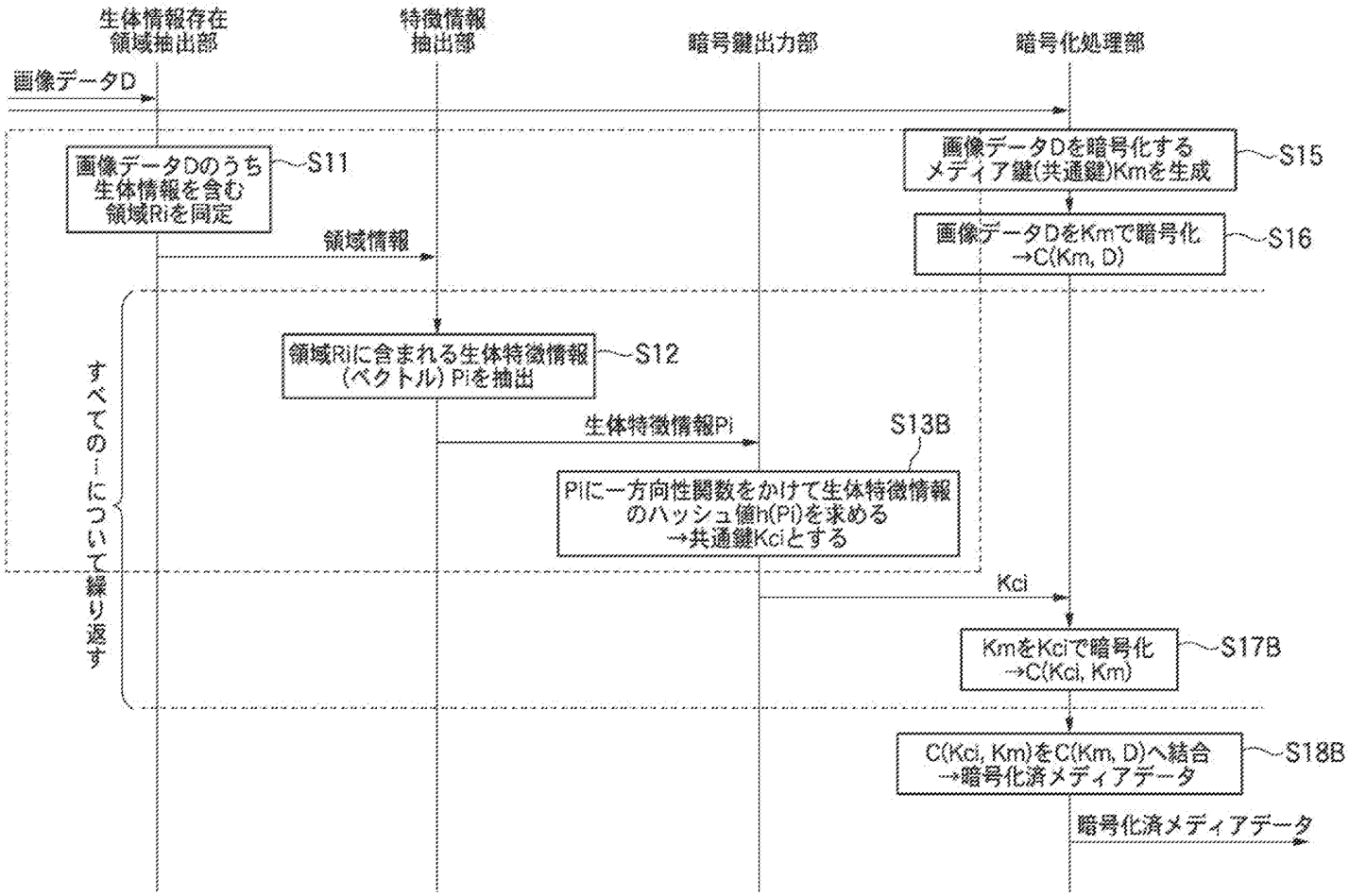
[図9]


暗号化済画像データ
ヘッダ部
鍵情報部
暗号化データ部
鍵情報部のサイズ（バイト単位）
27
"BBB@BBB. BBB"
"EAF0spSQyOuxZdl"
エントリのサイズ
メールアドレス
C(Kp2, Km)
エントリのサイズ
:
:
エントリのサイズ
:
:
C(Km, D)

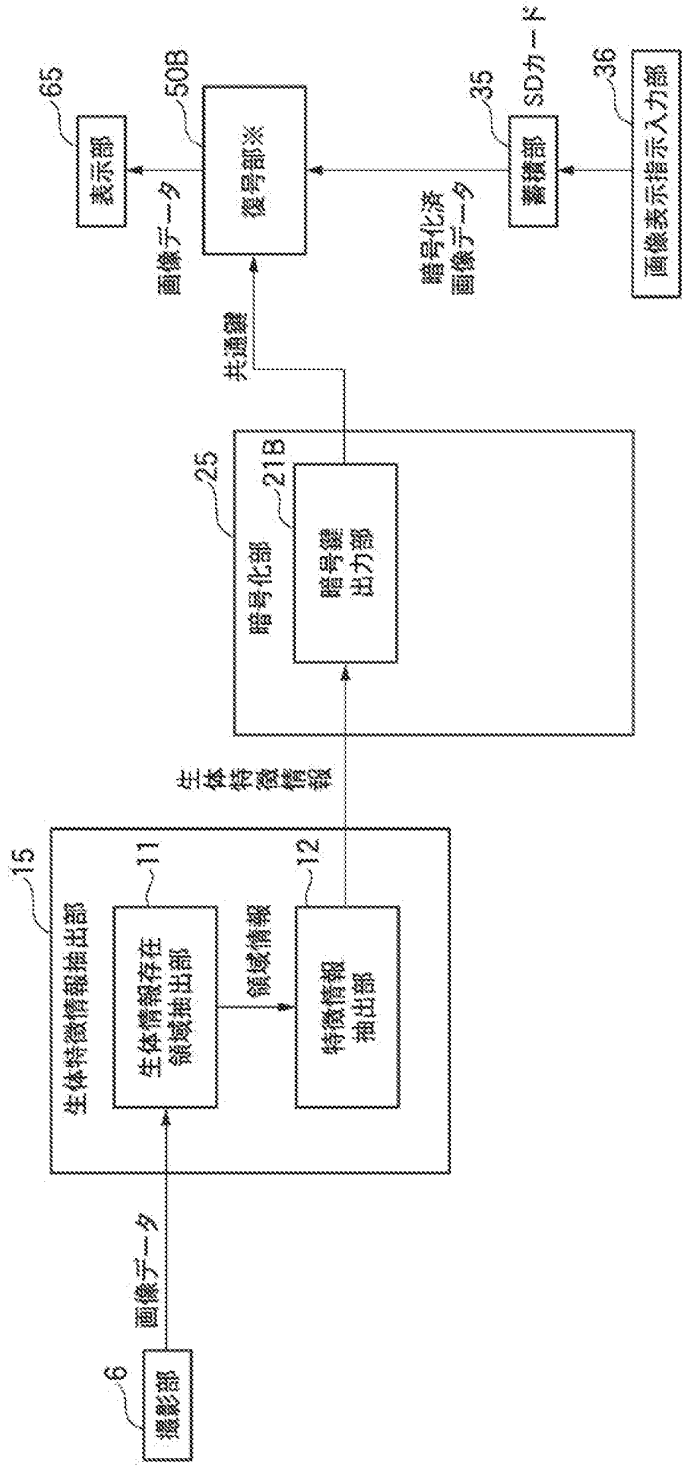
[図10]


SDカード
35
蓄積部
暗号化済
画像データ
20B
21B
22B
暗号化部
暗号鍵
出力部
共通鍵
暗号化部
生体特徴情報
10
生体特徴情報抽出部
11
12
生体情報存在
領域抽出部
領域情報
特徴情報
抽出部
画像データ
5
撮影部

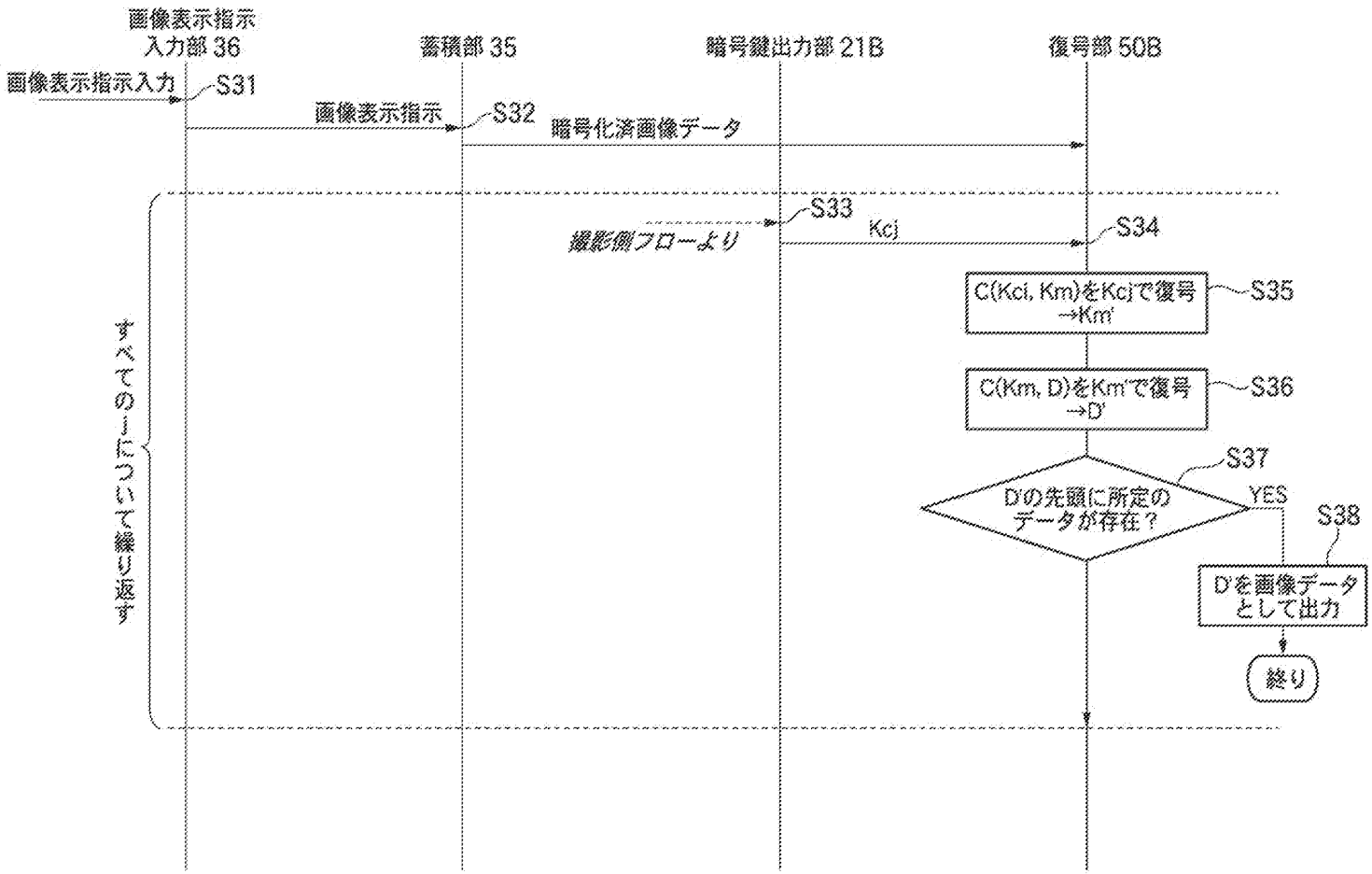
[図11]


生体情報存在領域抽出部
特徴情報抽出部
暗号鍵出力部
暗号化処理部
画像データD
画像データDのうち生体情報を含む領域Riを同定
S11
領域情報
画像データDを暗号化するメディア鍵(共通鍵)Kmを生成
S15
画像データDをKmで暗号化→C(Km, D)
S16
すべてのiについて繰り返す
領域Riに含まれる生体特徴情報(ベクトル)Piを抽出
S12
生体特徴情報Pi
S13B
Piに一方向性関数をかけて生体特徴情報のハッシュ値h(Pi)を求める→共通鍵Kciとする
Kci
KmをKciで暗号化→C(Kci, Km)
S17B
C(Kci, Km)をC(Km, D)へ結合→暗号化済メディアデータ
S18B
暗号化済メディアデータ

[図12]


撮影部
6
画像データ
15
生体特徴情報抽出部
11
生体情報存在領域抽出部
領域情報
12
特徴情報抽出部
生体特徴情報
25
暗号化部
21B
暗号鍵出力部
共通鍵
65
表示部
50B
復号部※
画像データ
35
SDカード
蓄積部
暗号化済画像データ
36
画像表示指示入力部

[図13]

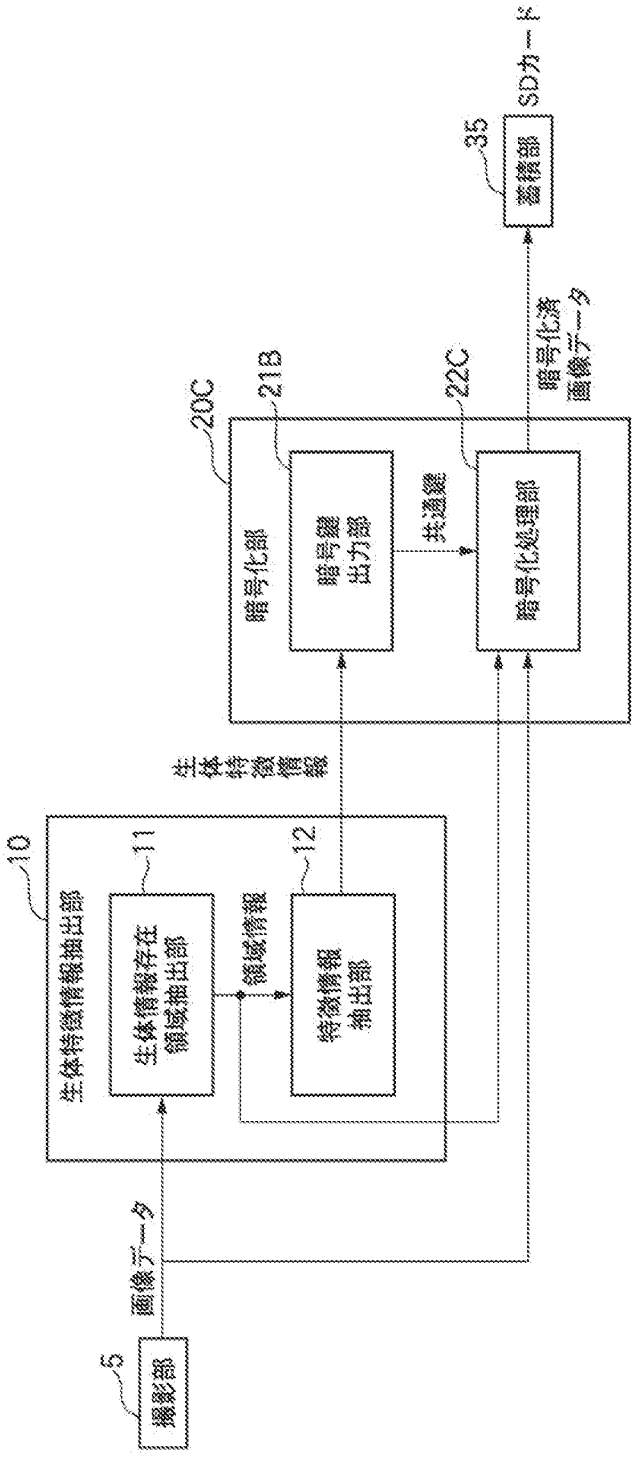


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0445
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

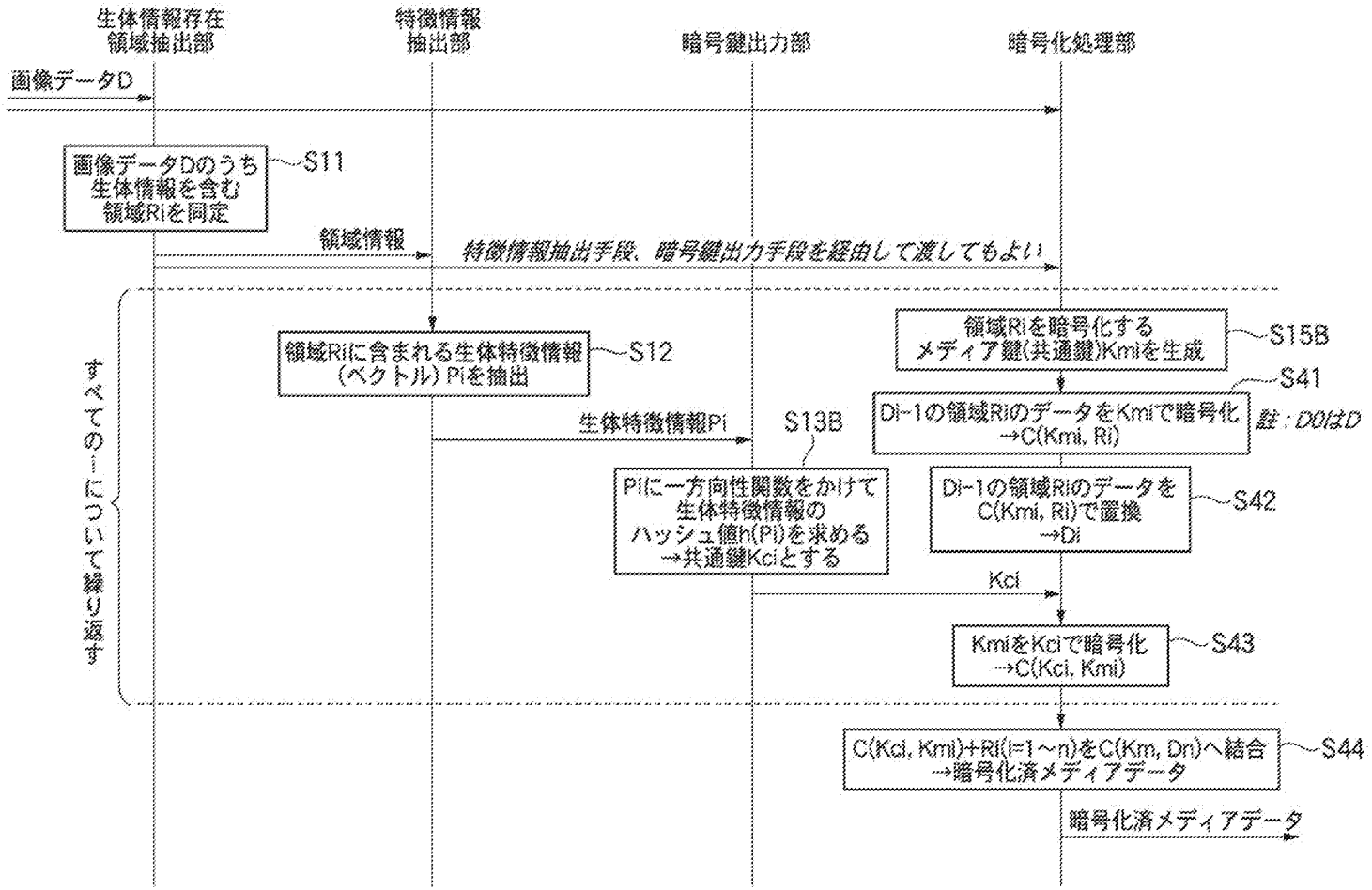
[図14]

生体特徴情報
P1 = (1.26, 1.29)
生体特徴情報のハッシュ値
h(P1) = "sY9aFf9zmaL3Wi1z"
共通鍵Kc1で暗号化したメディア鍵
C(Kc1, Km) = "8QAoMmLtN/b+EykM"
暗号化済画像データ
ヘッダ部
鍵情報部
暗号化データ部
鍵情報部のサイズ（バイト単位）
"8QAoMmLtN/b+EykM"
C(Kc2, Km)
:
:
C(Km, D)

[図15]


SDカード
35
蓄積部
暗号化済
画像データ
20C
21B
22C
暗号化部
暗号鍵
出力部
共通鍵
暗号化処理部
生体特徴情報部
10
11
12
生体特徴情報抽出部
生体情報存在
領域抽出部
領域情報
特徴情報
抽出部
画像データ
5
撮影部

[図16]


生体情報存在領域抽出部
特徴情報抽出部
暗号鍵出力部
暗号化処理部
画像データD
画像データDのうち生体情報を含む領域Riを同定
S11
領域情報
特徴情報抽出手段、暗号鍵出力手段を経由して渡してもよい
すべてのiについて繰り返す
領域Riに含まれる生体特徴情報(ベクトル)Piを抽出
S12
領域Riを暗号化するメディア鍵(共通鍵)Kmiを生成
S15B
S41
Di-1の領域RiのデータをKmiで暗号化→C(Kmi, Ri)
但し：D0はD
生体特徴情報Pi
S13B
Piに一方向性関数をかけて生体特徴情報のハッシュ値h(Pi)を求める→共通鍵Kciとする
Di-1の領域RiのデータをC(Kmi, Ri)で置換→Di
S42
Kci
KmiをKciで暗号化→C(Kci, Kmi)
S43
C(Kci, Kmi)+Ri(i=1～n)をC(Km, Dn)へ結合→暗号化済メディアデータ
S44
暗号化済メディアデータ

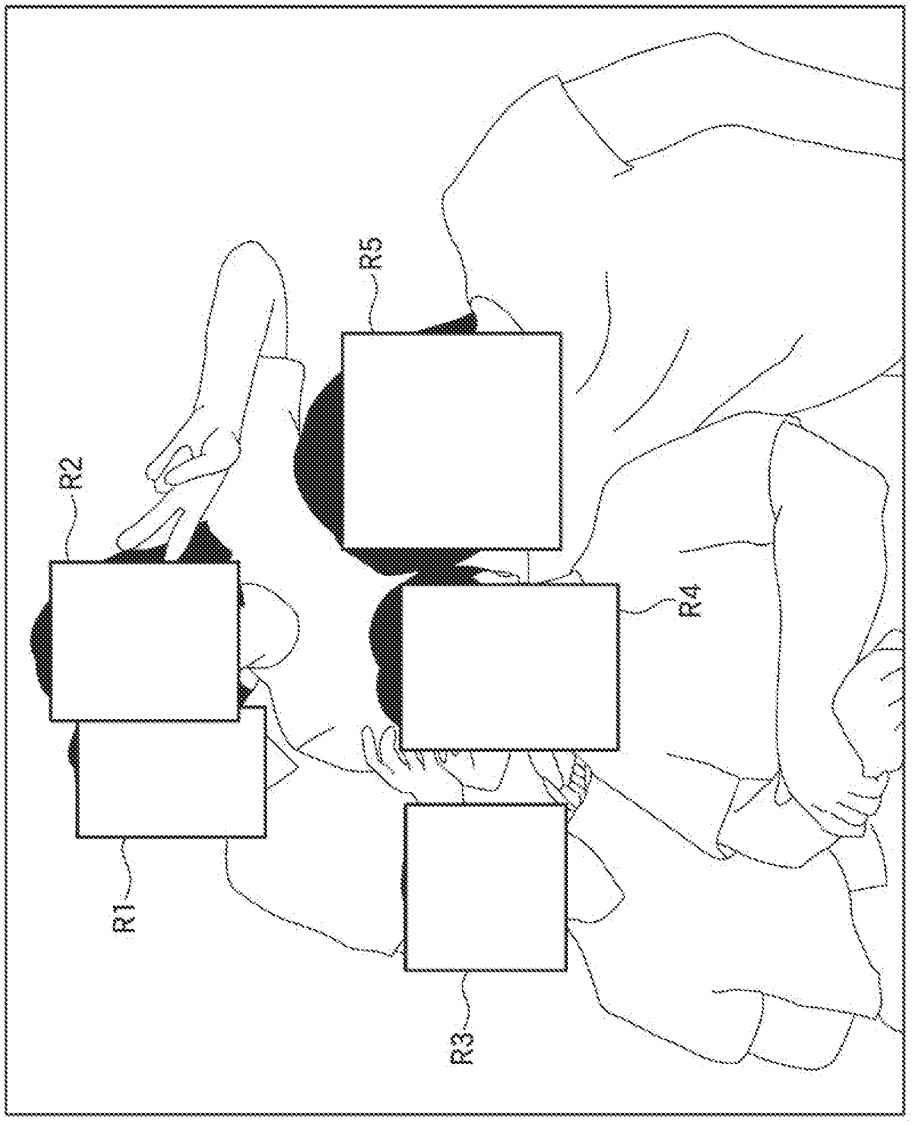
[図17]

画像表示指示
入力部 36
蓄積部 35
暗号鍵出力部 21B
復号部 50B
画像表示指示入力
S31
画像表示指示
S32
暗号化済画像データ
Dn→D'
S51
S33
撮影側フローより
Kcj
C(Kci, Kmi) (i=1～n)をKcjで復号
→Kmi'
S52
D'の領域Ri (i=1～n)の内容をKmi'で復号
→D'
S53
すべてのiについて繰り返す
D'を画像データとして出力
S54

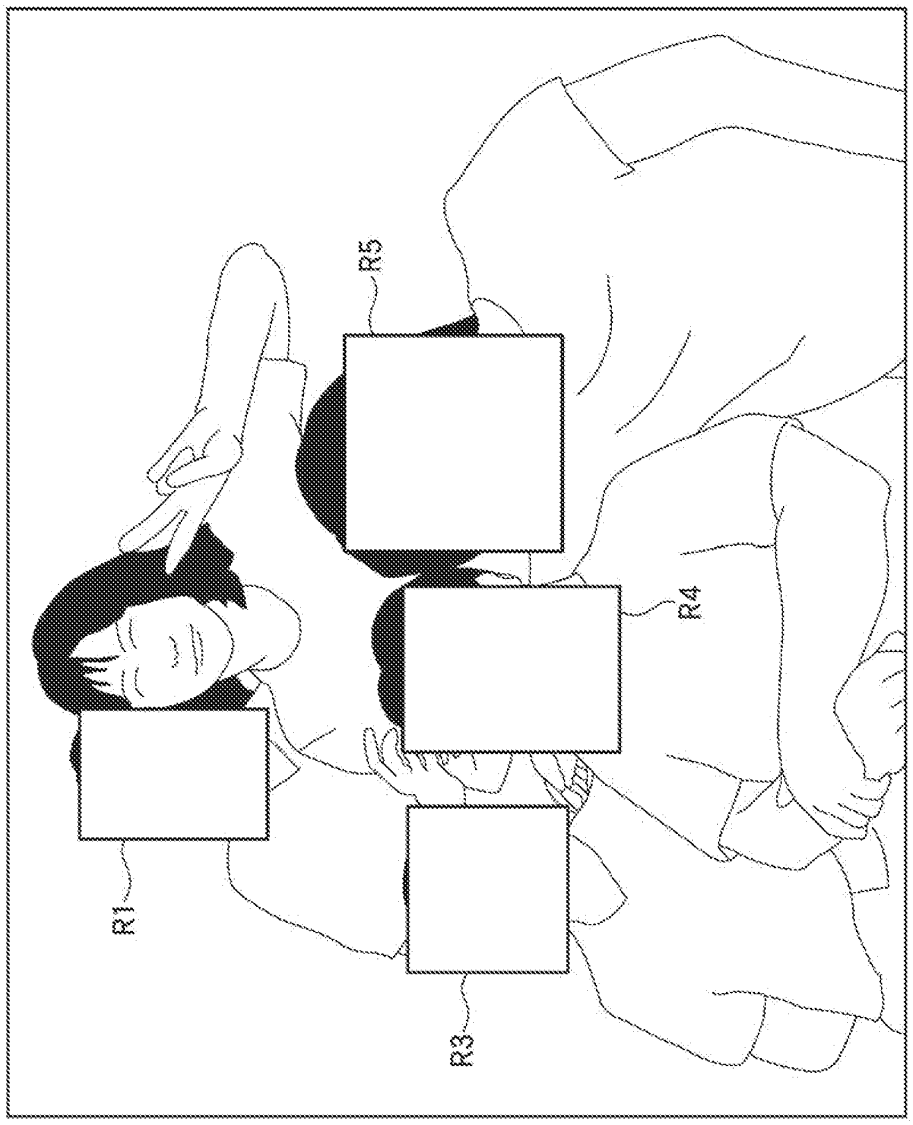
[図18]

生体特徴情報
P1 = (1.26, 1.29)
生体特徴情報のハッシュ値
h(P1) = "sY9aFf9zmaL3Wl1z"
共通鍵Kc1で暗号化したメディア鍵
C(Kc1, Km1) = "8QAoMmLtN/b+EykM"
暗号化済画像データ
ヘッダ部
鍵情報部のサイズ（バイト単位）
鍵情報部
(nエントリ)
203, 56, 298, 196
"8QAoMmLtN/b+EykM"
R2
C(Kc2, Km2)
暗号化データ部
Dn

[図19]



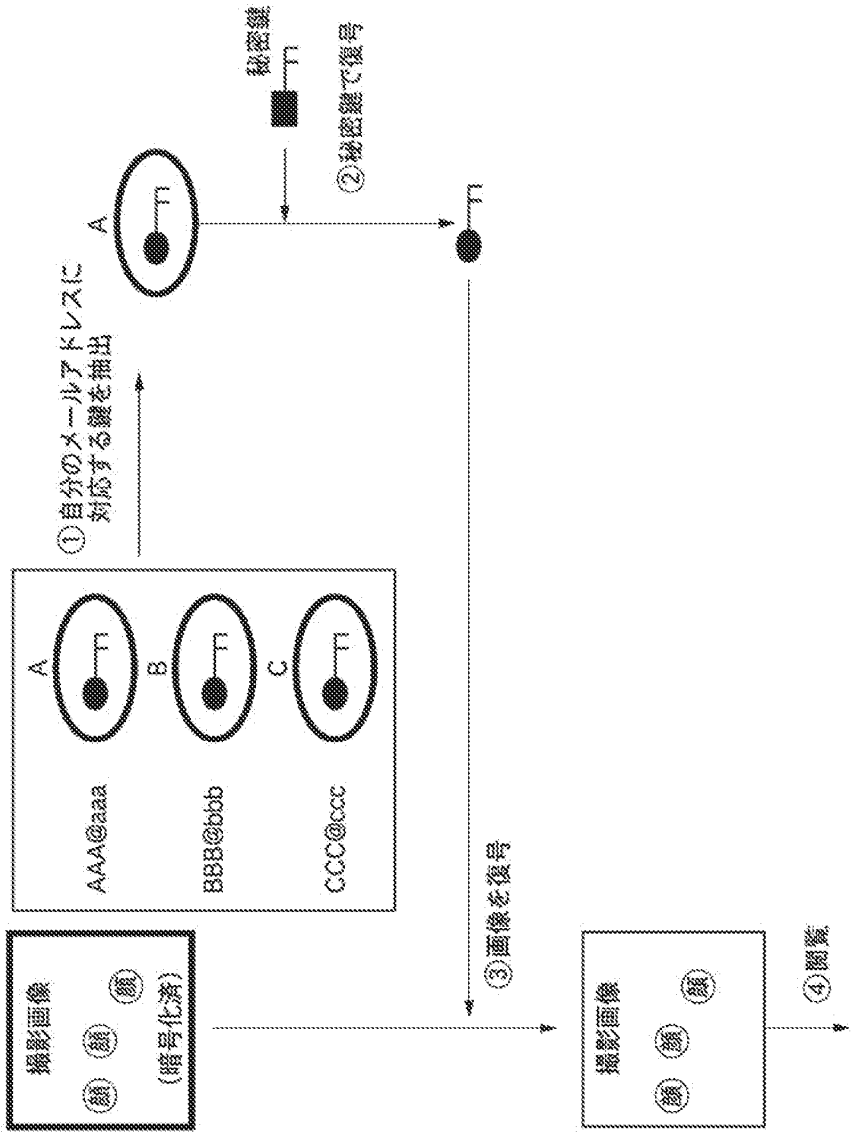
[図20]



[図21]

②顔領域を抽出
名刺データ等から作成したアドレス帳
①撮影
撮影画像
顔
顔
顔
氏名
メールアドレス
顔パラメータ
公開鍵
A
AAA@aaa
0011110000…
B
BBB@bbb
0022220000…
C
CCC@ccc
0033330000…
③特徴パラメータを照合
→被撮影者を特定
④暗号鍵を生成
⑤画像を暗号化
撮影画像
顔
顔
顔
(暗号化済)
⑥鍵を暗号化
A
AAA@aaa
B
BBB@bbb
C
CCC@ccc
⑦連結
⑧配付

[図22]


秘密鍵
②秘密鍵で復号
①自分のメールアドレスに
対応する鍵を抽出
A
B
C
AAA@aaa
BBB@bbb
CCC@ccc
撮影画像
(顔)(顔)(顔)
(暗号化済)
③画像を復号
撮影画像
(顔)(顔)(顔)
④閲覧

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2006/314296

A. CLASSIFICATION OF SUBJECT MATTER
H04L9/08(2006.01)i, *G06F21/24*(2006.01)i, *H04N7/167*(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L9/08, G06F21/24, H04N7/167

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2006
Kokai Jitsuyo Shinan Koho 1971-2006 Toroku Jitsuyo Shinan Koho 1994-2006

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y A	JP 2005-229268 A (KDDI Corp.), 25 August, 2005 (25.08.05), Particularly, Par. Nos. [0014] to [0036] (Family: none)	1,5-7,11 2-4,8,9 10
Y	JP 2001-211161 A (Pumpkin House, Inc.), 03 August, 2001 (03.08.01), Particularly, Par. Nos. [0034] to [0048] (Family: none)	2-4
Y	JP 2005-267146 A (NEC Corp.), 29 September, 2005 (29.09.05), Particularly, Par. Nos. [0013] to [0021] (Family: none)	3

☑ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
01 September, 2006 (01.09.06)	12 September, 2006 (12.09.06)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2006/314296

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP 2000-358025 A (NEC Corp.), 26 December, 2000 (26.12.00), Particularly, Par. Nos. [0026] to [0041], [0063], [0064] (Family: none)	4,8,9
A	JP 2000-56681 A (Casio Computer Co., Ltd.), 25 February, 2000 (25.02.00), Particularly, Par. Nos. [0064] to [0068] (Family: none)	1-11
A	JP 2002-305704 A (Canon Inc.), 18 October, 2002 (18.10.02), Particularly, Par. Nos. [0064] to [0068] (Family: none)	1-11

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

国際調査報告	国際出願番号 ＰＣＴ／ＪＰ２００６／３１４２９６

A． 発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl. H04L9/08(2006.01)i, G06F21/24(2006.01)i, H04N7/167(2006.01)i

B． 調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Int.Cl. H04L9/08, G06F21/24, H04N7/167

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２００６年
日本国実用新案登録公報	１９９６－２００６年
日本国登録実用新案公報	１９９４－２００６年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C． 関連すると認められる文献

引用文献のカテゴリー＊	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
X	JP 2005-229265 A（ＫＤＤＩ株式会社）2005.08.25（ファミリなし）	1,5-7,11
Y	特に第14-36段落を参照	2-4,8,9
A		10
Y	JP 2001-211161 A（株式会社パンプキンハウス）2001.08.03（ファミリなし） 特に第24-48段落を参照	2-4
Y	JP 2005-267146 A（日本電気株式会社）2005.09.29（ファミリなし） 特に第13-21段落を参照	3

☑ Ｃ欄の続きにも文献が列挙されている。 ☐ パテントファミリーに関する別紙を参照。

＊ 引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
０１．０９．２００６	１２．０９．２００６

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）		
日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号		５Ｓ	９３６４
	中里 裕正		
	電話番号 ０３－３５８１－１１０１ 内線 ３５４６		

様式ＰＣＴ／ＩＳＡ／２１０（第2ページ）（２００５年４月）

C（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	JP 2000-358025 A（日本電気株式会社）2000.12.26（ファミリなし） 特に第26-41、63、64段落を参照	4,8,9
A	JP 2000-56681 A（カシオ計算機株式会社）2000.02.25（ファミリなし） 特に第64-68段落を参照	1-11
A	JP 2002-305704 A（キヤノン株式会社）2002.10.18（ファミリなし） 特に第64-68段落を参照	1-11

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0458
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 May 2009 (22.05.2009)

PCT

(10) International Publication Number
WO 2009/065137 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2008/083816

(22) International Filing Date:
17 November 2008 (17.11.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/988,513 16 November 2007 (16.11.2007) US

(71) Applicant *(for all designated States except US)*: **DIVX, INC.** [US/US]; 4780 Eastgate Mall, San Diego, CA 92121 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **SOROUSHIAN, Kourosh** [US/US]; 4780 Eastgate Mall, San Diego, CA 92121 (US). **OSBORNE, Roland** [US/US]; 4780 Eastgate Mall, San Diego, CA 92121 (US). **BRANESS, Jason** [US/US]; 4780 Eastgate Mall, San Diego, CA 92121 (US). **PRIYADARSHI, Shaiwal** [GB/US]; 4780 Eastgate Mall, San Diego, CA 92121 (US).

(74) Agent: **KAUTH, POMEROY, PECK & BAILEY LLP**, P.O. Box 19152, Irvine, CA 92623 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BE, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
- *with international search report*

(54) Title: HIERARCHICAL AND REDUCED INDEX STRUCTURES FOR MULTIMEDIA FILES

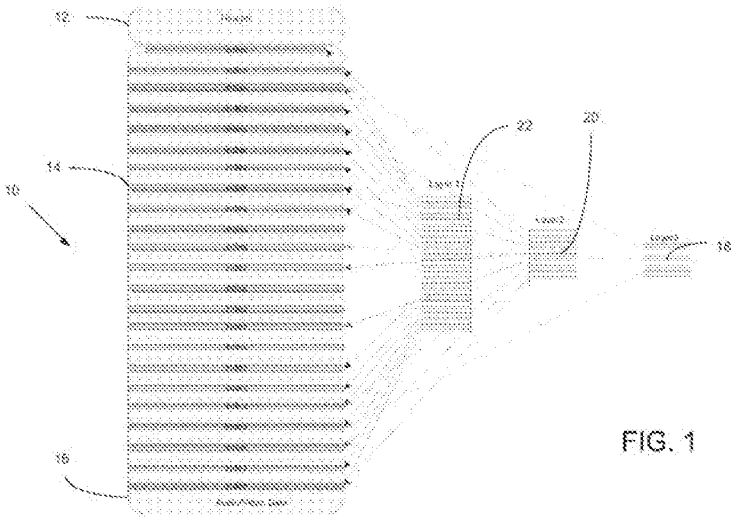


FIG. 1

(57) Abstract: Playback and distribution systems and methods for multimedia files are provided. The multimedia files are encoded with indexes associated with the content data of the multimedia files. Through the use of the indexes, playback of the content is enhanced without significantly increasing the file size of the multimedia file.

WO 2009/065137 A1

HIERARCHICAL AND REDUCED INDEX STRUCTURES FOR MULTIMEDIA FILES

BACKGROUND

[0001] The present invention relates generally to multimedia files and more specifically to the indexing of information within a multimedia file.

[0002] In recent years, the playback of multimedia files has become an integrated part of the average consumer's daily life. Cellular telephones, DVD players, personal computers, and portable media players are all examples of devices that are capable of playing a variety of multimedia files. While each device may be tailored to a particular multimedia format, the extensive proliferation of these devices encourages a certain level of interoperability amongst the different device classes and categories. Likewise, there are certain features such as fast-forward, reverse, start, stop, play, and pause which are expected to behave similarly across all device categories, despite their performance capabilities and use-case application.

[0003] One of the most common features of media playback devices is the support for random access, fast-forward and reverse playback of a multimedia file, which is sometimes referred to as "trick play". Performing trick play functionality generally requires displaying the video presentation at a higher speed in forward and reverse direction, and resuming the overall presentation from a position close to where the viewer terminated the video trick play activity. The audio, subtitle, and other elements of the presentation are typically not used during trick play operations, even though that can be subject to a device's operating preference. In accommodating trick play functionality, multimedia files typically contain an index section used to determine the location of all frames, and specifically the video frames which can be independently decoded and presented to the viewer. When all index information is stored in a single location within a file and linearly references the multimedia information within the file,

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0460
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

a player must seek to a specific index entry in order to be able to play a file. For example, a player that is instructed to play a multimedia presentation at the half-way point of the presentation typically processes the first half of the index data before being able to determine the set of data points required to commence playing.

[0004] The index section has many other potential applications as well: it may be a necessary element in basic playback of multimedia files that exhibit poor multiplexing characteristics; the index section may also be used to skip over non-essential information in the file; also, an index is often required for the resumption of playback after the termination of trick play functions.

Summary

[0005] Embodiments of the invention utilize indexes that can increase the efficiency with which a player can perform a variety of functions including trick play functions. In several embodiments, the index is a hierarchical index. In many embodiments, the index is a reduced index and, in a number of embodiments, the index is expressed using bit field flags and associated data fields.

[0006] In one embodiment, a method of playing back content by a playback device stored in a media file is provided. The method comprises providing a media file to a playback device, the media file having content data, cue data and a hierarchical index, the content data having a plurality of media frames, the cue data being associated with each frame of the plurality of frames and the hierarchical index being associated with a subset of the cue data; decoding the content data by a playback device; displaying content on a display screen from the decoded content data; receiving a user request; locating the index based on the user request; and decoding one or more media frames from the subset of the cue data based on the located index.

[0007] In another embodiment, a method of encoding a media file for playing back by a playback device comprises incorporating seek locations with content data, the content

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0461
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

data having audio, video and subtitle tracks, the seek locations identifying particular timestamps for particular frames within the audio, video or subtitle tracks; incorporating an index referencing the incorporated seek locations; and creating a media file having the content data, the incorporated seek locations, and the incorporated index.

[0008] In yet another embodiment, a system for playback of a media file comprises a media server configured to transmit a media file having at least one index having pointers that reference at least one frame of media content within the media file. A client processor in network communication with the media server and configured to send requests for the media file to the media server is also provided. The media server is configured to transmit the media file requested and the client processor comprises a playback engine that is configured to locate a pointer from the at least one index to locate and decode the portions of the media file based on the located pointer to comply with user playback instructions.

[0009] In one embodiment, a method of decoding a multimedia file having an index included with the multimedia file is provided. The method comprises locating an index within a multimedia file, the index having a plurality of index data-points and the multimedia file having a plurality of frames and a plurality of cue data-points having a time stamp and a location of a specific frame of the plurality of frames; loading the located index into storage; locating an index data-point from the plurality of index data-points based on a timestamp; and locating a cue data-point from the plurality of cue-data points based on the located index data-point.

[0010] The above-mentioned and other features of this invention and the manner of obtaining and using them will become more apparent, and will be best understood, by reference to the following description, taken in conjunction with the accompanying

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0462
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

drawings. The drawings depict only typical embodiments of the invention and do not therefore limit its scope.

Brief Description of the Drawings

[0011] FIG. 1 is a graphical representation of an index structure within a multimedia file in accordance with an embodiment of the invention.

[0012] FIG. 2A is a graphical representation of an index structure following the audio/video data of a multimedia file in accordance with an embodiment of the invention.

[0013] FIG. 2B is a graphical representation of an index structure interleaved within the audio/video data of a multimedia file in accordance with another embodiment of the invention.

[0014] FIG. 2C is a detailed graphical representation of an index structure relative to other portions of a multimedia file in accordance with an embodiment of the invention.

[0015] FIG. 2D is a graphical representation of an index structure relative to cue data of a multimedia file in accordance with an embodiment of the invention.

[0016] FIG. 3 is a graphical representation of index structure detailing bit flags and associated data filed within a multimedia file in accordance with an embodiment of the invention.

[0017] FIG. 4 is a graphical representation of index structure providing time codes and offset data fields within a multimedia file in accordance with an embodiment of the invention.

[0018] FIG. 5 is a graphical representation of index structure with time codes and multiple offset data fields relative to a size data field within a multimedia file in accordance with an embodiment of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0463
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0019] FIG. 6 is a graphical representation of index structure with time codes and primary offset data fields within a multimedia file in accordance with an embodiment of the invention.

[0020] FIG. 7 is a semi-schematic network diagram of playback system for streaming and fixed media file playback in accordance with an embodiment of the invention.

[0021] FIG. 8 is a flowchart of a process utilizing index structure within a multimedia file in accordance with an embodiment of the invention.

[0022] FIG. 9-11 are graphical representations with increasing detail of an index structure within a multimedia file in accordance with one embodiment of the invention and to further illustrate the process of FIG. 8.

DETAILED DESCRIPTION

[0023] Turning now to the drawings, multimedia files including indexes in accordance with embodiments of the invention are described. In a number of embodiments, the index is a hierarchical index. A hierarchical index is a representation of index information in a form that provides a coarse index to a few predetermined locations within the multimedia file followed by a further refined representation of the portions of the multimedia file. In many embodiments, the lowest level of the index is sufficiently granular as to identify every frame in the multimedia file. When a hierarchical index is used, a player need only request a small amount of relevant index information in order to commence playing a multimedia file. As such, the hierarchical index lowers the memory footprint needed by playback devices to effectively seek and perform trick-play operations on a multimedia file. Additionally, file load times for playback are reduced and trick-track load performance enhanced. In one embodiment, the hierarchical index has index information that includes offsets into cue points within a multimedia file with timestamps allows lookups to be fast and efficient.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0464
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0024] In several embodiments, the multimedia file includes a reduced index. Players in accordance with embodiments of the invention can utilize a reduced index to rapidly move between accesses or key-frames when performing trick play functions. The reduced index can be in conjunction with a hierarchical index. However, reduced indexes can be included in multimedia files that do not include a hierarchical index. A reduced index only provides the location of the accesses or key-frames within a multimedia file, along with a time-stamp value to indicate their corresponding time within the multimedia presentation. In a number of embodiments, bit field flags and associated data fields are used to represent index information. Such a representation can be used in accordance with embodiments of the invention to express index information, a hierarchical index and/or a reduced index.

HIERARCHICAL INDEXES

[0025] A multimedia file containing a hierarchical index in accordance with an embodiment of the invention is shown in FIG. 1. The multimedia file 10 includes header information 12, index information 14 interleaved amongst audio/video data 16 and a three layer hierarchical index. The coarsest layer 18 of the hierarchical index includes a small number of references to pieces of index information. The middle layer 20 and the finest layer 22 each include successively larger numbers of references to index information.

[0026] In many embodiments, the index information 14 interleaved amongst the audio/video data 16 lists the location of encapsulated audio, video, subtitle, and/or other similar data. Typically, each block of interleaved index information lists the encapsulated media immediately following the block of interleaved index information. In several embodiments, the index information 14 contains information that describes the absolute or relative location of the start of each piece of encapsulated media. In a number of embodiments, the interleaved index information 14 includes the size of each

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0465
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

indexed piece of encapsulated media, in addition to information indicating whether the indexed piece of encapsulated media can be used as an access or key-frame, its presentation time value, and other information, which may be helpful to a decoding device.

[0027] Each layer in the hierarchical index includes references to the interleaved index information 14 within the multimedia file 10. The implementation of the hierarchy structure can be inclusive or exclusive, meaning that the data in each layer can be repeated in the other layers or each layer may contain unique position information. In addition, the number of elements at each layer of a hierarchy and the total number of layers can be pre-determined, limited based on pre-determined values, or unbounded.

[0028] Although a specific implementation of a hierarchical index is shown in FIG. 1, hierarchical indexes can be implemented in many different ways. For example, the index values can be stored in a single part of the file, or distributed in clusters in the file. Multimedia files containing different distributions of index information in accordance with embodiments of the invention are shown in FIGs. 2A-B. For example, the index information could be appended or pre-pended to the audio/video data portion 16 of the multimedia file 10 as an entire unit 21. Index clusters 22 shown in FIG. 2B can also be woven into the audio/video data portion. In addition to distributing index information in different ways, the hierarchy itself can be implemented as a structure that points to the actual frames in a file (as opposed to blocks of index information), which may or may not start with access or key-frame positions.

[0029] FIG. 2C further details the hierarchical index 21 within a larger hypothetical file structure MKV file 200. This file structure is made of two primary sections, the EBML 24 and the Segment 26. In this file structure, the Segment may host the Seek Head 201, Segment Info 202, Tracks 203, Chapters 204, Cluster 205, Cues 29, and a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0466
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Hierarchical index 21. As shown, a plurality of hierarchical indexes 21 could be included with the multimedia file. Additionally, each hierarchical index can include multiple hierarchical index points 23. These index points in various embodiments have a timestamp 25 and a track position 27, specifying a specific media track 27a and a position or offset 27b from the timestamp 25. Cues 29 are also shown and as will be explained in more detail below are utilized by the index points 23 to increase access to specific points within a multimedia file. This dynamic structure for example is shown in FIG. 2D where multiple hierarchical index points 23 reference or point to multiple cue points 28. In various embodiments, the hierarchical index contains references to a fraction, e.g., one tenth, of cue points relative to the total number of cue points in a media file. One would appreciate that the references can increase to increase the granularity of pointers or references to the cue points.

[0030] A player attempting to decode a multimedia file that includes a hierarchical index in accordance with an embodiment of the invention typically uses the hierarchical index as necessitated by the functions the player is requested to perform. When trick play functions are requested, the player can locate an index in the hierarchy corresponding to a specific speed and decode each of the frames indicated by the index. The manner in which a specific frame is located using the index depends upon the nature of the index. In embodiments where each index in the hierarchy points directly to video frames, then the process is simple. In embodiments where the index points to additional index information within the multimedia file, the additional index information is accessed and used to locate a desired frame.

REDUCED INDEXES

[0031] Many multimedia files in accordance with embodiments of the invention use reduced index information. Reduced indexes can be used in conjunction with a hierarchical index or in multimedia files that do not include a hierarchical index. A

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0467
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

reduced index does not include information concerning every piece of multimedia information within a multimedia file. A reduced index typically is restricted to information concerning the location of access or key-frames and the time stamp of the access or key-frames. Access frames are generally video frames that can be independently decoded, although the reduced index can be used to point to any other type of key-frame for other streams stored in the multimedia file. The reduced index can enable a player to rapidly skip between key frames when performing trick play functions.

[0032] In a number of embodiments, a reduced index is only provided for a single or primary data type and offsets are provided for each of the other streams of data contained within the file which may be related to the primary data type. The offsets can be used by a player to facilitate synchronized playback of different media. In several embodiments, each piece of index information also includes the size of the access or key-frame and the data-type of the access or key-frame. A player decoding a multimedia file that contains a reduced index in accordance with an embodiment of the invention can use the reduced index to perform trick play functions in a similar fashion to the way in which a player uses a hierarchical index. The player can sequence through the reduced index inspecting the Timestamps of access or key frames to ascertain which frames to render in order to achieve a desired speed.

Expressing Index Information Using Bit Fields

[0033] Multimedia files in accordance with a number of embodiments of the invention utilize bit field flags and associated data fields to express index information. In many embodiments, the bit field flags are used to signal the presence of a set of corresponding variable length data fields that contain index information. Bit field flags 31 and data fields 32 that can be used to express index information concerning a piece of multimedia information in accordance with an embodiment of the invention are

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0468
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

shown in FIG. 3. In the illustrated embodiment, a set of bit-field flags signals the presence of additional data following the flags. The bit-field flags are specified as 8-bits in their entirety, but that is not necessarily a requirement for other implementations. The first bit of the flag may indicate an Absolute / Fixed Size field 31a, which determines whether the size of the frame is read from a pre-determined set of sizes stored in a separate section of the file, or whether they are available as a series of bytes following the flags field. Two additional bits, Fixed Size Index / Byte Numbers field 31b, are used to determine the index-position of the size value or the total number of bytes used to represent the value, depending on the setting of the Absolute / Fixed Size bit or field 31a. The next bit, a Primary Offset field 31c, determines the size of the offset value, which may be the location of the frame. This bit is selected amongst two pre-determined byte numbers, for example either a 4-byte value or 8-byte value. Likewise, a flag may indicate the presence of another predetermined offset, e.g., a Secondary Offset 33, which can be 4 bytes and represents a relative offset from the Primary Offset value. A bit 31e indicating the presence of a timecode byte sequence may also be present, along with another bit, Key Frame Flag bit 31f, which can be used to determine the presence of access or key frames. In many embodiments, bit field flags and data fields similar to those shown in FIG. 3 are used to index the location of all frames in a multimedia file.

[0034] The number of flags that can be represented via the structure shown in FIG. 3 is infinitely extensible using a "Flags Extension" bit 31g which signals the presence of a follow-on flag. Here, one bit 31h may be referred to as "Associated Offsets". Associated offsets may then signal the presence of a byte value, which is used to determine the number of streams which correspond to the current frame. These relative offsets may use the same flag and subsequent index information for other frames in the stream, to be used for synchronization purposes. The frames identified by the relative offsets,

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0469
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

when played back correctly, may provide a synchronized presentation of audio, video, subtitles, and other related data. The stream number value 32 often corresponds to the actual stream numbers stored in the file.

[0035] Index information represented using the two relative offset values 41a,b is shown in FIG. 4. In many embodiments, the data type for each frame is indicated for an entire group of frames, or alternatively is indicated on a frame-by-frame basis, in which case a "Data Type" field 35 is added to the index-structure. The presence of a Timecode value 37 to indicate the exact time of a frame in an overall presentation may be done via a set of pre-determined specifications. For example, the Timecode value could be required for all video access frames; alternatively, the presence of a Timecode could be mandatory on a periodic basis for audio samples. It is only important to note that the Timecode value is optionally present and is indicated by a corresponding bit-flag.

[0036] Through a set of pre-determined rules, structures similar to those described above can be applied for the representation of hierarchical indexing in accordance with embodiments of the invention. For example, the "Primary Offset" value 50 can point to a specific index position, along with the Timecode value 52 indicating the exact time-stamp of the index. An additional bit-field 39, the "SubIndex", can point to a relative offset from the position indicated by the "Primary Offset". This "SubIndex" position 54 is a refinement from the beginning of a larger index cluster. Use of various values to construct a hierarchical index in accordance with an embodiment of the invention is shown in FIG. 5.

[0037] Bit field flags and associated data fields can also be used to represent a reduced index structure pointing to a series of access or key frames for a particular stream in a file. A reduced index in accordance with an embodiment of the invention is shown in FIG. 6. In the illustrated embodiment, the "flags" field 602 is followed by a corresponding set of size bytes 604, a "Primary Offset" value 606, and a Timecode 608.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0470
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The access frames may typically be related to video frames in a file, though again this field could be defined for all stream types in a file. The structure 600 shown in FIG. 6 stores the location of all access or key-frames, and can contain the location of all related offsets for the encapsulated tracks in the file.

[0038] It is important to note that the use of flexible bit field flags enables the implementation of multiple data structures which may appear in the hierarchical, reduced, and conventional indexing schemes. The use of bit fields as flags indicating variable length data can help optimize the size of an overall index because not all members are in general required by all frames.

[0039] Referring now to FIG. 7, a progressive playback system in accordance with an embodiment of the invention is shown. The playback system 190 includes a media server 192 connected to a network 194. Media files are stored on the media server 194 and can be accessed by devices configured with a client application. In the illustrated embodiment, devices that access media files on the media server include a personal computer 196, a consumer electronics device such as a set top box 18 connected to a playback device such as a television 200, and a portable device such as a personal digital assistant 202 or a mobile phone handset. The devices and the media server 192 can communicate over a network 194 that is connected to the internet 204 via a gateway 206. In other embodiments, the media server 192 and the devices communicate over the internet.

[0040] The devices are configured with client applications that can request portions of media files from the media server 192 for playing. The client application can be implemented in software, in firmware, in hardware or in a combination of the above. In many embodiments, the device plays media from downloaded media files. In several embodiments, the device provides one or more outputs that enable another device to play the media. When the media file includes an index, a device configured with a client

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0471
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

application in accordance with an embodiment of the invention can use the index to determine the location of various portions of the media. Therefore, the index can be used to provide a user with "trick play" functions. When a user provides a "trick play" instruction, the device uses the index to determine the portion or portions of the media file that are required in order to execute the "trick play" function and requests those portions from the server. In a number of embodiments, the client application requests portions of the media file using a transport protocol that allows for downloading of specific byte ranges within the media file. One such protocol is the HTTP 1.1 protocol published by The Internet Society or BitTorrent available from www.bittorrent.org. In other embodiments, other protocols and/or mechanisms can be used to obtain specific portions of the media file from the media server.

[0041] Referring to FIGs. 8-11, one embodiment of a process of utilizing the index structure is shown. A media file, e.g., MFile 120, is received from, for example, a media server based on a media file request from a playback device or in particular a playback engine of the playback device (111). Upon locating the requested media file, the media server transmits all or some portions at a time of the media file to the playback device. The playback device in one embodiment decodes the transmitted media file to locate the hierarchical index (112). In one such embodiment, referring to FIG. 9, the playback device traverses or parses the file starting from EBML (Extensible Binary Meta Language) element 128, the Segment element 129 and then the contents of the Seek Head 121 to locate the Hierarchical Index 127. As such, the Segment information 122, Tracks 123, Chapters 124, Clusters 125 and Cues 126, although could be also parsed, can be bypassed to quickly locate the Hierarchical Index. The located index is then loaded into memory (113). Loading the index into memory facilitates access to locate a desired packet or frame to be displayed or accessed by the playback device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0472
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0042] The Hierarchical Index is small enough for many low memory playback devices, e.g., low level consumer electronic devices, to hold the entire Index in memory and thus avoiding a complex caching scheme. In cases, where the Index is too large to store in memory or generally more feasible, no loss in seek accuracy occurs. With the Index being a lookup table or mechanism into the cues or defined seek points for each of the tracks and not the actual seek points, the dropping of portions of the Index can cause a few additional reads when searching the cues for a desired seek point. The playback device accesses the bit stream packets or frames of the transmitted media file to play the audio, video, and/or subtitles of the media file [114].

[0043] Upon a user request, e.g., a trick-play request, the playback device searches the loaded or cached Hierarchical Index to find an entry or hierarchical point equal to or nearest and preceding to the desired time or seek point [115]. In one embodiment, the particular hierarchical point is located based on the presentation time or timestamp of the content being played and the user request, e.g., the speed and/or direction of trick-play function. In the illustrated case, FIGS. 10-11, the desired timestamp is 610 seconds within the bit stream.

[0044] FIG. 10 demonstrates a total of 6 hierarchical access points 130, starting from Index Time zero (Hierarchical Index Time 131) to Index Time 600 (Hierarchical Index Time 132), where five of the Hierarchical Points on this diagram have not been shown. After locating the closest hierarchical point to the desired seek time (in this case Index Time 600), an Index Position or offset 134 is retrieved from Track Position 133 to locate a portion of cues that contains the desired seek point [116]. The playback device seeks to the located portions of cues [117] and the cues are read through until an entry equal to, or nearest and preceding to the desired time or seek point is located [118].

[0045] Utilizing the located cue, the playback device retrieves an offset value to seek and find the desired cluster [119]. A block in the desired cluster that has a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0473
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

corresponding timestamp as the desired timestamp, e.g., 610, is located and decoded (120) for display by the playback device. The process continues until a user request stops playback of the media file.

[0046] This concept is further clarified in FIG. 11. The Hierarchical Index time of 600 is identified from the Hierarchical Index structure 127 as previously described in reference to FIG. 10. In this particular example, the Index position within the Cues structure 151 is used to access the particular Cue Point 152 which corresponds to time 610 (Cue Time 153). The Cue Point 152 through data in Track Position 154 and Cluster Position 155 generally points to the Cluster structure 160 which may host several seconds' worth of multimedia data.

[0047] The multimedia data within a Cluster 160 may be stored as a Block Group 163, where individual Blocks of data corresponding to one or more access units of the elementary audio, video, subtitle, or other multimedia information exist. As such, Clusters contain block groups but can also contain only simple blocks. In the absence of a Block Group, it may be possible that a Cluster can host individual Blocks or a Simple Block. The corresponding Cluster Position 155 from the Cue Point 152 is used to locate the Cluster 160 and the desired Block 161 can be identified based on its time stamp (Block Time 162). In case where an exact time stamp is not matched, the Block with the closest time stamp can be identified.

[0048] The procedure for locating a Block according to a particular time may be repeated for multiple tracks of multimedia data such that all of the data in the corresponding Blocks are presented in a synchronized manner.

[0049] While the above description contains many specific embodiments of the invention, these should not be construed as limitations on the scope of the invention, but rather as an example of one embodiment thereof. Accordingly, the scope of the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0474
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

invention should be determined not by the embodiments illustrated, but by the appended claims and their equivalents.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0475
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS CLAIMED IS:

1. A method of playing back content by a playback device stored in a media file, the method comprising:

providing a media file to a playback device, the media file having content data, cue data and a hierarchical index, the content data having a plurality of media frames, the cue data being associated with each frame of the plurality of frames and the hierarchical index being associated with a subset of the cue data;

decoding the content data by a playback device;

displaying content on a display screen from the decoded content data;

receiving a user request;

locating the index based on the user request; and

decoding one or more media frames from the subset of the cue data based on the located index.

2. The method of claim 1 wherein the index has reference values that point directly to particular media frames of the plurality of media frames.

3. The method of claim 1 wherein the hierarchical index includes pointers to the cue data within the media file and further comprising accessing the cue data through the pointers to locate a frame to be decoded.

4. The method of claim 1 wherein the hierarchical index is associated with a subset of the plurality of media frames through a second hierarchical index and further comprising locating a subset of the second hierarchical index based on the hierarchical index relative to the user request.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0476
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5. The method of claim 4 wherein decoding one or more media frames from the subset of the plurality of media frames is based on the located subset of the second hierarchical index.

6. The method of claim 1 wherein the media file has an upper hierarchical index associated with the hierarchical index and the hierarchical index associated with one or more media frames from the subset of the plurality of media frames and wherein the upper hierarchical index has a greater number of associations with one or more media frames than the hierarchical index.

7. A method of encoding a media file for playing back by a playback device, comprising:

incorporating seek locations with content data, the content data having audio, video and subtitle tracks, the seek locations identifying particular timestamps for particular frames within the audio, video or subtitle tracks;

incorporating an index referencing the incorporated seek locations; and

creating a media file having the content data, the incorporated seek locations, and the incorporated index.

8. The method of claim 7 wherein incorporating the index further comprises interleaving an index between audio and video tracks.

9. The method of claim 8 wherein the index points only to key frames within the audio, video or subtitle tracks.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0477
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

10. The method of claim 9 wherein the index contains locations and timestamps to key frames.

11. A system for playback of a media file, comprising:

a media server configured to transmit a media file having at least one index having pointers that reference at least one frame of media content within the media file; and

a client processor in network communication with the media server and configured to send requests for the media file to the media server, the media server configured to transmit the media file requested, the client processor comprises a playback engine configured to locate a pointer from the at least one index to locate and decode the portions of the media file based on the located pointer to comply with user playback instructions.

12. The system of claim 11 wherein the at least one index contains pointers to key frames only.

13. The system of claim 11 wherein the at least one index contains pointers to another index that contains pointers that reference at least one frame of media content within the media file.

14. The system of claim 11 wherein the at least one index comprises a plurality of pointers, each pointer referencing a frame of media content and the plurality of pointers in its entirety referencing each frame of media content.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0478
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

15. The system of claim 11 wherein the at least one index comprises a coarse index having pointers referencing frames of media content and a fine index having pointers referencing frame of the media content, the fine index having more pointers than the coarse index.

16. The system of claim 11 wherein the at least one index comprises a reduced index having pointers only to access frames of the media content.

17. A method of decoding a multimedia file having an index included with the multimedia file, the method comprising:

locating an index within a multimedia file, the index having a plurality of index data-points and the multimedia file having a plurality of frames and a plurality of cue data-points having a time stamp and a location of a specific frame of the plurality of frames;

loading the located index into storage;

locating an index data-point from the plurality of index data-points based on a timestamp, and

locating a cue data-point from the plurality of cue-data points based on the located index data-point.

18. The method of claim 17 further comprising displaying a video frame on a display screen, the displayed video frame being a frame that corresponds to a frame referenced in the located cue data-point.

19. The method of claim 18 further comprising playing audio through a speaker system in a synchronized manner relative to the displayed frame and the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0479
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

played audio being based on audio information that corresponds to a beginning of audio data for the displayed frame.

20. The method of claim 18 further comprising displaying subtitles in a synchronized manner relative to the displayed frame and the displayed subtitles being based on subtitle information that corresponds to subtitle data for the displayed frame.

21. The method of claim 18 further comprising displaying metadata information relative to the displayed frame and the displayed metadata information being based on metadata information that corresponds to metadata data for the displayed frame.

22. The method of claim 17 further comprising locating a cluster referenced by the located cue data-point.

23. The method of claim 19 further comprising locating a block within the located cluster and referenced by the located cue data-point.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0480
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

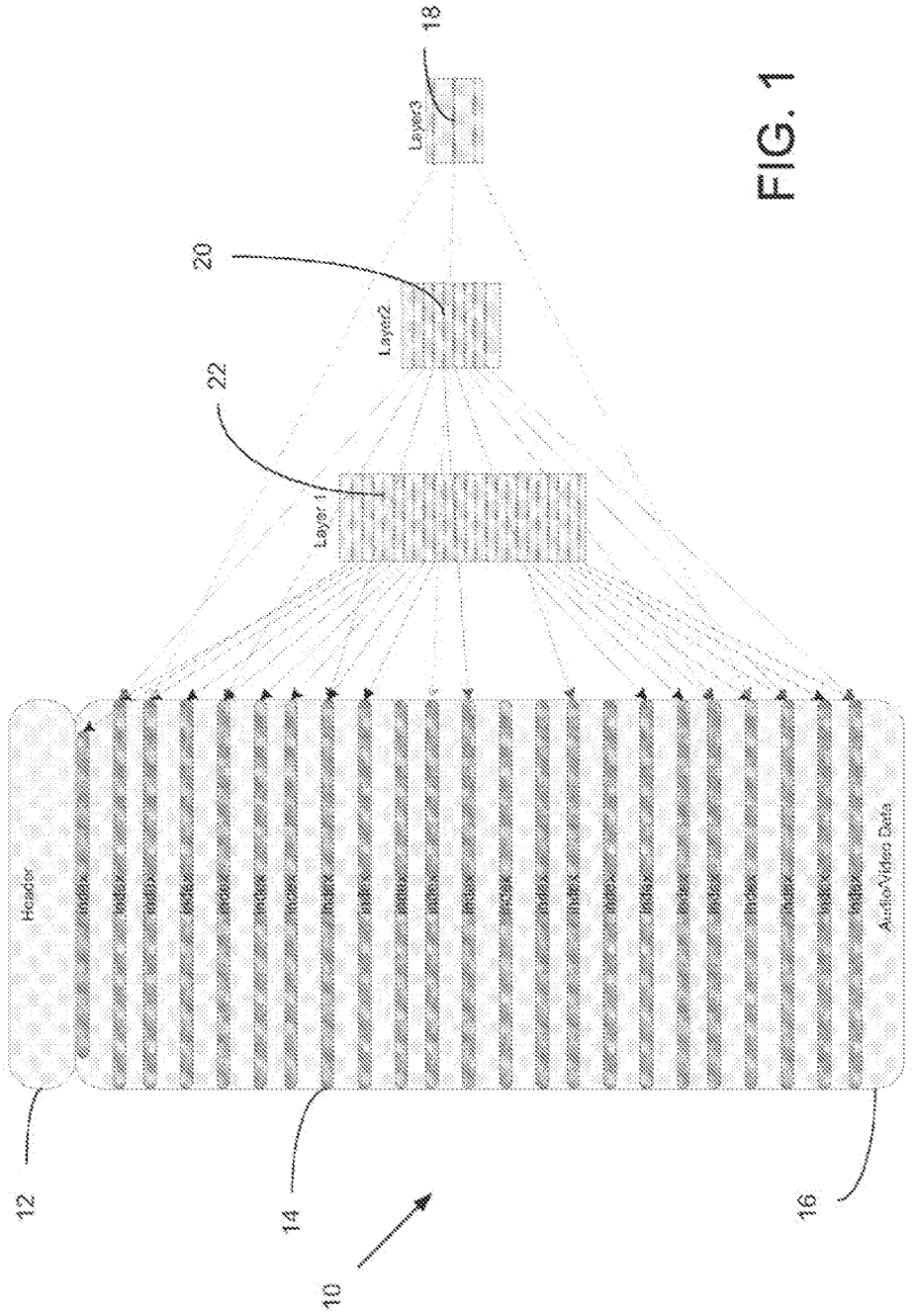

18
Layer3
20
Layer2
22
Layer1
Header
Audio+Video Data
12
14
16
10

FIG. 1

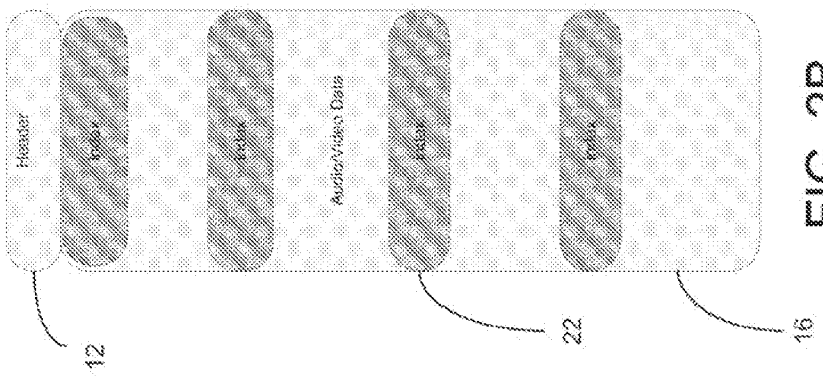

Header
Index
Index
Audio/Video Data
Index
Index
12
22
16

FIG. 2B

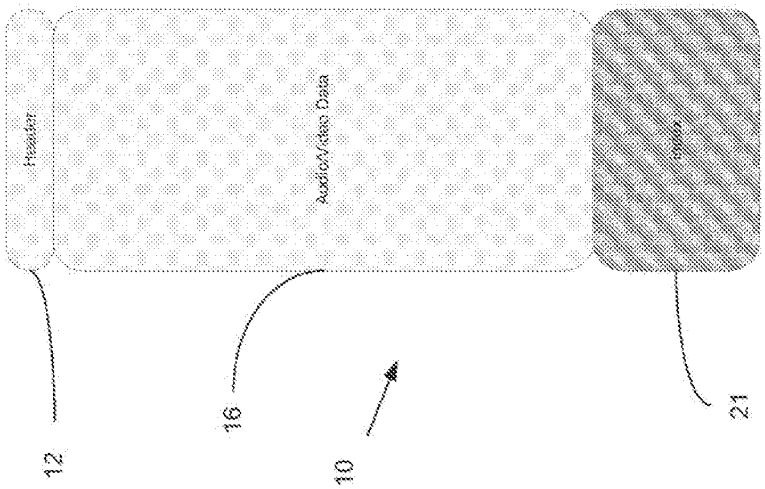

Header
Audio/Video Data
Index
12
16
10
21

FIG. 2A

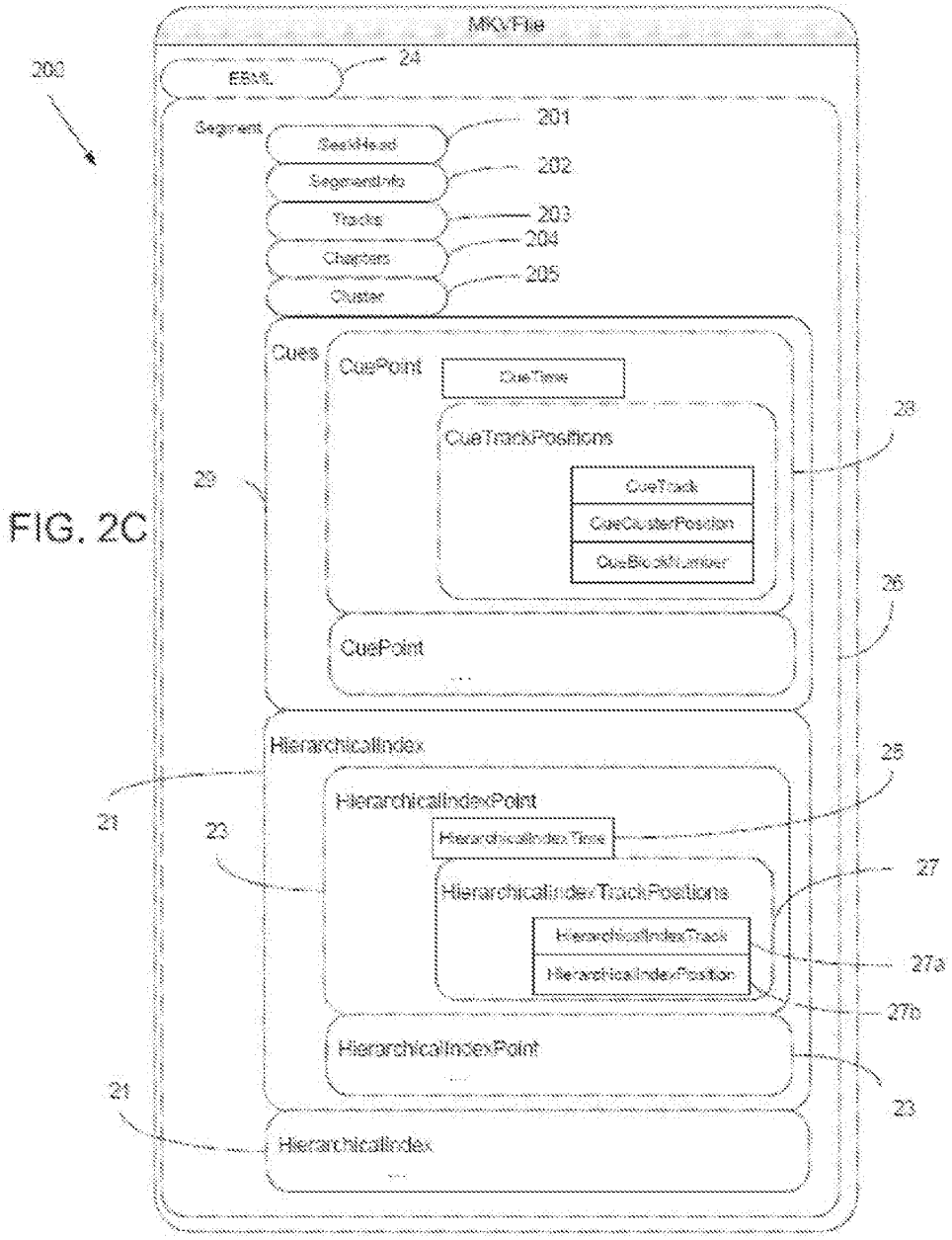

MKV File
200
EBML
24
Segment
SeekHead
201
SegmentInfo
202
Tracks
203
Chapters
204
Cluster
205
Cues
CuePoint
CueTime
CueTrackPositions
28
29
CueTrack
CueClusterPosition
CueBlockNumber
26
CuePoint
...
HierarchicalIndex
25
21
HierarchicalIndexPoint
HierarchicalIndexTime
23
27
HierarchicalIndexTrackPositions
HierarchicalIndexTrack
27a
HierarchicalIndexPosition
27b
HierarchicalIndexPoint
...
23
21
HierarchicalIndex
...

FIG. 2C

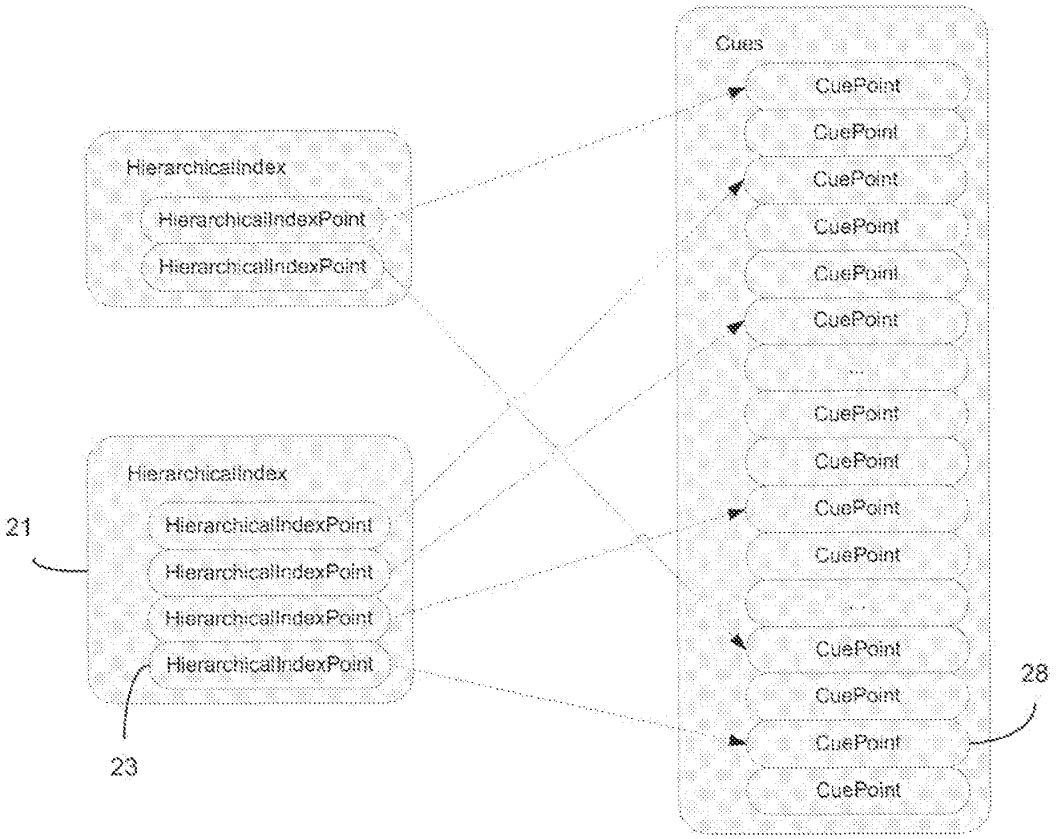

Cues
CuePoint
CuePoint
CuePoint
CuePoint
CuePoint
CuePoint
...
CuePoint
CuePoint
CuePoint
CuePoint
...
CuePoint
CuePoint
CuePoint
CuePoint
HierarchicalIndex
HierarchicalIndexPoint
HierarchicalIndexPoint
HierarchicalIndex
HierarchicalIndexPoint
HierarchicalIndexPoint
HierarchicalIndexPoint
HierarchicalIndexPoint
21
23
28

FIG. 2D

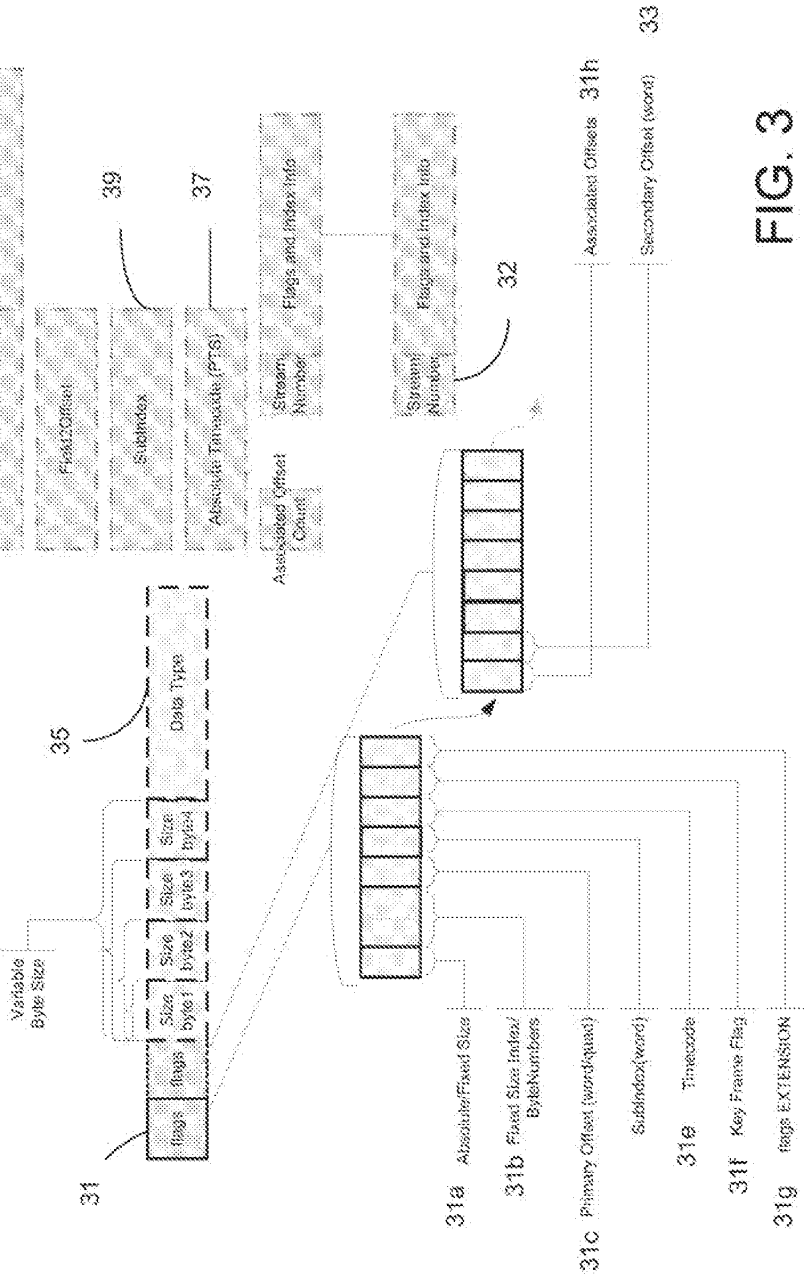

Offset (Word/Quad)
Field Offset
SubIndex
Absolute Timecode (PTS)
Associated Offset Count
Stream Number
Flags and Index Info
Stream Number
Flags and Index Info
Associated Offsets
Secondary Offset (word)
31h
33
32
39
37
35
31
Variable Byte Size
flags
Size byte1
Size byte2
Size byte3
Size byte4
Data Type
31a Absolute/Fixed Size
31b Fixed Size Index/ ByteNumbers
31c Primary Offset (word/quad)
SubIndex(word)
31e Timecode
31f Key Frame Flag
31g flags EXTENSION

FIG. 3

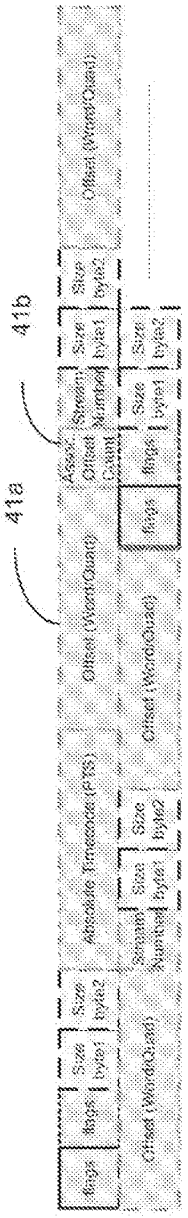

41a
41b
flags
Size byte1
Size byte2
Absolute Timecode (PTS)
Offset (WordQuad)
Assoc Offset Count
Stream Number
Stop byte1
Stop byte2
Offset (WordQuad)
Size byte1
Size byte2

FIG. 4

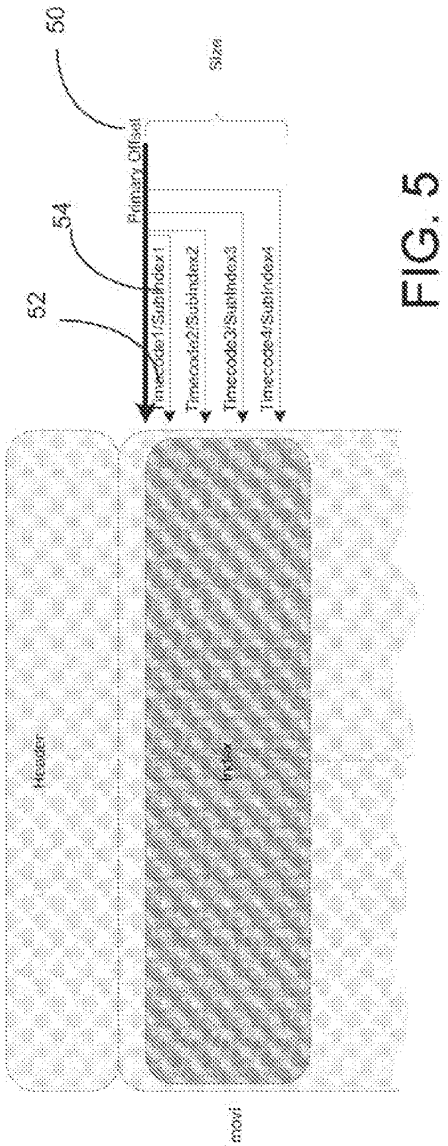

50
Size
Primary Offset
52
54
Timecode1/SubIndex1
Timecode2/SubIndex2
Timecode3/SubIndex3
Timecode4/SubIndex4
Header
Index
movi

FIG. 5

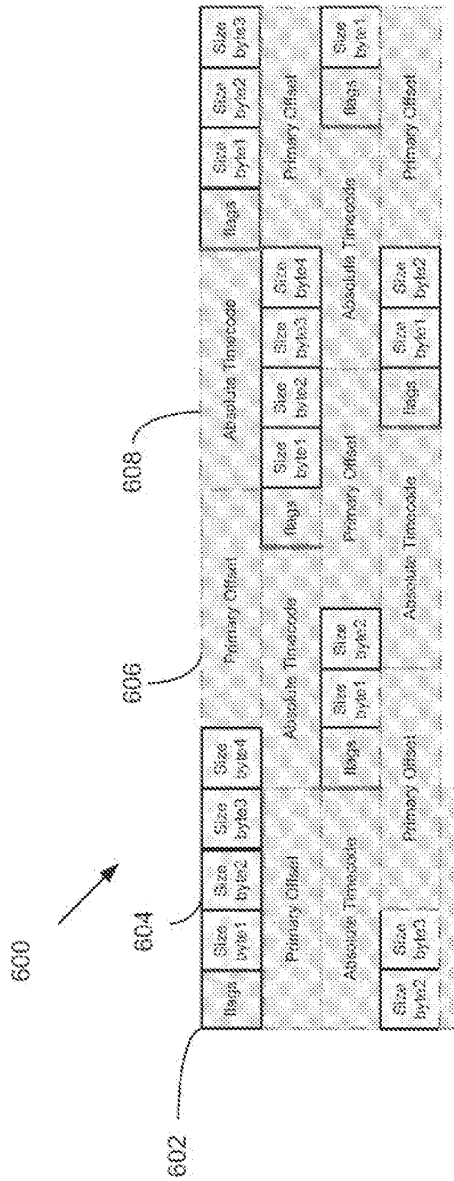

600
602
604
606
608
Flags
Size byte1
Size byte2
Size byte3
Size byte4
Primary Offset
Absolute Timecode

FIG. 6

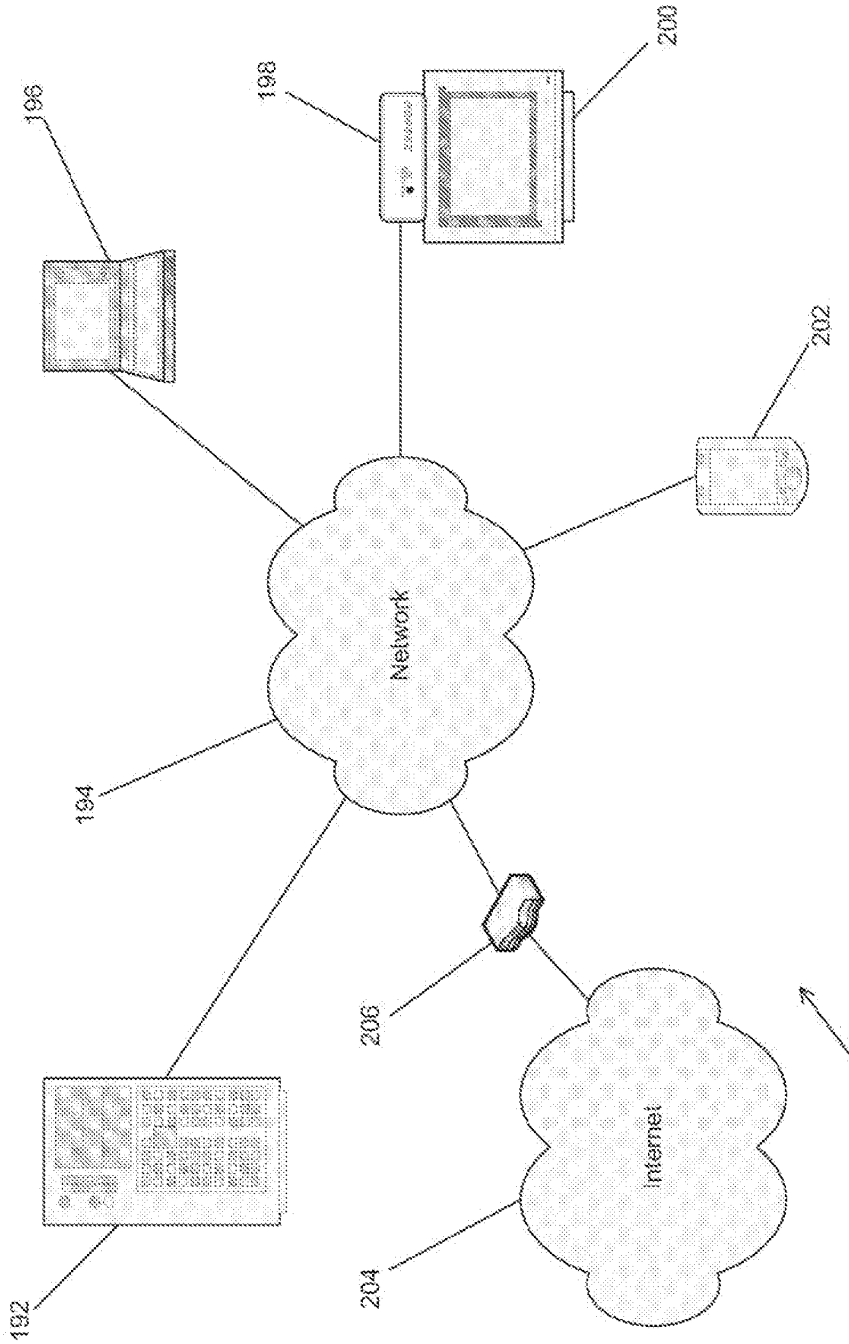


FIG. 7

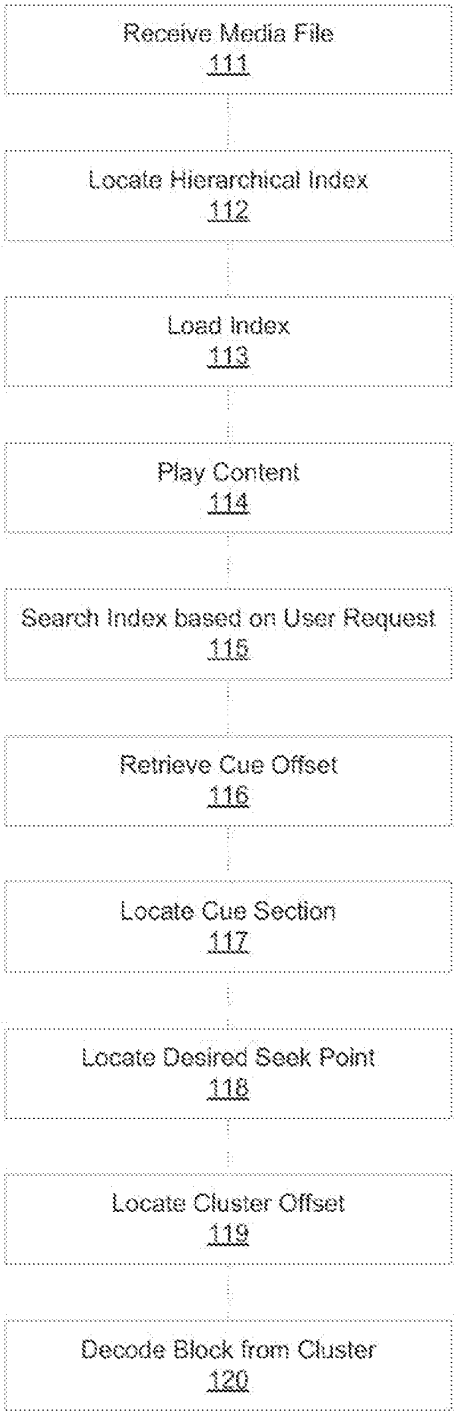

Receive Media File
111
Locate Hierarchical Index
112
Load Index
113
Play Content
114
Search Index based on User Request
115
Retrieve Cue Offset
116
Locate Cue Section
117
Locate Desired Seek Point
118
Locate Cluster Offset
119
Decode Block from Cluster
120

FIG. 8

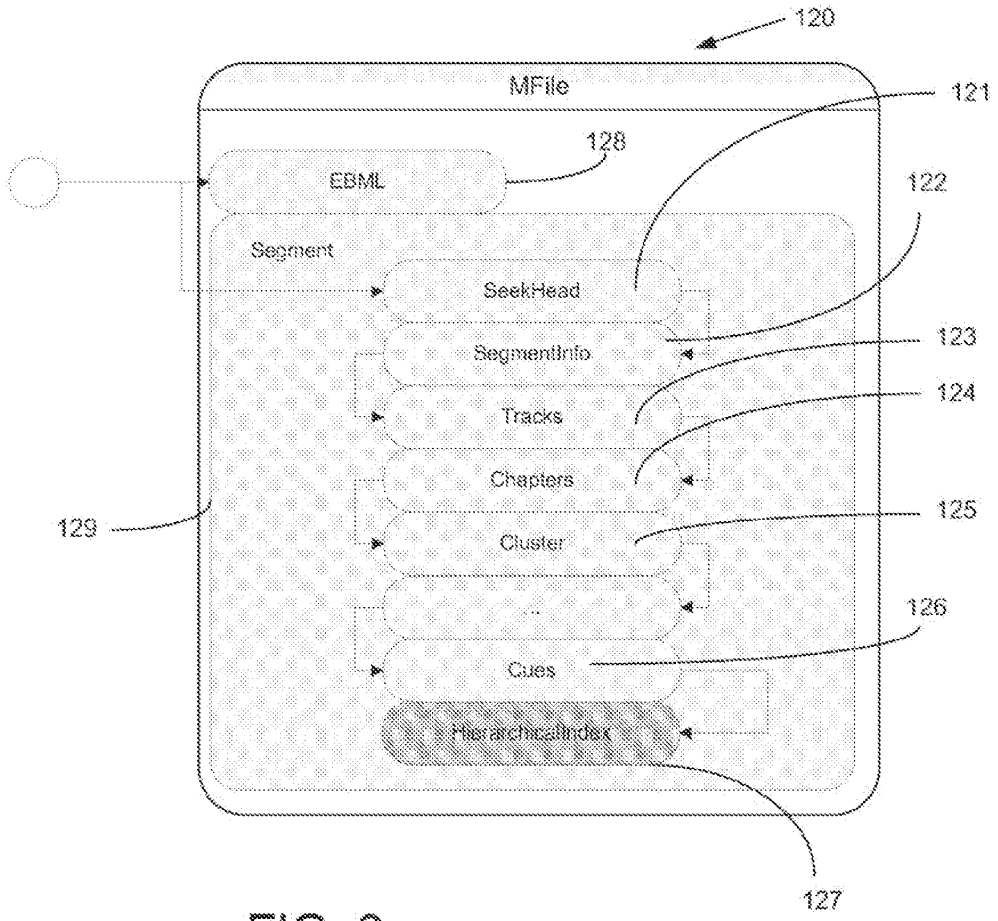

120
MFile
121
128
EBML
122
Segment
SeekHead
SegmentInfo
123
Tracks
124
Chapters
129
125
Cluster
..
126
Cues
HierarchicalIndex
127

FIG. 9

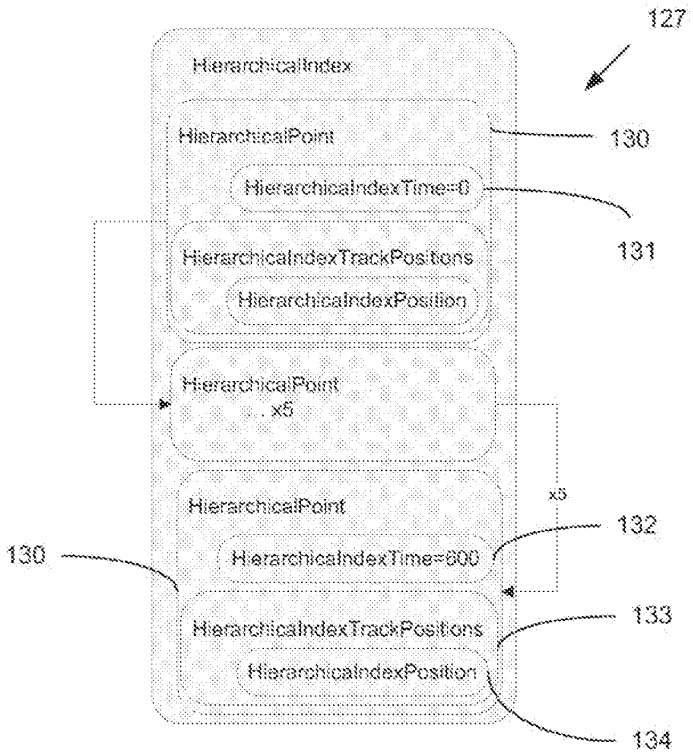
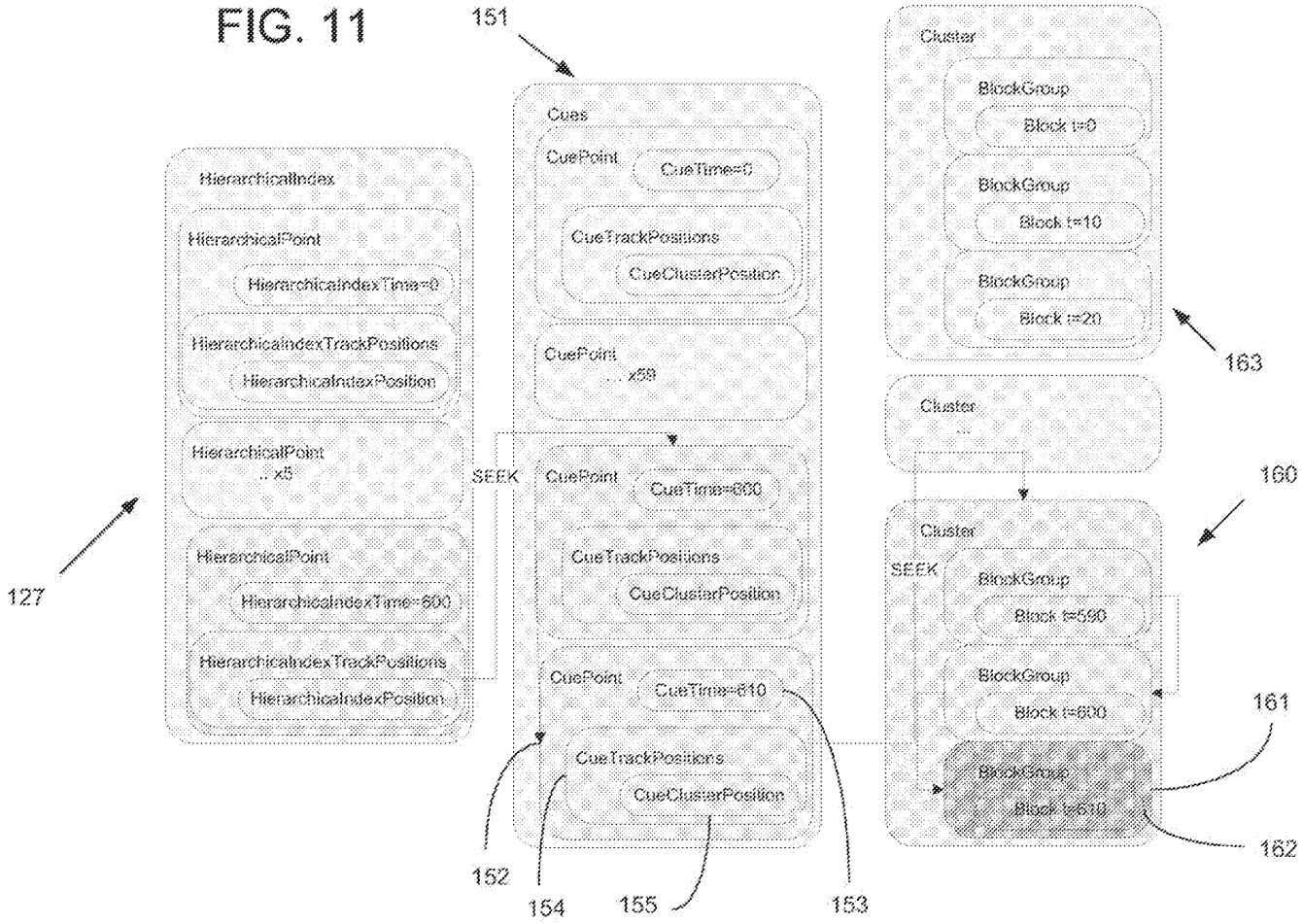

127
HierarchicalIndex
HierarchicalPoint
130
HierarchicalIndexTime=0
131
HierarchicalIndexTrackPositions
HierarchicalIndexPosition
HierarchicalPoint
. x5
x5
HierarchicalPoint
132
130
HierarchicalIndexTime=600
HierarchicalIndexTrackPositions
133
HierarchicalIndexPosition
134

FIG. 10

FIG. 11


151
Cues
CuePoint
CueTime=0
CueTrackPositions
CueClusterPosition
CuePoint
... x59
CuePoint
CueTime=600
CueTrackPositions
CueClusterPosition
CuePoint
CueTime=610
CueTrackPositions
CueClusterPosition
HierarchicalIndex
HierarchicalPoint
HierarchicalIndexTime=0
HierarchicalIndexTrackPositions
HierarchicalIndexPosition
HierarchicalPoint
... x5
HierarchicalPoint
HierarchicalIndexTime=600
HierarchicalIndexTrackPositions
HierarchicalIndexPosition
SEEK
127
Cluster
BlockGroup
Block t=0
BlockGroup
Block t=10
BlockGroup
Block t=20
163
Cluster
...
160
Cluster
SEEK
BlockGroup
Block t=590
BlockGroup
Block t=600
BlockGroup
Block t=610
161
162
152
154
155
153

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 08/83816

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 15/16 (2009.01)
USPC - 709/203
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
USPC - 709/203

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 709/201-202; 380/200,255

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
USPTO WEST (PGPB, USPT, EPAB, JPAB), GOOGLE
multimedia, content, cue data, index, reduced index, hierarchical index, hierarchical, play, trick play, playback

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/0235880 A1 (Qian) 19 October 2006 (19.10.2006), entire document, especially: Figs. 3, 4, 9, para [0017], [0019], [0020]-[0022], [0024], [0027], [0029], [0032], [0037], [0042], [0044]-[0046], [0051], [0053], [0054]	1-23

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 January 2009 (10.01.2009)	22 JAN 2009
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 May 2010 (27.05.2010)

PCT

(10) International Publication Number
WO 2010/060106 A1

(51) International Patent Classification:
H04L 12/26 (2006.01)

(21) International Application Number:
PCT/US2009/065805

(22) International Filing Date:
24 November 2009 (24.11.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/117,505 24 November 2008 (24.11.2008) US
61/161,376 18 March 2009 (18.03.2009) US

(71) Applicant *(for all designated States except US)*: **ANKEENA NETWORKS, INC.**, [US/US]; 4500 Great America Parkway, Suite 110, Santa Clara, California 95054 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **KOHLI, Jaspal** [US/US]; 1142 Shenandoah Drive, Sunnyvale, California 94087 (US). **NARAYANAN, Kumar** [US/US]; 956 Furlong Dr., San Jose, California 95123 (US). **SUNDARRAJAN, Prabakar** [US/US]; 4500 Great America Parkway, Suite 110, Santa Clara, California 95054 (US). **HE, Max** [CN/US]; 13961 Lacey Ave., Saratoga, California 95070 (US). **NISHIMOTO, Michael** [US/US]; 12686 Orella Ct., Saratoga, California 95070 (US). **VISHARAM, Mohammed Zubair** [IN/US]; 4500 Great America Parkway, Suite 110, Santa Clara, California 95054 (US). **VENKATARAGHAVAN, Vikram** [IN/US]; 4142 Tobin Circle, Santa Clara, California 95054 (US).

(74) Agents: **WONG, Kirk D.** et al.; 2055 Gateway Place, Suite 550, San Jose, California 95110 (US).

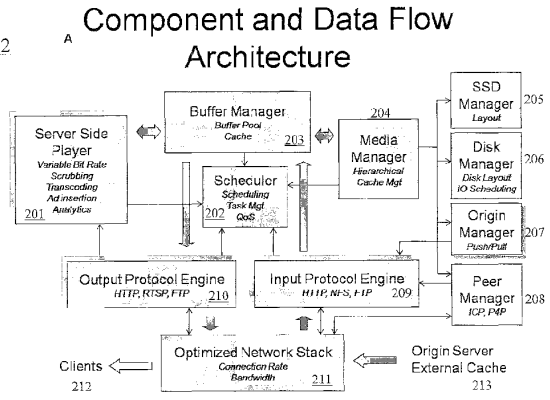
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

[Continued on next page]

(54) Title: ADAPTIVE NETWORK CONTENT DELIVERY SYSTEM

FIG. 2



A Architecture des composants et des flux de données
201 Lecteur du côté serveur
Débit binaire variable
Nettoyage
Conversion de codes
Insertion d'annonces
Analytique
202 Planificateur
Planification de la qualité de service du gestionnaire de tâches
203 Gestionnaire tampon
Pool tampon
Mémoire cache
204 Gestionnaire de contenu multimédia
Gestion de la mémoire cache hiérarchique
205 Gestionnaire SSD
Configuration
206 Gestionnaire de disque
Configuration du disque
Planification E/S
207 Gestionnaire d'origine
Pousser/tirer
208 Gestionnaire homologue
209 Moteur de protocole d'entrée
210 Moteur de protocole de sortie
211 Empilement réseau optimisé
Débit de connexion
Largeur de bande
213 Mémoire cache externe du serveur d'origine

(57) Abstract: A method and apparatus stores media content in a variety of storage devices, with at least a portion of the storage devices having different performance characteristics. The system can deliver media to a large number of clients while maintaining a high level of viewing experience for each client by automatically adapting the bit rate of a media being delivered to a client using the client's last mile bit rate variation. The system provides clients with smooth viewing of video without buffering stops. The client does not need a custom video content player to communicate with the system.

WO 2010/060106 A1

GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

ADAPTIVE NETWORK CONTENT DELIVERY SYSTEM

TECHNICAL FIELD

[0001] The present disclosure generally relates to media content delivery.

BACKGROUND

[0002] The approaches described in this section could be pursued, but are not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated herein, the approaches described in this section are not prior art to the claims in this application and are not admitted to be prior art by inclusion in this section.

[0003] Delivery of content across the Internet has been accomplished in several different ways. Content servers have been used to deliver content directly to requesting clients. As the content servers became overloaded from the traffic demands, mirroring of the content across multiple servers were implemented. Mirror servers were added as client traffic increased beyond the capacity of the installed base of mirror servers.

[0004] As mirroring became costly to the content providers, caching content in content delivery networks (CDN) became popular. The content provider's content was distributed among caching servers across the Internet. The caching servers were shared among multiple content providers and operated by a CDN provider. The caching servers were essentially file servers that served different types of files to client systems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0006] FIG. 1 illustrates a media flow director (MFD) in communication with client systems, an origin server, a peer MFD, and a central management station across a network, according to an embodiment of the invention;

[0007] FIG. 2 illustrates an example of an MFD component and data flow architecture, according to an embodiment of the invention;

[0008] FIG. 3 illustrates an example of dynamically determining the popularity of portions of content, according to an embodiment of the invention;

[0009] FIG. 4 illustrates a MFD deployment that replaces a conventional video server site, according to an embodiment of the invention;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0496
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0010] FIG. 5 illustrates an MFD network delivering content to client systems, according to an embodiment of the invention;

[0011] FIG. 6 illustrates a computer system upon which an embodiment may be implemented;

[0012] FIG. 7 illustrates flow diagram of the workflow between the HPE and the server side player, according to an embodiment of the invention;

[0013] FIG. 8a illustrates the interaction between a disk manager and a policy module and an analytics module, according to an embodiment of the invention;

[0014] FIG. 8b illustrates the interaction between a disk manager and dictionaries for storage devices, according to an embodiment of the invention;

[0015] FIG. 8c illustrates the interaction between a disk manager and a reverse block map, according to an embodiment of the invention;

[0016] FIG. 9 illustrates the interaction between a buffer pool and buffer manager, according to an embodiment of the invention;

[0017] FIG. 10a illustrates the interaction between a media manager, buffer manager, and media providers, according to an embodiment of the invention;

[0018] FIG. 10b illustrates the interaction between a media manager and analytics module for determining hot video segments, according to an embodiment of the invention;

[0019] FIG. 11 illustrates a flow chart of a network manager in a MFD, according to an embodiment of the invention;

[0020] FIG. 12 illustrates a progressive download hint track location in a container file, according to an embodiment of the invention;

[0021] FIG. 13 illustrates a graphical user interface screenshot of a parameter specification screen, according to an embodiment of the invention;

[0022] FIG. 14 illustrates a graphical user interface screenshot of a real time byte delivery monitoring graph, according to an embodiment of the invention; and

[0023] FIG. 15 illustrates a graphical user interface screenshot of a network connection performance monitoring graph, according to an embodiment of the invention.

DETAILED DESCRIPTION

[0024] An adaptive network content delivery system is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0497
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

[0025] Embodiments are described herein according to the following outline:

1.0 General Overview
2.0 Structural and Functional Overview
3.0 Adaptive Content Delivery over a Network
3.1 Media Flow Director Architecture
3.2 Media Flow Director Placement
4.0 Implementation Mechanisms—Hardware Overview
5.0 Examples
6.0 Extensions and Alternatives

1.0 GENERAL OVERVIEW

[0026] A method for an adaptive network content delivery system is described. In other embodiments, the invention encompasses a computer apparatus and a computer-readable medium configured to carry out the foregoing steps.

2.0 STRUCTURAL AND FUNCTIONAL OVERVIEW

[0027] An adaptive network content delivery system is described. Referring to FIG. 1, a possible embodiment provides a media flow director (MFD) 103. A MFD is a network device that can efficiently deliver video content (and other types of file content of various sizes, e.g., media (audio, pictures, etc.), games, software, HTML, scripts, etc.) over a network 104 to a plurality of clients 101a, 101b, 101n, via HTTP, FTP, streaming, or other protocols. Video content includes such content as feature length movies, sitcoms, variety shows, talk shows, news shows, advertisements, etc. Client devices 101a, 101b, 101n, may be a personal computing device such as a desktop or laptop computer, set top box, handheld computing device, cellular phone, portable media player, or any other portable device capable of displaying multimedia content, and are also coupled to network 104 through any proper interface.

[0028] The MFD 103 stores video content internally in a variety of storage devices, including HDD, SSD, RAM, non-volatile memory, etc. The MFD 103 can deliver video to a large number of clients while maintaining a high level of viewing experience for each client. The MFD 103 automatically adapts the bit rate of a video being delivered to a client 101 by measuring the client's last mile bit rate variation. The MFD provides the clients with smooth viewing of video without buffering stops. It assures delivery of in-progress videos to clients

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0498
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

using the adaptive bit rate delivery. The MFD dynamically adapts the bit rate of the video to match the effective last mile bandwidth and to adapt to changing network conditions for the client. The MFD can dynamically measure the last mile bandwidth to the client in order to dynamically adjust the bit rate. The client does not need a custom video content player to communicate with the MFD. The MFD supports industry standard video players such as Flash, Quicktime, SilverLight, Windows Media Player, etc.

[0029] The MFD 103 provides policy controls on the storage and delivery of videos that can be defined by a customer as well as the MFD provider. The MFD can be deployed in a variety of ways across the network 104, including at an edge cache location or an origin server location. The network 104 may be implemented by any medium or mechanism that provides for the exchange of data between devices in the communication system. Examples of network 104 include, without limitation, a network such as a Local Area Network (LAN), Wide Area Network (WAN), Ethernet, intranet, the Internet, or one or more terrestrial, satellite or wireless links.

[0030] The MFD 103 communicates with network 104 through any proper communication interface, such as an Ethernet or wireless communications port. Alternatively or in addition, any number of devices connected to network 104 may also be directly connected to each other through a communications link. For example, the MFD 103 can request content from an origin server 102 via the network 104 or via a local network in order to obtain URL content when a client 101 requests a URL content that the MFD 103 does not have stored in its memory. Additionally, MFD 103 can communicate with another MFD 105 via the network 104 or via a local network in order to obtain URL content, MFD content information, MFD status, etc.

[0031] A central management station 106 may be available to allow administrators to set policies for MFD management such as cache management, bandwidth limitations, content popularity determination, etc. The central management station 106 also allows administrators to manage where and how content is stored in the multiple storage devices within a MFD. The central management station 106 may communicate via a local network or connection to MFDs 103, 105 or via network 104. An administrator logs into the central management system 106 and selects a MFD or set of MFDs that the administrator wants to interact with. The administrator can define policies and have the policies sent to a MFD or pushed out to multiple MFDs by the central management station 106. Alternatively, the MFDs may contact the central management station 106 periodically or when notified that by the central management station that updates are available for that MFD.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0499
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

3.0 ADAPTIVE CONTENT DELIVERY OVER A NETWORK

[0032] 3.1 MEDIA FLOW DIRECTOR ARCHITECTURE

[0033] FIG. 2 illustrates a possible embodiment showing the architecture of a content delivery switch that is configured as a MFD to deliver video content. The components in the architecture are described below.

Output Protocol Engine 210

[0034] The output protocol engine 210 handles requests from clients. A client sends a request to the MFD to make a connection with client, e.g., an HTTP get request for a particular URL, the output protocol engine 210 passes for requests URL content to the server side player 201.

Server Side Player 201

[0035] The server side player 201 is primarily responsible for the encapsulation and enforcement of video specific logic for each video stream. Such an enforcement could happen both at the inbound side when a request is made to the MFD for a video stream or at the outbound side when a specific video stream is heading out for delivery.

[0036] The server side player 201 interprets a URL received from the output protocol engine 210, decides what bit rate is to be used for the requested video content, handles what is to be sent, and where to play the video from (what frame within the video to start streaming from). The server side player 201 is bit rate aware and knows the bit rate of the video content that it is trying to serve. It can use information obtained from the optimized network stack 211 to find out what the effective bandwidth is for the client in the last mile. The server side player 201 can also authorize the request. In a possible embodiment, a signed hash key embedded as a query string in the URL is used to verify that the request is authorized.

[0037] The server side player 201 provides an assured bit rate and quality of service (QoS) feature. If the client has to buffer, then it means that the MFD has fallen behind in its delivery rate and the client has experienced a dead period where there is no content to play or display. In one possible embodiment, the server side player 201 calculates how much data has been sent to the client and how much time has passed. The server side player 201 tracks the amount of data sent to each client and the bit rate of the video content that each client is being sent. As each client is sent an amount of data, the server side player 201 records the amount of data that was sent and the amount of time that has passed since the last amount of data was sent. Alternatively, the server side player 201 can record the total amount of data sent to the client and the total amount of time that has passed during the client's session.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0500
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Using the recorded information, the server side player 201 can calculate if it has fallen behind in its delivery to a particular client. For example, if the server side player 201 has sent 30 seconds of data at a certain bit rate to the client and 31 seconds have elapsed since the last transmission of data, the server side player 201 knows that it has fallen behind.

[0038] In another possible embodiment, the server side player 201 calculates how much more data it needs to deliver to the client and the time window that the data must be delivered in to avoid buffering on the client side. This calculation is slightly different than the previous example. The server side player 201 knows the size of the video file and thus can calculate the amount of data that remains to be sent to the client. It can also calculate the amount of time it has to deliver the remaining amount of data given the bit rate and the total size of the video file. The server side player 201 can calculate the deadline that it needs to set for a task that it creates to deliver a segment of the video file to the client.

[0039] The MFD is a task-based system that can use deadline-based scheduling. The server side player 201 sets up deadlines in the tasks that it creates in order to avoid client buffering.

[0040] In a possible embodiment, the server side player 201 calculates an assured flow rate (AFR) value (described below). The AFR is the transmission rate required to ensure a smooth viewing experience. The server side player 201 uses the AFR value to set deadlines for tasks that it creates for requesting data from the buffer manager 203. The server side player 201 calculates the AFR and the output protocol engine 210 manages the transmission rate on each connection based on the AFR and sets the task deadlines such that the AFR can be met. In the example above, the server side player 201 tries to stay ahead of the AFR and would know that it has fallen behind if, for example, the server side player 201 has sent 30 seconds of data and 30 seconds have elapsed since the last transmission, the server side player 201 immediately knows that it has fallen behind. In an alternative embodiment, the server side player 201 can scan a media file to set an AFR for the media file. The server side player 201 knows that a certain quality or resolution of the media file requires a certain transmission rate in order to ensure that the client will not stall.

[0041] The server side player 201 also handles scrubbing requests (in order to decide what to do in the future), transcoding, and ad insertion. The server side player 201 is video-aware as opposed to a typical file server that does not know what the video file format or video content is. When the server side player 201 changes the bit rate after the video has started, the server side player 201 already knows where the associated frame exists in the different bit rate version, so the server side player 201 can easily change bit rates without the client having to send a different offset for the new bit rate. A possible embodiment stores

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0501
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

video content in a custom format that can be transcoded to any bit rate that it requires. This allows the server side player 201 to know exactly where the associated frame is for each bit rate transcoding. Another possible embodiment stores several different bit rate versions for a specific video content. The server side player 201 stores index tables for each of the bit rate versions so the server side player 201 can quickly refer to a table to find the associated frame for the new bit rate video.

[0042] The server side player 201 can perform advertisement insertion into the video streams. Ads can be placed in the video stream as overlays. This makes the ad insertion independent of the client player. The content can be dynamically created, e.g., subtitle language selection can be made on the fly and the appropriate subtitles can be overlaid on the fly. The server side player 201 can splice an ad into the middle of a video (referred to as mid-roll). The server side player 201 can also overwrite the voice channel of a video with another channel to provide multiple language support without having to store different versions of the video content.

[0043] In a possible embodiment, the server side player 201 provides two interfaces, both of which are accessed by the HTTP protocol engine (HPE) in the input protocol engine 209:

[0044] northbound interface: This serves as the northbound interface where every incoming request parsed by the HPE is examined by the server side player 201 to validate the request with its customer specific logic. Once the request is examined and the necessary parameters are enforced, the request is sent back to the HPE so that it can make a valid request to the buffer manager 203 (and other providers thereafter) for the content.

[0045] southbound interface: This serves as the southbound interface where all outgoing data is sent to server side player 201 to be examined before delivery. This is the final place where the server side player 201 can enforce any restrictions before the content is sent out for delivery. The server side player 201 has the ability at this stage to modify the data buffers returned by the buffer manager 203 if required before sending it out.

[0046] Fig. 7 shows a possible embodiment of the workflow between the HPE and the server side player 201. The following steps are performed:

[0047] Step 1: User request comes to the HPE 701 through the optimized network stack 211 after a connection has been established.

[0048] Step 2: HPE 701 does minimal parsing to get the query string and name/value pairs.

[0049] Step 3: HPE 701 then initiates a call to the server side player 201 with the query string.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0502
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0050] Step 3a: Server side player 201 may act as a pass through for files that are not related to video, but per se security access files, etc such as those requested by client players.

[0051] Step 4: For every first call per new connection, server side player 201 remaps the query string to an internal URL pointing to a metafile. This metafile stores the number of profiles present, their rates, time period, and other information of the video.

[0052] Step 5: Server side player 201 returns the remapped URL to HPE 701 and also sets a flag that indicates to HPE 701 to not send the returned metafile on the wire, but back to the server side player 201 for further processing.

[0053] Step 6: HPE 701 creates a task to fetch the metafile from cache. HPE 701 receives the metafile and passes it back to server side player 201.

[0054] Step 6a: The server side player 201 stores the information in the metafile in the socket structure for that connection. The metafile information is stored in the cache as a result of this fetch for fast lookup in the future.

[0055] Step 7: Once the socket structure is populated, server side player 201 can now remap the original URL with the translation logic plugin 704. The translation logic plugin 704 (does both seek and rate adaptation) uses the information present in the server side player 201 structure in the socket connection to do the necessary translation to different profiles/chunks.

[0056] Step 8: Server side player 201 returns the remapped URL and resets the flag in HPE 701 to convey that it is ok to fetch and send the data retrieved from the cache on the wire.

[0057] Step 9: The query string is passed to the server side player 201. Server side player 201 checks if the socket structure is already populated.

[0058] Step 10: If populated and if the name/value pair for time offset is not minimal (time offsets or byte offsets that denote a scrub or seek are present), then server side player 201 calls translation logic to remap to an appropriate URL and server side player 201 calls HPE 701 to fetch/pass data.

[0059] Step 11: If name/value pair in query string is minimal (no time offsets or byte offsets that denote a scrub or seek), then repeat from Step 4.

Scheduler 202

[0060] The scheduler 202 uses deadlines set by callers to organize scheduling of tasks. There can be hundreds of thousands of tasks in that the scheduler 202 is managing. It keeps the tasks organized by the deadlines. The scheduler 202 monitors the tasks to see if any tasks have missed their deadline. This tells the scheduler 202 that the MFD has fallen behind. It

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0503
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

sends feedback to the server side player 201 to tell it that the delivery rate has fallen behind. The scheduler 202 can send information such as task ID and amount of time that the task missed the deadline.

[0061] In a possible embodiment, the scheduler 202 may refuse to accept a task for a new connection if it determines that it cannot meet the requested task's deadline. This is one component of admission control that ensures that MFD resources are not over-committed and existing connections play smoothly. Similarly, all components that implement tasks (e.g. buffer manager, disk manager and origin manager) can look at their internal resource commitments and fail the first task of a new connection as a way of refusing to accept the session.

[0062] The server side player 201 can identify which tasks are associated with each client. It can calculate, possibly using the above described calculations, how late it is in delivering video content to a client and adjust deadlines for subsequent tasks accordingly. The server side player 201 can then stop accepting new connections. In a possible embodiment, the server side player 201 can, possibly based on policy settings, lower the bit rate of some of the existing connections in order to accept additional connections. These approaches allow the server side player 201 to dynamically measure its performance and thus it does not need to set a static cap on the number of client connections. Given that it is handling many different bit rates, a static cap on the number of connections does not represent the actual capacity of the MFD, so a dynamic cap can be more efficient and allows the MFD to decide on its own whether it can handle more or less connections.

[0063] A possible embodiment performs deadline scheduling where each task state can have multiple deadline queues. Each deadline queue can correspond to fixed time periods, for example, 1/10ms chunks. The scheduler 202 can pop the queue that has the shortest deadline.

[0064] Task handlers are pre-registered with the scheduler 202 at init time. They are always non-blocking. Blocking handlers are another set of threads and are described below. In a possible embodiment, any task switching does not involve a thread context switch.

[0065] Blocking tasks are given off to a pool of threads. Whenever there is a task to be completed, one thread from the thread pool services the task. The scheduler 202 at this point has handed off the blocking task. Hence, no other tasks will be blocked.

[0066] The scheduler checks relevant queues (state/deadline queues) in a WRR (Weighted Round Robin) fashion. It dispatches tasks by calling pre-registered task handlers. The scheduler 202 schedules according to a deadline by keeping track of elapsed time and popping relevant queues. The deadline scheduling is not strictly real time. The deadlines are

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0504
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

a recommendation to the scheduler 202 that a particular task needs to be scheduled before another.

[0067] The scheduler 202 enables different types of threads. Threads are task threads, I/O threads, event threads, and network threads. These enumerated threads do not limit the number and types of threads that can be implemented. Task threads are the primary engines for task execution. The scheduler 202 is the start routine. It implements a dispatch look based on priority management. These threads block when there are no tasks to run.

[0068] I/O threads are used to implement blocking threads (e.g., disk I/O in the disk manager 206). They can block on read/write calls. Event threads are asynchronous and map network events to task state changes. Network threads implement the user space network stack and typically do not block on reads. Dedicated threads are used for scaling and management of timers for AFR. Alternatively, the network activity could be performed using tasks on the scheduler threads.

Disk Manager 206

[0069] The disk manager 206 manages objects spread over multiple disk drives, regardless of size and count and type. Also referring to Fig. 8a, disk manager 206 obtains hints about object placement from the policy modules 801 (PM) and analytics module 802 (AM). The media manager 204 can interact with the PM 801 to help it decide how to apportion videos into segments and also to determine when a portion of a video is determined as hot (discussed below). The media manager 204 can also interact with the AM 802. The AM 802 determines which video portions are hot based on the statistical information covering video portion hits provided by the media manager 204.

[0070] Objects stored within disk manager 206 are most likely video related but can be anything. The disk layout of objects is particularly suited for video objects because they are consumed in a serial fashion. For small objects, disk manager 206 performs special packing so many sequential, small objects can be packed closely together in time sequence order. This works very well in the case where the small objects are chunked pieces of a larger video in time sequence. This packing allows disk manager 206 to optimize read performance of disk drives because the time it takes to read a small amount of data is about the same as the time it takes to read about 1-2MB of data. In another embodiment, the disk manager 206 can allow for multiple strategies for packing objects. In addition to time based locality, it can also support packing based on locality within the name structure or other custom schemes.

[0071] Referring also to Fig. 8c, the disk manager 206 can use a reverse block map (RBM) 805 in order to consistently read large blocks 806 (even in the presence of small

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0505
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

objects). The block sizes are set to a specific size that allows for efficient reads from the disk drives, e.g., 1MB, 2MB, etc. For each block, the disk manager 206 keeps a list of extents in the RBM 805, where each extent is a contiguous portion of an object. In this example, the RBM 805 lists block 806 with related extents 807-812. Note that for smaller objects, multiple small objects themselves or portions of the smaller objects may fit into a block and may be read into cache memory together. When the disk manager 206 receives a request for a specific portion of the object, it locates the block where the specific portion resides. It then reads the entire block into cache memory 813 supplied by the buffer manager 203. Using the RBM 805, the disk manager 206 is able to assign the correct identity (UOL) to each buffer (as described below). This allows the buffer manager 203 to subsequently consume the additional data without doing any additional IO. In an embodiment, the disk manager 206 creates entries in the RBM for all of the objects stored in the MDF. The disk manager 206 may create an entry in the RBM for an object when the object is loaded from an origin server.

[0072] The disk manager 206 accounts for many types of disk controllers and disk drives, including SATA, SAS, SSD and flash. Each drive type has different size and performance characteristics. As such, the disk manager 206 will vary the data layout for each drive type to allow optimal performance and utilization of disk space. In order to self-tune for variances in drive types and controllers, the disk manager 206 performs a brief bandwidth test on startup, so it can categorize the large number of configurations that it sees within an open hardware framework.

[0073] An administrator can configure the disk manager 206 to store content from an origin server onto the hard drive or in a cache. The administrator can specify that the disk manager 206 store the content on the first hit or any specific number of hits before the disk manager 206 stores the content on the hard drive or cache. This allows the administrator the flexibility to adjust the cached content to any known or predicted demands. Alternatively, how the disk manager 206 stores the content can be driven by internal analytics. The administrator can instruct the disk manager 206 to store specific new content on the hard drive because, for example, he knows that new episodes of a show will be in demand.

[0074] Eviction policies are enforced by the disk manager 206. Content can be removed from the hard drive or cache in order to make room for new content based on eviction policies defined by an administrator or customer. In a possible embodiment, eviction policies can be internally derived based on analytical algorithms, e.g., using a least recently used (LRU) algorithm or its derivatives. The customer can install enough storage capacity in the MFD such that the storage space may never be filled by the customer's content. The customer may desire to have its content in the fastest areas of the hard drives and sparsely

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0506
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

populate the hard drives in order to accomplish this goal. The customer can then have a set of MFDs dedicated to their hottest content with sparsely populated hard drives and have another set of MFDs caching their less-popular content. The administrator can specify that certain hard drives on a MFD are kept sparsely populated with the hottest content and other hard drives on the MFD are used for all other content. This allows a customer to implement an efficient storage scheme for a particular MFD.

[0075] Referring also to Fig. 8b, the disk manager 206 treats each device as an independent cache module. The meta-data and data on each device comprise a separate and independent store, similar to a filesystem. However, the naming hierarchy is common across all modules, unlike a traditional filesystem, where a given name is bound to a logical device (filesystem). For example, if a traditional filesystem were mounted at /a/b/c, then all objects under that directory would be located in that filesystem. By contrast, the disk manager 206 is able to store /a/b/c/one.gif, /a/b/c/two.gif, and /a/b/c/three.gif in any of the cache devices 803a, 803b, 803c, it manages. During initial placement (as well as relocation for load balancing), the disk manager 206 selects the device to store the object based on usage statistics it keeps on a per device basis. During lookup (to serve the STAT task), the disk manager 206 determines which devices have the object and selects one, again based on usage statistics. This allows objects, or portions thereof, to be placed and served from any cache module independent of its name. The disk manager 206 is able to optimize the bandwidth from each device by placing objects (and relocating them dynamically) based on usage statistics. In an embodiment, the disk manager 206 keeps track of per device traffic in terms of read and write throughput for each device. During initial placement and relocation, it chooses the device with the least amount of traffic. In another embodiment, it uses the queue depth (number of outstanding requests) and average latency as the measure of traffic. The flexible placement results in linear scaling across cache devices.

[0076] In an embodiment, the disk manager 206 caches the dictionary 804a, 804b, 804c, for each cache device 803a, 803b, 803c, in RAM. The dictionary 804a, 804b, 804c, consists of the tree of objects stored in that device 803a, 803b, 803c, along with a small amount of metadata about the object. This allows the disk manager 206 to answer the STAT query very efficiently and without incurring any disk IO.

[0077] The disk manager 206 provides a simple fault recovery model. The failure of a single cache device just results in the loss of the objects contained in that cache device. It has no impact on the performance of the other cache devices. The failed cache device can be replaced (online, for a hardware system that supports hot swapping) and reconfigured as a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0507
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

new cache module. In an embodiment, the disk manager 206 can also selectively replicate objects in multiple cache devices in order to get better fault recovery and performance.

[0078] In an embodiment, all information about videos is self-contained within each drive such that drives can be moved between MFDs. Over time, disk manager 206 queries the analytics module 802 to atomically move entire movies or clips between drives and/or to inner or outer tracks in drives in the SAS disks 803. Disk manager 206 provides data necessary for the media manager 204 to promote and demote objects between various layers within the cache hierarchy. Initial placement of content can be dependent on data from the analytics module 802 and/or the policy module 801. The disk manager 206 chooses the data layout that optimizes read performance, even for small objects.

[0079] The disk manager 206 can avoid fragmentation problems that affect performance for most filesystems. It does so by reclaiming entire blocks whenever a configurable amount of space has been freed in that block. For example, if a 2MB block is filled with 64 objects of 32KB each and half the objects are deleted, e.g., based on eviction, the disk manager 206 can free the entire block. The RBM is used by the disk manager 206 to logically delete or relocate the remaining objects.

[0080] Video information can include: compressed/uncompressed video data, video attributes needed by an application (header information), and video attributes needed to aid internal operation (cache expiry, cache initial time). The disk manager 206 readjusts cache contents as drives are removed and added. It also gracefully handles runtime errors from the physical media. SAS, SATA, SSD, and flash drives are supported together on the same system. The disk manager 206 allows high levels of parallelism to multiple drives. A possible embodiment allows pre-loading of the cache with media which is can be served until a specific date and/or time.

Intelligent Caching

[0081] Hard drives typically do not operate well as caches, when a hard drive is used as a cache and it is caching hundreds, thousands, or even tens of thousands of videos, the end result is a random input/output (IO) pattern. The MFD places interesting content in the same area of the hard drive so the MFD can perform large reads to compensate for bandwidth, e.g., for a low bandwidth connection, the MFD can read 30 seconds of video in one read and store it in the cache in the RAM or SSD (or buffered on the hard drive depending on the number of connections being served and bandwidth considerations of those connections). This allows the client to be served out of that cache and the client does not need its next IO for 30 seconds. The SSD is used for content that is accessed more frequently or for content that

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0508
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

requires smaller random IOs that cannot be efficiently served from the hard drive. A popularity index is used for portions of content to determine whether a portion is hot. For videos, people may not be watching the entire video and some portions of a video may be hotter than others, so the hottest portion of a content is cached. This allows the MFD to be more intelligent in its caching. The MFD does not have to cache an entire video stream, for example. The MFD dynamically computes the popularity of portions of content.

[0082] Referring to FIG. 3, the boundaries of the hot portions of video are determined dynamically as the MFD measures accesses of a video stream. In a possible embodiment, the server side player 201 uses a logical chunking approach to dynamically determine where the hot portion of the video is. For example, the server side player 201 may use frame identifiers such as time tags embedded in the video frames or storage memory address locations. The frame identifiers may be obtained by the server side player 201 using a table of offsets created for the video stream. The table of offsets may be created by the system upon receipt of the video stream or created by a content provider which is loaded into the system at the time the video stream is received. Multiple tables may be stored for different bit rates of the video stream.

[0083] The server side player 201 can determine the portion or boundaries of the video stream that covers the most viewed frames of the video stream and places the hot portion of the video stream into one of the caches (hard drive, SSD, or RAM). In a possible embodiment, the analytics module monitors video 1 301, video 2, 302, video 3, 303, and video 4 404. It finds that there are sections of the videos that are viewed more frequently than other parts of the videos or other videos by client systems. Those portions of the videos 305, 306, 307, 308, are stored in the cache 309. The cache 309 may be any of the caches, RAM cache, SSD cache, or hard drive cache, or any combination thereof.

[0084] In a possible embodiment, the server side player 201 supports physical chunking of a stored video stream that partitions the video stream into small files and determines which file is popular. Physical chunking splits the video into small named chunks that are visible to the client player.

[0085] Time-based access patterns are used to determine which videos tend to be watched together, e.g., subject-related news items or content-related videos, and the disk manager 206 co-locates the hot portions of the videos in the same area in the hard drive cache, SSD cache or RAM cache, thereby increasing cache hits and allowing for faster access times.

[0086] The RAM cache is integrated with the buffer manager 203. It uses analytics and/or policies to manage the RAM cache and select eviction candidates. As the server side player 201 finds each hot video portion, the buffer manager 203 determines which hot

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0509
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

portions already stored in the cache are no longer determined to be hot. The buffer manager 203 removes portions that are no longer hot from the caches in order to make room for new hot portions. This is done by the buffer manager 203 based on internal analytics derived from access patterns. In an embodiment, the buffer manager 203 keeps all buffers that are not in active use organized by eviction order. The ordering takes into account factors like the number of times the buffer was accessed, the time frequency of accesses, and the increasing/decreasing rate of use.

[0087] The buffer manager 203 can use main memory (RAM) as a cache for high-bit rate videos and SSD memory for low-bit rate videos. This allows the MFD to more efficiently use the main memory.

[0088] The media manager 204 manages the cache hierarchy (e.g., SSD, hard disk, peer). It looks at analytics and/or policies to determine when to promote, demote, or evict cache contents. The media manager 204 initiates the hierarchical cache management which includes cache promotion and demotion actions. The implementation for each element in the cache hierarchy is called a cache provider. Providers are dynamically registered and assembled into a hierarchy based on the properties that they register, e.g., bandwidth, latency, capacity, etc. There are two classes of providers: cache providers that are local caches to the system that support reads and writes; and origin providers that are networked sources that supply origin content and are read-only. The units of cache migration can incorporate the logical chunk boundaries or in some cases be purely derived from device level block management. For example, in the HDD, popular blocks (that may contain portions of multiple videos) can be promoted to SSD or to the outer tracks of the HDD.

[0089] Metadata describes where media is stored among the disks. Most operating systems cache information about files. The information is called an inode. An inode cache can hold many inodes, typically on the order of millions. This cache will consume precious memory which the MFD could use to cache media data. The MFD has information to tell it where various pieces of media data are stored within the raw partition of a disk. This information falls under metadata and is called an "extent". Each disk block which holds a large media object can have a single extent to refer to that data. When a disk block holds multiple small objects, each small object has a separate extent to describe it.

[0090] In order to reduce the inode cache space which is hard to control in some systems, the metadata is packed, including extents, into single inodes/files called containers. Each container holds the extents for a set of objects that are common. Because the objects are common the MFD would have normally read the metadata for these objects.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0510
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0091] In a possible embodiment, the MFD creates a dynamic disk hierarchy that allows the MFD to determine the bandwidth of the disks within the MFD. The disk manager 206 determines which drives are possible cache targets. It runs a short test to categorize each drive according to disk bandwidth. This is done because the same type of drive can see different performance numbers depending on which disk controller is in front of it. By running a disk test, the disk manager 206 does not need to distinguish between various drive types or figure out what performance each disk controller can deliver. The drive/controller bandwidth combination will place the pair within some category set by the factory or by an administrator. There typically are a preset number of categories.

[0092] In a possible embodiment, the MFD creates a disk cache hierarchy. Each level of the cache hierarchy contains a set of objects. Objects near the top in terms of number of accesses are good candidates to move into the next fastest hierarchy, thereby enabling a decrease in total access time for all objects. Movement of an object from a lower level to a higher level can be determined using the bandwidth usage and the space usage to determine when it is appropriate to move the object. Objects near the bottom in terms of number of accesses are good candidates to move into the next slowest hierarchy, freeing up room for higher access count objects. The MFD attempts to maximize the available bandwidth for the highest level to the lowest level. For example, three levels, A, B, and C, are defined, with A being the fastest cache. If A is not full, then it would make sense to promote the highest accessed objects in B to A. This would leave more room to promote objects from C to B. If some objects in B are accessed less frequently than objects in C, then it would make sense to demote the least accessed objects in B by moving them from B into C and the promote the most accessed objects of C into B. This movement process is called "promotion" and "demotion".

[0093] The outer 10-15% of each disk drive can deliver 15-20% better performance than the random performance of an entire drive. A possible embodiment can carve out the outer part of a drive and migrate the hottest parts of media objects to this area.

Buffer Manager 203

[0094] The buffer manager 203 provides the primary data path within the MFD. All content served by the MFD are sent via buffers. Buffers are used both for speed matching (buffering) between the outbound network and the cache providers as well as a RAM cache. The latter is the (implicit) first level in the cache hierarchy. When requested content is not available in the cache, the buffer manager 203 uses the services of the media manager 204 to retrieve that data and cache it for subsequent requests if possible.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0511
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0095] The buffer manager 203 receives a request for content from a task that the HPE creates. The buffer manager 203 fetches content. Given a URL and an offset to play from, it will supply the content. The buffer manager 203 retrieves the content from media manager 204. The buffer manager 204 creates a task to retrieve a portion of the video content starting at the offset. The media manger 204 retrieves the content form the hard drive or SSD. The buffer manager 203 can check if the requested content matches any content stored in the main memory cache (RAM). The RAM cache is composed of a plurality of buffers. A hit can be directly served from the cache without any further copying. A miss is served by filling a new buffer using the media manager 204 to locate the content in storage. The buffer manager 203 passes a pointer to the buffer containing the portion of video content to the output protocol engine 210 which streams the portion of video content to a client.

[0096] The buffer manager 203 efficiently manages a large amount of memory. It has the ability to share buffers across connections that are serving the same content. The buffer manager 203 also has an efficient re-use scheme derived from the usage pattern in the MFD.

[0097] The buffer manager 203 handles buffer eviction based on a cost function that is calculated based on the use count and age. A higher use count results in a higher cost, whereas the age acts as a decay function on the cost, e.g., older buffers have a lower cost. The eviction algorithm attempts to approximate finding the lowest cost buffer while operating very efficiently (constant time).

[0098] A buffer that can be evicted (has a zero reference count) is inserted at the tail of an LRU list. The lists are ordered by the use count of the buffers. Each LRU list is effectively ordered by timestamp. A buffer that is not in the cache (e.g. expired, purged) is put on the head of the lowest LRU list with a zero timestamp.

[0099] During the eviction, the cost is computed for the head of each LRU list and the lowest cost buffer is evicted.

[0100] Each object (URI) has an expiry time that is part of the attributes. On access in the buffer cache, the expiry time is checked against the current time. If expired, all the buffers and the attributes associated with the object are removed from the cache. The buffers are put on the LRU lists when the reference count becomes zero. An object can be revalidated with the origin server during a time window (that can be configured by the administrator or user) prior to the expiry time.

[0101] Referring to Fig. 9, in a possible embodiment, the buffer manager 203 manages the buffer pool 904. The memory available for data is statically divided into an array of fixed size pages. Each page has a corresponding buffer structure to encapsulate the metadata associated with a buffer. The buffer structures are also statically allocated from an array.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0512
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Each buffer can be linked via the buffer cache and the free list. Some of the elements of the metadata are:

- Identity. The identity of the buffer. Data buffers have an ID consisting of the UOL (URI, Offset and Length). Attribute buffers have an ID of URI only.
- Reference Count. The number of active users of the buffer.
- Use Count. The number of times the buffer has been used. This is the basic measure of the buffers "popularity".
- Timestamp. The time at which the buffer is put on the LRU list (i.e. reference count is zero).

[0102] The buffer manager 203 manages the buffer cache. A hash map using the UOL as the key is maintained for all buffers that have an identity and can be shared with other connections. A possible embodiment supports a non-cached mode for a buffer whereby it has an identity, but is not shareable.

[0103] Data buffers associated with the same object (URI) are linked together with a list rooted in the corresponding attribute buffer. This allows fast application of actions that pertain to the entire object (e.g., cache expiry and object purge).

[0104] A free list is also maintained by the buffer manager 203. All buffers, including those with an identity, that have a zero reference count are on this list. This allows for efficient eviction of buffers as needed. The list can be composed of multiple lists ordered by the use count of the buffers.

[0105] Each GET task allocates some private data in the cache request private data 905 that is used to keep a list of its buffers for reference counting. It is also used to link to an underlying media request. The cache request handler 902 manages the cache request private data 905 and handles the GET task requests.

[0106] The media request handler 903 captures the in/out arguments for each media GET task sent to the media manager 204. Media Request objects are linked into a media request table 906 so that multiple requests to the same, or co-stored, objects can be synchronized.

[0107] Referring to Fig. 10a, the interface between buffer manager 203 and the media manager 204 (which implements the cache hierarchy) has unique characteristics that enable the high performance caching system. The selection of the device that provides the data is done using a STAT task that acts like a bidding system. Each layer in the cache hierarchy, e.g., solid state device (SSD) manager 205, disk manager 206, and origin manager 207, can respond, in turn, to bid for serving the request. The bid includes a "blocksize" that tells buffer manager 203 to allocate the requisite set of buffers. The buffer manager 203 then

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0513
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

sends down a GET request to the specific cache provider that won the bid. The request and response are logically decoupled. The provider fills the supplied buffers with content and sets the appropriate identity for each buffer. This results in the new buffers being inserted into the RAM cache. Upon completion of the task, buffer manager 203 looks up the RAM cache again and is able to find the buffers it was looking for. This model allows the providers to supply additional data, either subsequent portions of the same object or other related objects, in the same request. It also allows various portions of an object to be supplied by different providers. Naming and location are orthogonal. For example, a video object with a name /a/b/c/one.flv can exist (in parts or whole) in any of providers in the cache hierarchy.

Media Manager 204

[0108] The media manager 204 promotes and demotes videos based on the relative hotness of the video. The hotness of the video is a function of the number of hits on that video, the frequency of hits, and the increasing or decreasing trend of the frequency.

[0109] The media manager 204 has n different sources, the SSD(s) 205, hard disk(s) 206, origin server 207, or peer 208. It receives a buffer and fills it with a portion of the requested content. The media manager 204 sends the filled buffer to the buffer manager 203. If the content is not in local storage the media manager 204 retrieves the content from the origin server, a peer, or a client. The following is the hierarchy of speeds, in a possible embodiment, associated with each source:

1. Memory (RAM): 10 Gbps
2. Flash: 4 Gbps
3. Solid State Device (SSD): 1.6 Gbps
4. Disk (SAS): 550 Mbps
5. Disk (SATA) – 400 Mbps
6. Others
7. NFS and other file caches.
8. Origin Manager.

[0110] Referring to Fig. 10b, the media manager 204 first places a new video in the lowest memory hierarchy. On a timer expiry, the media manager 204 checks the analytics/policies manager 1002 for the hottest and the coldest video. The analytics/policies manager 1002 provides the media manager 204 with the hottest videos and/or portions of videos accessed. The hottest video or portion of video is promoted to the next memory cache in the hierarchy and the coldest one is demoted to the next lowest in the hierarchy. In this

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0514
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

way, a video can be placed in the lowest tier of a cache hierarchy and can bubble to the top of the hierarchy if its hotness increases relative to other videos.

[0111] The media manager 204 drives the eviction of videos from the respective caches. It implements an eviction timer. When the timer fires, the media manager 204 goes through each cache provider 205, 206, 207, 1006, checking whether the cache is full to a certain threshold. If a particular cache is full, the media manager 204 calls the eviction functionality of the provider. The provider then performs its specific algorithms for eviction.

Peer Manager 207

[0112] The peer manager 207 can be configured to obtain certain content from a peer MFD upon a cache miss. Given that a MFD has a limited amount of storage space in local storage such as one or more hard drives and one or more SSDs, a scenario may exist where a customer has more video content than a single MFD can store. The customer's video content can be distributed among a plurality of peer MFDs installed at the customer's site or, where the peer MFDs are accessible to each other over a network such as an intranet, Internet, LAN, or WAN, distributed among the plurality of peer MFDs that are in communication with each other.

[0113] An administrator can inform each peer MFD of the IP addresses of its peers so a peer MFD is able to communicate with its peers. If the peer MFDs are locally attached, they can discover their peers by themselves through any type of discovery mechanism. The peer MFDs can communicate among themselves and let each peer MFD know what URLs it caches. This means that after each peer MFD has loaded content from an origin server or another peer MFD, it informs each of its peer MFDs about the URLs that it serves. This can be via a message or series of messages sent to each peer MFD or broadcasted to all peer MFDs. If a request for a specific URL is sent to a peer MFD that causes a cache miss, the peer mgr in that MFD can request the URL content from a peer MFD that caches that specific URL.

Origin Manager 208

[0114] The origin server may push content to the origin manager 208 or the origin manager 208 may pull content from the origin server. The origin server may push content to the origin manager 208 by pre-staging the content to the origin manager 208. The origin server can FTP or HTTP into the origin manager 208 and tell the origin manager 208 to cache certain URLs because the content provider knows that the content will be hot.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0515
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0115] The origin manager 208 pulls content from the origin server as a result of a client request that misses all locally available caches (also a proxy function). Pulling content can also occur due to a policy or command driven action to fetch content separately from a client request. The proxy function is separate from the cache ingestion. There are several reasons for this: (a) the proxy may have to serve oddly aligned byte range requests that are not convenient for caching; (b) the system may need to perform processing functions like chunking and transcoding during ingest (these are difficult to do in real time in the proxy); and (c) different policies for the two functions.

Input Protocol Engine 209

[0116] The input protocol engine 209 operates in the same manner as the output protocol engine 210, except that it communicates with origin servers using protocols such as HTTP, FTP or NFS.

Optimized Network Stack 211

[0117] The optimized network stack 211 uses the round trip time (RTT) in the TCP stack to estimate the last mile bit rate variation to the client. The RTT time tells the optimized network stack 211 if there is congestion down the line. The RTT provides an estimate of bandwidth available in the network going to the client. The optimized network stack 211 also implements error handling and congestion control using algorithms suitable for time sensitive applications such as video. The MFD can change the bit rate in response to the detected bit rate changes without the client knowing.

Assured Flow Rate

[0118] In a possible embodiment, the MFD has the ability to guarantee a certain class of service to every flow admitted into the system for data delivery. This is called the assured flow rate. The MFD calculates the assured flow rate for each session and sets up a maximum bandwidth for the session. The bandwidth usage for the session should be no less than the assured flow rate and should attempt to reach maximum bandwidth if there is bandwidth available.

[0119] The MFD implements flow control to ensure that existing sessions get the required rate and no more than the maximum. This is achieved in the network socket level. The socket level manages the data send rate. The session bandwidth queue and server timer are used for calculating flow control.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0516
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0120] The MFD performs admission control and connection management by ensuring that a new session is not admitted if the server does not have enough resources to accept the connection.

[0121] Four metrics can be used for the MFD to maintain the assured flow rate (AFR):

- $\sum$ AFR of all connections $\leq$ Appliance Interface port Bandwidth
- Session AFR $\leq$ Client Session Bandwidth
- $\sum$ AFR of all connections $\leq$ (Disk + Buffer) Read speed
- Poorest Quality video bit rate $\leq$ Session AFR $\leq$ Highest Quality video bit rate

Where:

- Session: A single TCP connection.
- $\sum$ AFR of all connections: Sum of AFR of all accepted sessions.
- Session AFR: Individual AFR for one single session.
- Client Session Bandwidth: The client session maximum bandwidth. Typically the client session maximum bandwidth refers to the max TCP throughput due to the limitation of the WAN environment. For a DSL download, the typical bandwidth is 1,500Kbits/sec.
- Video bit rate: Each video has a profile. Different profiles have a different quality. Higher quality videos require a higher bit rate. Not every video has a profile for all bit rates. A typical video profile bit rate could range from 200 Kbits/sec to 1,000 Kbits/sec.

[0122] To ensure that the MFD does not over commit server resources so that the AFR for existing sessions can be maintained, the MFD either rejects a new session or lowers the profile (if possible) for existing sessions to accommodate the new session based on a policy setting. A session is rejected by sending over a server busy HTTP reply or by simply closing the connection. The MFD does not close sessions in the middle of serving data. It simply lowers the video profile.

[0123] For example, if the total appliance network bandwidth is limited to 1Gbits/sec for each network interface port, then the $\sum$AFR of all sessions should be less than this bandwidth. When the sum of the AFR of all of the accepted sessions is equal to 1Gbits/sec, the MFD can stop accepting new sessions. In a possible embodiment, by considering the network overhead used by packet retransmission, ACK packet, TCP/IP header, other usages such as ARP, the off-line origin manager 208, etc., the bandwidth can be reduced to 0.8Gbits/sec for all session AFR calculations in order to take in to account a more realistic bandwidth limitation.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0517
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0124] The AFR bit rate can be between the poorest quality video bit rate and highest quality video bit rate. If the AFR < poorest quality video bit rate, then screen jitter will occur at the client side. If the AFR > highest quality video bit rate, then bandwidth resources are being wasted. The server side player 201 does not close a continuous session GET request but is allowed to close a new session by returning a failure error code.

[0125] In a possible embodiment, if a client is gaining access via a dial-in modem, the AFR is limited by this slow last mile connection. The network driver module detects the client connection ISP speed based on RTT calculation. An example equation can be: client session bandwidth speed = 1460 bytes / RTT in seconds.

[0126] As discussed above, video content is stored in the disk or RAM. Once the MFD determines the AFR, it needs to make sure that the disk + buffer can achieve this speed. Disk speed, as a capacity, is not something that is easy to measure, unlike CPU and network capacity. It is highly workload dependent. A more practical way to measure if the disk is at or close to capacity is to look at queue depth and/or average latency. The scheduler's 202 deadline miss counter is a good indicator of what is happening in the system.

[0127] Under certain situations, for example, the Web master wants to create a VIP list. The Web master through the user interface management console may set up a policy so that the VIP list receives high quality video which results in a higher AFR for the session. In another example, the Web master can set an average video quality. The Web master can use the user interface manager console to set an average AFR.

[0128] In a possible embodiment, the MFD does not want to send a higher video quality to a client that cannot play the high quality video. For example, a PC/Laptop can play high quality videos, but an iPhone/g-phone/Palm, etc., only has very small screen and therefore cannot play a high quality video. It is completely unnecessary to send over high quality videos to these kinds of devices.

[0129] Referring to Fig. 11, a data flow is shown for a possible embodiment of the AFR portion of the network manager in the optimized network stack 211 sending data out after the HPE posted a task to the scheduler 202 and the scheduler 202 returned a chunk of data. If the calculated size of data cannot be sent out within one timer interval slot (e.g., once a second), it means an AFR miss occurred in the network manager. If scheduler 202 does not return a task within a deadline miss time, it means an AFR miss in the disk manager 206.

[0130] The network manager calculates the deadline time for the task. It then posts a task with the calculated deadline time 1101. The network manager then calculates the maximum number of bits that can be sent over the connection given the AFR and the timer interval for the transmission period for the socket. This results in the maximum amount of data that can

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0518
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

be sent over the connection in one timer interval 1102. The network manager then sends the data over the socket 1103.

[0131] If the send was incomplete 1104, e.g., there was not enough space for the data to be sent or the timer interval has lapsed, then the network manager attempts to resend the data over the connection 1107, 1103.

[0132] If there was enough space for the data to be sent out/received 1104 (e.g., the buffer level on the client was sufficient, the TCP level was sufficient, the window size was sufficient, more data is to be sent, etc.), then the network manager checks to see if there is more data to send 1105. If more data is available to send, then the network manager puts the socket back into the timer queue to wait for the next timer interval to send more data 1108 and calculates the next maximum number of bits that can be sent over the connection 1102. Otherwise, the network manager is finished with the timer interval and gets the next chunk of data to send 1106.

[0133] The process continues until all data from the chunk(s) of data from the scheduler 202 has been sent.

[0134] In a possible embodiment, the MFD performs the following in order to maintain the AFR:

1. Connection requested on a network port. If this network port's total connections > max allowed connections, reject the connection.
2. Each connection can only share up to the calculated session bandwidth. In this example = 1Gbps/total connections on this network port.
3. Modification due to maximum session bandwidth configuration.
    a. If maximum configured session bandwidth is configured and the calculated session bandwidth > maximum session bandwidth, set the calculated session bandwidth to maximum configured session bandwidth.
    b. Else make no change.
4. Calculate session AFR.
    a. Set session AFR to configured AFR as base suggested AFR.
    b. Call server side player 201 for adjusting the AFR. If error message is returned by server side player 201, close session.
5. Modification due to balanced AFR calculation.
    a. If calculated session bandwidth > 1.2 * AFR, set the calculated session bandwidth to 1.2 * AFR.
    b. Else make no change.
6. Modification due to AFR.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0519
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

a. If the calculated session bandwidth < AFR, set the calculated session bandwidth to AFR.

b. Else make no change.

7. The calculated session bandwidth is enforced in every epollout event or task completion, on whenever data is about to be sent out to the client.

[0135] The network and HTTP manager provide three parameters to the server side player 201 for calculating the server side player 201 AFR value. The three parameters are:

- URI.
- Client Session Bandwidth (Ceiling value of AFR).
- Total available AFR of network bandwidth.

Where:

- Total available AFR may be trimmed by network module.
- Total available AFR ≠ Total network bandwidth - ∑AFR of existing sessions.
- Total available AFR = min (Total network bandwidth - ∑AFR of existing sessions, configured maximum AFR)

[0136] In the server side player 201 AFR calculation, the return AFR value can be less than client session bandwidth and total available AFR. The server side player 201 can be called at the time after the HPE finishes HTTP parsing. The HPE provides all HTTP information (such as URI, etc.) as well as a flag to show this is a new session or a continuing session with the same connection.

[0137] The server side player 201 does not close a continuing session GET request but should be allowed to close a new session. The server side player 201 sets the AFR to the value specified in the URI request. If the server side player 201 is unable to set the AFR to the requested value, it returns an error code to the HTTP Engine that will close the connection.

[0138] The scheduler 202 deadline misses can be used to measure disk throughput during runtime. The HPE calculates the deadline (using deadline = func (total HTTP bytes/ AFR)) in microseconds and sends the deadline to the scheduler 202 when submitting a task to the scheduler 202. If a disk operation is over loaded, then the scheduler 202 drops some new tasks. The selected dropped tasks should belong to the first tasks created by a new session. A flag can be set by the HPE to tell the scheduler 202 if a task belongs to a new session or not.

[0139] When there are too many disk deadline misses, the scheduler 202 sends feedback to the network manager to reduce the maximum allowed connections in the server structure. This reduces the total session number.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0520
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0140] After a disk operation is recovered and the deadline miss issue is passed over, the scheduler 202 tries to recover back to the maximum allowed connections setting in the server structure.

Hint Tracks

[0141] Referring to Fig. 12, in a possible embodiment, a progressive download hint track (PDH) 1206 describes to a MFD how to serve media data residing in a file 1201 using progressive download over HTTP. Only one container file 1201 needs to reside in the MFD. Within this container file 1201 a user could store streams 1204, 1205, of varying bit-rates, spatial resolutions or scalable streams. The container file 1201 can also contain metadata 1202 as with MP4 container files. The PDH-Hint track 1206 would be a higher level meta box within the video file container (moov in this example) 1203 that describes to the MFD all the options available to it to support PDH.

[0142] In a possible embodiment, depending on requests from the client to adapt to varying bandwidth conditions, available client CPU resources, or by the MFD polling the network to account for network latency, the MFD side logic can make intelligent decisions as to how and what kind of bit rate, frame rate, or resolutions it should serve to best meet the conditions of the network and the requirements of the client.

[0143] In a possible embodiment, the hint track can be used by the MFD to decide the best rate or resolution to serve and then encapsulate the relevant portion of the time period of media data as a self containerized chunk of MP4 to deliver over HTTP. The media data itself that resides in the generic container does not need to be reformatted in any way.

[0144] In a possible embodiment, multiple predefined PDH hint tracks could be created that cover most of the common usage conditions to reduce the load on the MFD side.

[0145] In a possible embodiment, a single stream of media data can be used in a server composed to serve at a predefined frame rate (e.g., an H.264 stream coded at B_1 kbps at 30 fps). During delivery, depending on network conditions, the MFD can choose to drop the higher temporal enhancement layers (information for which can be easily obtained by the layering boxes in MP4). As such, the rate of the PDH chunk for these lower frame rate portions (e.g., 15 fps, 7.5 fps) would be less than B_1 kbps thereby providing a mechanism to quickly adapt to bandwidth fluctuations while not affecting the playback quality experience of the user to a large extent.

[0146] In a possible embodiment, an additional hint track (or higher level box) can be used to describe to various options available to support trickplay. Such a hint track (or box) can provide information to the MFD about the speed of fast-forward or rewind (2x/4x/-2x/-

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0521
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4x, etc.) along with precise information about relative frame offsets and playback rate that will aid the client player during playback.

[0147] In a possible embodiment, the hint track usage could also be extended to support MFD side playlists. These hint tracks can provide information about where ads or video segments need to be inserted and their exact durations. The hint tracks can also provide control mechanism information, for example, if a client player can skip or fast-forward through an ad or video segment. The hint tracks may also contain information about which ad server to contact, if needed, for the dynamic placement of ads or video segments.

MFD Policies

[0148] An administrator is able to set policies for the MFD to operate upon. For example, the administrator may desire to have the MFD serve as many clients as possible or serve HD content and retain the quality of the HD content at the cost of serving a large number of clients.

[0149] The feedback from the scheduler 202 as to the amount of time that tasks have been late, the number of tasks that have been late, or any other useful metric, can be sent to the server side 201 and/or the output protocol engine 210. The server side 201 can use the information to adjust the number of clients that it is serving or even adjust the quality of the video that it delivers to certain clients in order to resolve the delivery latency. It may be that certain clients running at certain bandwidths are flexible enough to have their bandwidth dynamically lowered.

[0150] The output protocol engine 210 can use the information from the scheduler 202 to deny any new client connections until the scheduler 202 passes the output protocol engine 210 statistics that show that the MFD has caught up sufficiently with the desired delivery efficiency. The parameters defining what criteria are used to make such a determination may be defined by an administrator. Such parameters may include performing an action based on the number of clients that are being affected by late deliveries or the amount of latency that exists, e.g., the number of tasks that are missing deadlines.

[0151] An administrator may configure the origin server to notify the MFD about which content servers it should communicate with in order to retrieve content when the MFD experiences a cache miss. The origin server can instruct the origin manager 207 to obtain certain content from a specific content server using a specific protocol. The origin server can also define policies on the MFD to tell the origin manager 207 which content is cacheable and which content must be obtained from an origin server, e.g., seldom requested content.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0522
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MFD Extensions

[0152] The MFD may also be bandwidth limited at the factory. The MFD may be set to not exceed a certain peak bandwidth or number of open sessions. This allows the MFD provider to sell certain levels of service. If a customer requires a higher peak bandwidth or more open sessions, the MFD provider can remotely set the peak bandwidth or maximum number of open sessions of the customer's MFD remotely for a fee that may be a one-time fee or a recurring fee. The MFD may also automatically exceed the maximum bandwidth or maximum number of open sessions and notify the MFD provider of the amount exceeded. The MFD provider may then charge the customer a fee for the exceeded amount. This allows the customer to temporarily handle high activity bursts and not have to purchase an additional MFD or a MFD upgrade.

Load Balancer Interaction

[0153] There typically is a load balancer in communication with the peer MFDs. The load balancer can be configured to send a client request to a certain peer MFD when the request is for a specific URL. One of the peer MFDs may a send a table of what URL is available for each peer MFD. Alternatively, each peer MFD can tell the load balancer what URLs it serves. The load balancer can gather all of the information from the peer MFDs and create its own tables or database entries in order to direct client URL requests to the correct peer MFD. The load balancer may use a hashing scheme based on the URL and/or other parameters to determine the MFD to direct a URL request to. This allows the URL content to be distributed among the peer MFDs.

Monitoring and Statistical User Interface

[0154] An administrator, or the like, can look at the MFD's stored statistics that the server side player 201 records and can see via textual or graphical representations how the MFD is performing. The media manager 204 also records the number of cache hits and misses as well as what content is the most popular (hot content) or least popular. The media manager 204 can also record the origin servers that it has requested content from and the number of times such requests were made. The user interface allows the administrator to observe the content performance data in a graphical or textual representation. The user interface can present any recorded information for a specified period of time or in real time.

[0155] A graphical user interface allows the MFD to use statistical information that it can collect with regard to disk use and content accesses to display graphical charts that visually represent any of:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0523
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

- Histogram data showing recent data retrieval times per disk.
- Disk access counters.
- Number of videos per disk.
- Amount of free space.
- Number of videos requested.
- Number of videos requested per disk.
- Recent throughput per disk.
- Hits/misses in the buffer cache.
- Various buffer operations.
- Errors (from media manager statistics, media request, etc).
- Number of videos stored.
- Number of videos per cache.
- Number of videos created.
- Number of videos deleted.
- Number of media providers.
- Meta-information about media providers.
- Bandwidth served per cache level (memory, SSD, Disk) as well as Origin sources (HTTP, NFS, etc.).
- Bandwidth served per cache device (each SSD or Disk unit). The caching logic is designed to load level across devices as the workload profile changes with respect to content.
- Bandwidth and connections served per interface port.
- Bandwidth and connections served per protocol (HTTP, RTSP, etc).

[0156] In a possible embodiment, a user may configure parameters relating to the performance characteristics of the MFD. Components such as the server side player and media manager record performance statistics that are used by the graphical user interface to display statistics and graphs to the user. Fig. 13 shows a user interface screen in a possible embodiment where the user can specify parameters such as the AFR 1304 or the maximum bit rate per session 1305. The user can also specify a namespace 1301 along with its URI 1302 and the host name of the HTTP origin server 1303 for the namespace.

[0157] Referring to Fig. 14, in a possible embodiment, a graphical user interface screen 1400 can show a real time graph of the aggregate number of bytes 1401 transferred from the cache, origin server, and total overall bytes delivered to clients over time 1402. The

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0524
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

graphical user interface screen 1400 allows the user to see the actual performance of the MFD as it delivers content to clients 1403. The user can customize the policies in the MFD to adjust for cache storage, for example, to handle certain hot content that the user always wants available. The graphical user interface 1400 immediately shows the user the affect that his changes have on the MFD.

[0158] Fig. 15 shows a screenshot 1500 of the graphical user interface displaying the performance characteristics over the MFD's connections 1501.

3.2 MEDIA FLOW DIRECTOR PLACEMENT

[0159] FIG. 4 illustrates a possible embodiment using MFD's 401 to replace existing video servers 402. Existing video server implementations are typically a group of file servers 403 that cache content from an origin server or a group of origin servers 404. The content from the origin servers 404 are mirrored across many video servers 403. A user, via a client, visits a web page on a portal server 405 to search for video content. The user selects a video from the web page via a link on the page. The URL is sent to a DNS server that resolves the URL to an IP address that a load balancer is responsible for. The client is passed the resolved IP address which is used by the client to make the video request. A router, in a group of edge routers 407, routes the request to the load balancer in a distribution center 406. The load balancer receives the request and forwards the URL request to a video server in the video server cluster 402. The URL request may inform the video server the bit rate of the requested video that the client is expecting. The video server is a file server that has no knowledge of the characteristics of the file that it is delivering other than the fact that it must deliver a certain bit rate version of the video to a client. Typically the name and content is opaque to the video server. Each bit rate of the video is typically a named file. The video server streams the video file to the client and attempts to deliver the video to the client such that the client's video buffer does not become empty.

[0160] The client must buffer the received video in order to cover for delays in the streaming of the video from the video server. The client's buffer is meant to give the user a continuous viewing experience. Given that the video servers are serving large amounts of videos to many clients, delivery delays that are created by a server running out of bandwidth or by network delays to a client are frequent and client buffers become empty. As a result, the user either sees the video stream stop playing until the buffer starts to fill again or, for small video files, waits until the entire video has loaded into the buffer before the video starts to play.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0525
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0161] In a possible embodiment, one MFD has the ability to achieve a throughput of 10 Gbps+ resulting in a single MFD replacing ten or more video servers. Two MFDs 401 can replace a bank of 20 video servers. The MFDs 401 communicate with origin servers 402 in order to obtain video content when a MFD receives a request for URL content that the MFD does not have in storage or the when MFD accepts new video content from an origin server. The MFDs are a direct replacement of video servers and the infrastructure of the portal servers 405, edge routers 407, and distribution switches 406 remains the same.

[0162] FIG. 5 illustrates the placement of MFDs at edge and/or origin locations. In this example, the dashed lines are the data paths for a conventional implementation. A MFD may be installed at a content owner's site 501, a CDN 502, 506, a content aggregator site 504, or an ISP. Any combination of locations may have a MFD installed. resulting in faster and more reliable content delivery to clients 508. For example, when two MFDs are installed at an ISP 507, the MFDs are deployed somewhere in the ISP network. This can be either in the same data center in a cluster or in different data centers but in the same ISP access network. The MFDs can communicate with each other to deliver the content more efficiently because, if the MFDs can retrieve the content from within the same network, it is more efficient than retrieving the content from across different and/or multiple different networks or from an origin site. If a large content aggregator 504 deploys an all-MFD private delivery network, the content aggregator sees even more efficiency since the MFDs can use private protocols to processes cache miss processing, pre-staging, data collection, etc., even more efficiently.

[0163] When an all-MFD implementation is achieved, there is a dramatic increase in efficiency due to the MFD's ability to retrieve content via the most efficient route.

[0164] Note that the term "video content" has been used in embodiment descriptions throughout the text. The term "video content" may be replaced by any media type in the embodiments discussed throughout as the MFD is adaptable to deliver any type of content files, such as: media (audio, pictures, etc.), games, software, HTML, scripts, etc.

4.0 IMPLEMENTATION MECHANISMS -- HARDWARE OVERVIEW

[0165] Figure 6 is a block diagram that illustrates a computer system 600 upon which an embodiment of the invention may be implemented. Computer system 600 includes a bus 602 or other communication mechanism for communicating information, and a processor 604 coupled with bus 602 for processing information. Computer system 600 also includes a main memory 606, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 602 for storing information and instructions to be executed by processor 604. Main memory 606 also may be used for storing temporary variables or other intermediate

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0526
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

information during execution of instructions to be executed by processor 604. Computer system 600 further includes a read only memory (ROM) 608 or other static storage device coupled to bus 602 for storing static information and instructions for processor 604. A storage device 610, such as a magnetic disk or optical disk, is provided and coupled to bus 602 for storing information and instructions.

[0166] Computer system 600 may be coupled via bus 602 to a display 612, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 614, including alphanumeric and other keys, is coupled to bus 602 for communicating information and command selections to processor 604. Another type of user input device is cursor control 616, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 604 and for controlling cursor movement on display 612. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0167] The invention is related to the use of computer system 600 for implementing the techniques described herein. According to one embodiment of the invention, those techniques are performed by computer system 600 in response to processor 604 executing one or more sequences of one or more instructions contained in main memory 606. Such instructions may be read into main memory 606 from another machine-readable medium, such as storage device 610. Execution of the sequences of instructions contained in main memory 606 causes processor 604 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

[0168] The term "machine-readable medium" as used herein refers to any medium that participates in providing data that causes a machine to operation in a specific fashion. In an embodiment implemented using computer system 600, various machine-readable media are involved, for example, in providing instructions to processor 604 for execution. Such a medium may take many forms, including but not limited to storage media and transmission media. Storage media includes both non-volatile media and volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 610. Volatile media includes dynamic memory, such as main memory 606. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 602. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications. All such media must be tangible to

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0527
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

enable the instructions carried by the media to be detected by a physical mechanism that reads the instructions into a machine.

[0169] Common forms of machine-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, or any other medium from which a computer can read.

[0170] Various forms of machine-readable media may be involved in carrying one or more sequences of one or more instructions to processor 604 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 600 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 602. Bus 602 carries the data to main memory 606, from which processor 604 retrieves and executes the instructions. The instructions received by main memory 606 may optionally be stored on storage device 610 either before or after execution by processor 604.

[0171] Computer system 600 also includes a communication interface 618 coupled to bus 602. Communication interface 618 provides a two-way data communication coupling to a network link 620 that is connected to a local network 622. For example, communication interface 618 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 618 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 618 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0172] Network link 620 typically provides data communication through one or more networks to other data devices. For example, network link 620 may provide a connection through local network 622 to a host computer 624 or to data equipment operated by an Internet Service Provider (ISP) 626. ISP 626 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 628. Local network 622 and Internet 628 both use electrical, electromagnetic or optical signals that carry digital data streams.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0528
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0173] Computer system 600 can send messages and receive data, including program code, through the network(s), network link 620 and communication interface 618. In the Internet example, a server 630 might transmit a requested code for an application program through Internet 628, ISP 626, local network 622 and communication interface 618.

[0174] The received code may be executed by processor 604 as it is received, and/or stored in storage device 610, or other non-volatile storage for later execution.

5.0 EXAMPLES

[0175] In an embodiment, an apparatus comprises: a media delivery request receiver that receives requests for media content from client systems, a flow rate subsystem that calculates a maximum bandwidth and a target flow rate for each outbound media flow, a delivery subsystem that, for each outbound media flow, delivers at least a portion of a requested media content to a client system, a media delivery scheduling subsystem that determines when the delivery subsystem delivers at least a portion of a requested media content, the media delivery scheduling subsystem dynamically adjusts flow rate for each outbound media flow in order to meet the target flow rate for each outbound media flow, the media delivery scheduling subsystem instructs the media delivery request subsystem to refuse incoming media requests when the media delivery scheduling subsystem detects that target flow rates for media flows may not be achieved if additional media content requests are accepted.

[0176] In an embodiment, an apparatus further comprises wherein the media delivery scheduling subsystem instructs the delivery subsystem to deliver at least a portion of a lower quality media content in order to meet the target flow rate for a specific media flow.

[0177] In an embodiment, an apparatus further comprises wherein the media delivery scheduling subsystem instructs the delivery subsystem to deliver at least a portion of a higher quality media content in order to meet the target flow rate for a specific media flow.

[0178] In an embodiment, an apparatus further comprises wherein the media delivery scheduling subsystem detects that a target flow rate for a media flow has not been met when a deadline for delivery of a next portion of a media content for the media flow has been missed.

[0179] In an embodiment, an apparatus further comprises a plurality of storage devices, at least a portion of the plurality of storage devices having differing performance characteristics, a media caching subsystem that stores at least a portion of a media content in a specific storage device based at least upon the target flow rate for the media flow associated with the media content and the performance characteristic of the specific storage device, the media caching subsystem instructs the delivery subsystem to deliver the stored portions of the media content from the specific storage device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0529
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0180] In an embodiment, an apparatus further comprises wherein the flow rate subsystem determines a target flow rate for a media flow using at least a measured bandwidth speed of a client system associated with the media flow.

[0181] In an embodiment, an apparatus further comprises wherein the flow rate subsystem dynamically determines a target flow rate for a media flow using at least a dynamic measurement of bandwidth speed of a client system associated with the media flow.

[0182] In an embodiment, a method comprises or a computer-readable storage medium carrying one or more sequences of instructions, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of: receiving requests for media content from client systems, calculating a maximum bandwidth and a target flow rate for each outbound media flow, delivering at least a portion of a requested media content to a client system for each outbound media flow, determining when the delivering step delivers at least a portion of a requested media content, dynamically adjusting flow rate for each outbound media flow in order to meet the target flow rate for each outbound media flow, refusing incoming media requests when upon detecting that target flow rates for media flows may not be achieved if additional media content requests are accepted.

[0183] In an embodiment, a method or computer-readable storage medium further comprises wherein the delivering step delivers at least a portion of a lower quality media content in order to meet the target flow rate for a specific media flow.

[0184] In an embodiment, a method or computer-readable storage medium further comprises wherein the delivering step delivers at least a portion of a higher quality media content in order to meet the target flow rate for a specific media flow.

[0185] In an embodiment, a method or computer-readable storage medium further comprises detecting that a target flow rate for a media flow has not been met when a deadline for delivery of a next portion of a media content for the media flow has been missed.

[0186] In an embodiment, a method or computer-readable storage medium further comprises storing at least a portion of a media content in a specific storage device, among a plurality of storage devices with at least a portion of the plurality of storage devices having different performance characteristics, based at least upon the target flow rate for the media flow associated with the media content and the performance characteristic of the specific storage device, wherein the delivering step delivers the stored portions of the media content from the specific storage device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0530
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0187] In an embodiment, a method or computer-readable storage medium further comprises determining a target flow rate for a media flow using at least a measured bandwidth speed of a client system associated with the media flow.

[0188] In an embodiment, a method or computer-readable storage medium further comprises dynamically determining a target flow rate for a media flow using at least a dynamic measurement of bandwidth speed of a client system associated with the media flow.

[0189] In an embodiment, an apparatus comprises: a plurality of storage devices, at least a portion of the plurality of storage devices having differing performance characteristics, a media caching subsystem that stores at least a portion of a media content in a specific storage device based at least upon a performance characteristic of the specific storage device, the media caching subsystem orders the plurality of storage devices in a hierarchical fashion based on bandwidth of each storage device among the plurality of storage devices, a storage optimization subsystem that optimizes bandwidth from each storage device among the plurality of storage devices by dynamically relocating portions of media content stored on each storage device based on usage measurements of the storage device.

[0190] In an embodiment, an apparatus further comprises wherein the media caching subsystem stores the at least a portion of the media content in the specific storage device based additionally upon a target flow rate for the media flow associated with the media content.

[0191] In an embodiment, an apparatus further comprises wherein the storage optimization subsystem relocates a portion of a media content from a first storage device to a second storage device having better performance characteristics than the first storage device based upon an increase in popularity of the portion of the media content.

[0192] In an embodiment, an apparatus further comprises wherein the storage optimization subsystem relocates a portion of a media content from a first storage device to a second storage device having lower performance characteristics than the first storage device based upon a decrease in popularity of the portion of the media content.

[0193] In an embodiment, an apparatus further comprises wherein the media caching subsystem stores multiple portions of media content that include the at least a portion of the media content in the specific storage device when the multiple portions fit in a specifically sized block of memory on the specific storage device.

[0194] In an embodiment, a method comprises or a computer-readable storage medium carrying one or more sequences of instructions, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of: storing at least a portion of a media content in a specific storage device,

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0531
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

among a plurality of storage devices with at least a portion of the plurality of storage devices having differing performance characteristics, based at least upon a performance characteristic of the specific storage device, ordering the plurality of storage devices in a hierarchical fashion based on bandwidth of each storage device among the plurality of storage devices, optimizing bandwidth from each storage device among the plurality of storage devices by dynamically relocating portions of media content stored on each storage device based on usage measurements of the storage device.

[0195] In an embodiment, a method or computer-readable storage medium further comprises wherein the storing step stores the at least a portion of the media content in the specific storage device based additionally upon a target flow rate for the media flow associated with the media content.

[0196] In an embodiment, a method or computer-readable storage medium further comprises relocating a portion of a media content from a first storage device to a second storage device having better performance characteristics than the first storage device based upon an increase in popularity of the portion of the media content.

[0197] In an embodiment, a method or computer-readable storage medium further comprises relocating a portion of a media content from a first storage device to a second storage device having lower performance characteristics than the first storage device based upon a decrease in popularity of the portion of the media content.

[0198] In an embodiment, a method or computer-readable storage medium further comprises storing multiple portions of media content that include the at least a portion of the media content in the specific storage device when the multiple portions fit in a specifically sized block of memory on the specific storage device.

6.0 EXTENSIONS AND ALTERNATIVES

[0199] In the foregoing specification, embodiments of the invention have been described with reference to numerous specific details that may vary from implementation to implementation. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0532
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

CLAIMS:

What is claimed is:

1. An apparatus for simultaneously distributing media across a network to a plurality of client systems, comprising:

a media delivery request receiver that receives requests for media content from client systems;

a flow rate subsystem that calculates a maximum bandwidth and a target flow rate for each outbound media flow;

a delivery subsystem that, for each outbound media flow, delivers at least a portion of a requested media content to a client system;

a media delivery scheduling subsystem that determines when the delivery subsystem delivers at least a portion of a requested media content, the media delivery scheduling subsystem dynamically adjusts flow rate for each outbound media flow in order to meet the target flow rate for each outbound media flow, the media delivery scheduling subsystem instructs the media delivery request subsystem to refuse incoming media requests when the media delivery scheduling subsystem detects that target flow rates for media flows may not be achieved if additional media content requests are accepted.

2. The apparatus of Claim 1, wherein the media delivery scheduling subsystem instructs the delivery subsystem to deliver at least a portion of a lower quality media content in order to meet the target flow rate for a specific media flow.

3. The apparatus of Claim 1, wherein the media delivery scheduling subsystem instructs the delivery subsystem to deliver at least a portion of a higher quality media content in order to meet the target flow rate for a specific media flow.

4. The apparatus of Claim 1, wherein the media delivery scheduling subsystem detects that a target flow rate for a media flow has not been met when a deadline for delivery of a next portion of a media content for the media flow has been missed.

5. The apparatus of Claim 1, further comprising:

a plurality of storage devices, at least a portion of the plurality of storage devices having differing performance characteristics;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0533
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

a media caching subsystem that stores at least a portion of a media content in a specific storage device based at least upon the target flow rate for the media flow associated with the media content and the performance characteristic of the specific storage device, the media caching subsystem instructs the delivery subsystem to deliver the stored portions of the media content from the specific storage device.

6. The apparatus of Claim 1, wherein the flow rate subsystem determines a target flow rate for a media flow using at least a measured bandwidth speed of a client system associated with the media flow.

7. The apparatus of Claim 1, wherein the flow rate subsystem dynamically determines a target flow rate for a media flow using at least a dynamic measurement of bandwidth speed of a client system associated with the media flow.

8. A method for simultaneously distributing media across a network to a plurality of client systems, comprising:

receiving requests for media content from client systems;

calculating a maximum bandwidth and a target flow rate for each outbound media flow;

delivering at least a portion of a requested media content to a client system for each outbound media flow;

determining when the delivering step delivers at least a portion of a requested media content;

dynamically adjusting flow rate for each outbound media flow in order to meet the target flow rate for each outbound media flow;

refusing incoming media requests when upon detecting that target flow rates for media flows may not be achieved if additional media content requests are accepted.

9. The method of Claim 8, wherein the delivering step delivers at least a portion of a lower quality media content in order to meet the target flow rate for a specific media flow.

10. The method of Claim 8, wherein the delivering step delivers at least a portion of a higher quality media content in order to meet the target flow rate for a specific media flow.

11. The method of Claim 8, further comprising:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0534
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

detecting that a target flow rate for a media flow has not been met when a deadline for delivery of a next portion of a media content for the media flow has been missed.

12. The method of Claim 8, further comprising:

storing at least a portion of a media content in a specific storage device, among a plurality of storage devices with at least a portion of the plurality of storage devices having different performance characteristics, based at least upon the target flow rate for the media flow associated with the media content and the performance characteristic of the specific storage device;

wherein the delivering step delivers the stored portions of the media content from the specific storage device.

13. The method of Claim 8, further comprising:

determining a target flow rate for a media flow using at least a measured bandwidth speed of a client system associated with the media flow.

14. The method of Claim 8, further comprising:

dynamically determining a target flow rate for a media flow using at least a dynamic measurement of bandwidth speed of a client system associated with the media flow.

15. A computer-readable storage medium carrying one or more sequences of instructions for simultaneously distributing media across a network to a plurality of client systems, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of:

receiving requests for media content from client systems;

calculating a maximum bandwidth and a target flow rate for each outbound media flow;

delivering at least a portion of a requested media content to a client system for each outbound media flow;

determining when the delivering step delivers at least a portion of a requested media content;

dynamically adjusting flow rate for each outbound media flow in order to meet the target flow rate for each outbound media flow;

refusing incoming media requests when upon detecting that target flow rates for media flows may not be achieved if additional media content requests are accepted.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0535
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

16. The computer-readable storage medium of Claim 15, wherein the delivering step delivers at least a portion of a lower quality media content in order to meet the target flow rate for a specific media flow.

17. The computer-readable storage medium of Claim 15, wherein the delivering step delivers at least a portion of a higher quality media content in order to meet the target flow rate for a specific media flow.

18. The computer-readable storage medium of Claim 15, further comprising:

detecting that a target flow rate for a media flow has not been met when a deadline for delivery of a next portion of a media content for the media flow has been missed.

19. The computer-readable storage medium of Claim 15, further comprising:

storing at least a portion of a media content in a specific storage device, among a plurality of storage devices with at least a portion of the plurality of storage devices having different performance characteristics, based at least upon the target flow rate for the media flow associated with the media content and the performance characteristic of the specific storage device;

wherein the delivering step delivers the stored portions of the media content from the specific storage device.

20. The computer-readable storage medium of Claim 15, further comprising:

determining a target flow rate for a media flow using at least a measured bandwidth speed of a client system associated with the media flow.

21. The computer-readable storage medium of Claim 15, further comprising:

dynamically determining a target flow rate for a media flow using at least a dynamic measurement of bandwidth speed of a client system associated with the media flow.

22. An apparatus for simultaneously distributing media across a network to a plurality of client systems, comprising:

a plurality of storage devices, at least a portion of the plurality of storage devices having differing performance characteristics;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0536
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

a media caching subsystem that stores at least a portion of a media content in a specific storage device based at least upon a performance characteristic of the specific storage device, the media caching subsystem orders the plurality of storage devices in a hierarchical fashion based on bandwidth of each storage device among the plurality of storage devices;

a storage optimization subsystem that optimizes bandwidth from each storage device among the plurality of storage devices by dynamically relocating portions of media content stored on each storage device based on usage measurements of the storage device.

23. The apparatus of Claim 22, wherein the media caching subsystem stores the at least a portion of the media content in the specific storage device based additionally upon a target flow rate for the media flow associated with the media content.

24. The apparatus of Claim 22, wherein the storage optimization subsystem relocates a portion of a media content from a first storage device to a second storage device having better performance characteristics than the first storage device based upon an increase in popularity of the portion of the media content.

25. The apparatus of Claim 22, wherein the storage optimization subsystem relocates a portion of a media content from a first storage device to a second storage device having lower performance characteristics than the first storage device based upon a decrease in popularity of the portion of the media content.

26. The apparatus of Claim 22, wherein the media caching subsystem stores multiple portions of media content that include the at least a portion of the media content in the specific storage device when the multiple portions fit in a specifically sized block of memory on the specific storage device.

27. A method for simultaneously distributing media across a network to a plurality of client systems, comprising:

storing at least a portion of a media content in a specific storage device, among a plurality of storage devices with at least a portion of the plurality of storage devices having differing performance characteristics, based at least upon a performance characteristic of the specific storage device;

ordering the plurality of storage devices in a hierarchical fashion based on bandwidth of each storage device among the plurality of storage devices;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0537
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

optimizing bandwidth from each storage device among the plurality of storage devices by dynamically relocating portions of media content stored on each storage device based on usage measurements of the storage device.

28. The method of Claim 27, wherein the storing step stores the at least a portion of the media content in the specific storage device based additionally upon a target flow rate for the media flow associated with the media content.

29. The method of Claim 27, further comprising:

relocating a portion of a media content from a first storage device to a second storage device having better performance characteristics than the first storage device based upon an increase in popularity of the portion of the media content.

30. The method of Claim 27, further comprising:

relocating a portion of a media content from a first storage device to a second storage device having lower performance characteristics than the first storage device based upon a decrease in popularity of the portion of the media content.

31. The method of Claim 27, further comprising:

storing multiple portions of media content that include the at least a portion of the media content in the specific storage device when the multiple portions fit in a specifically sized block of memory on the specific storage device.

32. A computer-readable storage medium carrying one or more sequences of instructions for simultaneously distributing media across a network to a plurality of client systems, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of:

storing at least a portion of a media content in a specific storage device, among a plurality of storage devices with at least a portion of the plurality of storage devices having differing performance characteristics, based at least upon a performance characteristic of the specific storage device;

ordering the plurality of storage devices in a hierarchical fashion based on bandwidth of each storage device among the plurality of storage devices;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0538
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

optimizing bandwidth from each storage device among the plurality of storage devices by dynamically relocating portions of media content stored on each storage device based on usage measurements of the storage device.

33. The computer-readable storage medium of Claim 32, wherein the storing step stores the at least a portion of the media content in the specific storage device based additionally upon a target flow rate for the media flow associated with the media content.

34. The method of Claim 32, further comprising:

relocating a portion of a media content from a first storage device to a second storage device having better performance characteristics than the first storage device based upon an increase in popularity of the portion of the media content.

35. The computer-readable storage medium of Claim 32, further comprising:

relocating a portion of a media content from a first storage device to a second storage device having lower performance characteristics than the first storage device based upon a decrease in popularity of the portion of the media content.

36. The computer-readable storage medium of Claim 32, further comprising:

storing multiple portions of media content that include the at least a portion of the media content in the specific storage device when the multiple portions fit in a specifically sized block of memory on the specific storage device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0539
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

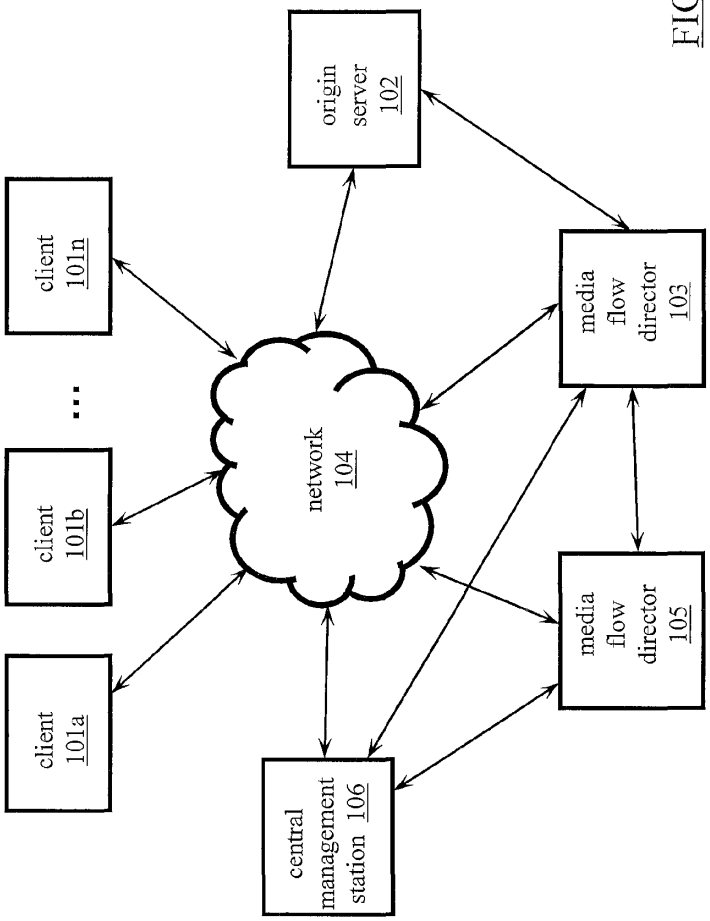


FIG. 1

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0540
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Component and Data Flow Architecture

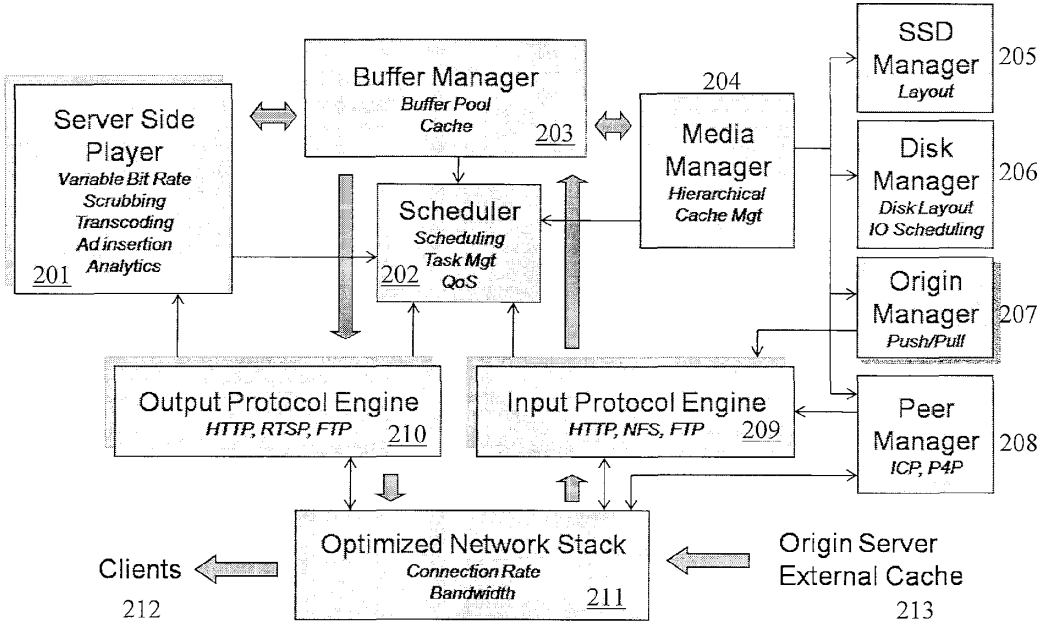


FIG. 2

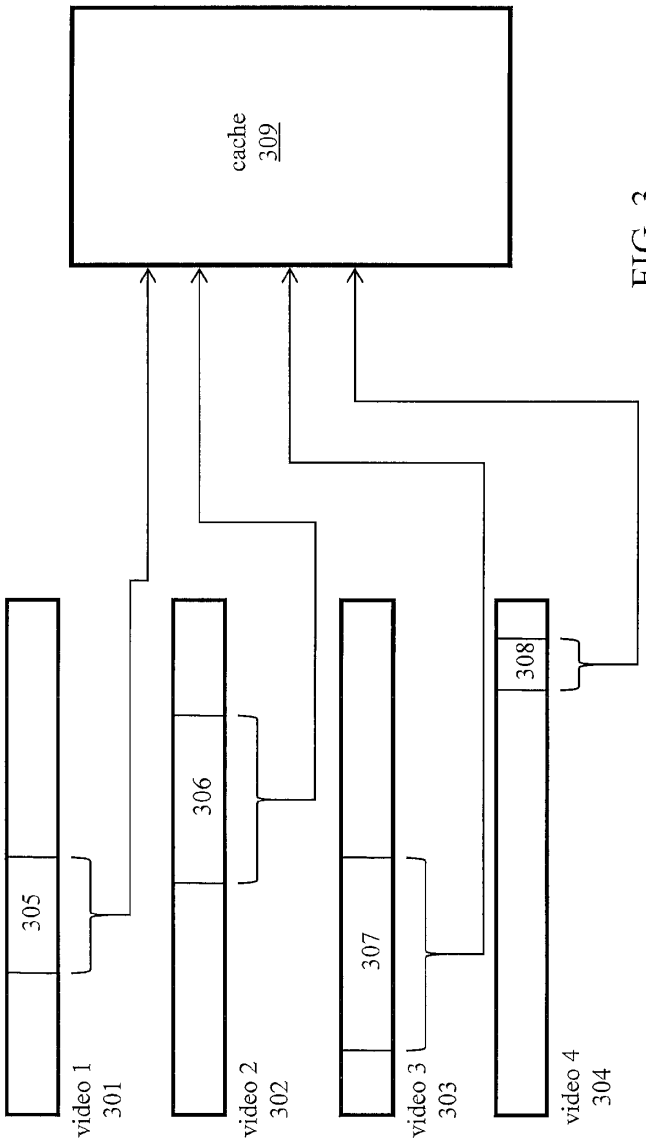


FIG. 3

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0542
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

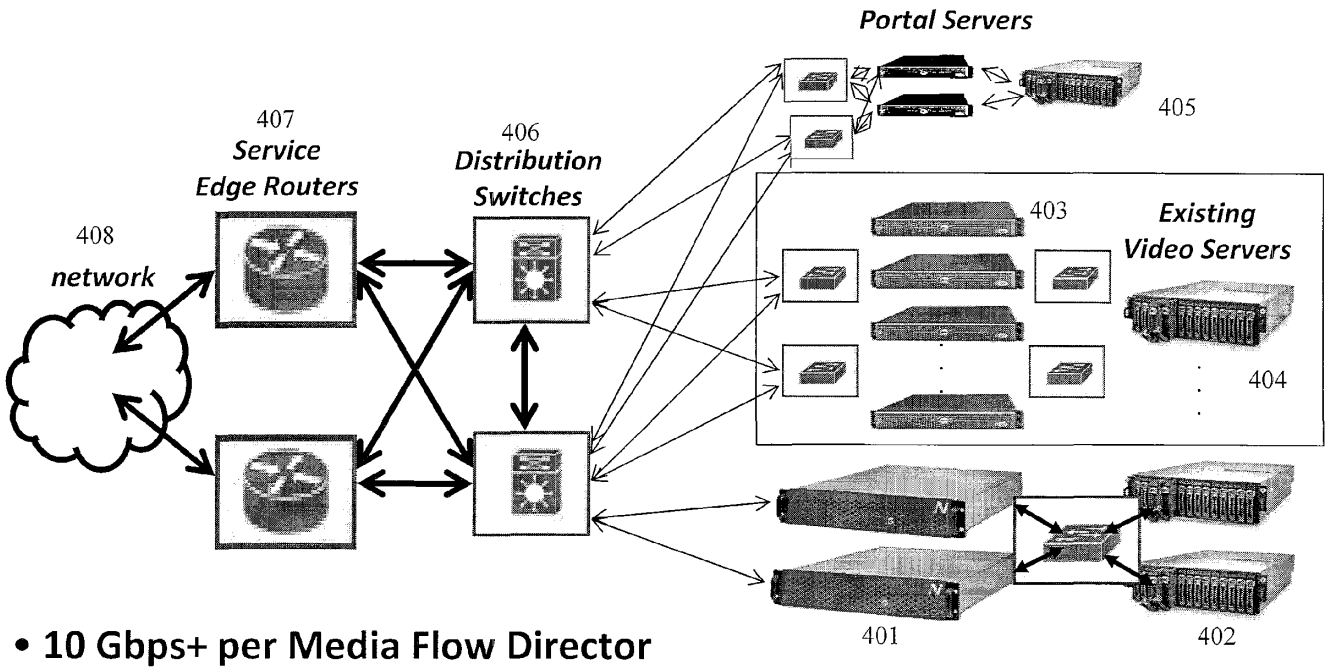

Portal Servers
405
407
Service
Edge Routers
406
Distribution
Switches
408
network
403
Existing
Video Servers
404
401
402
• 10 Gbps+ per Media Flow Director
• Scale up with 2 MFD instead of 20 servers

FIG. 4

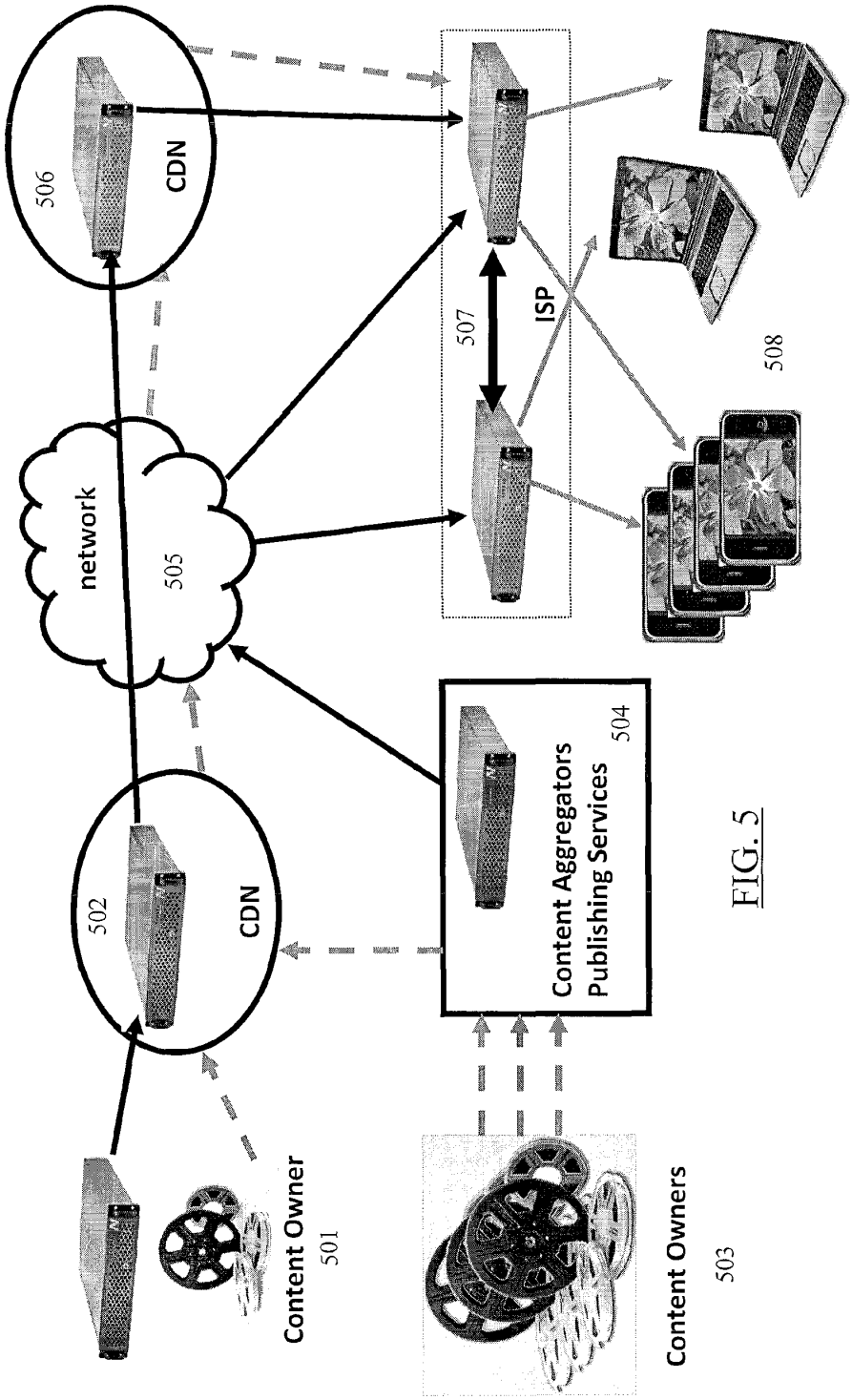
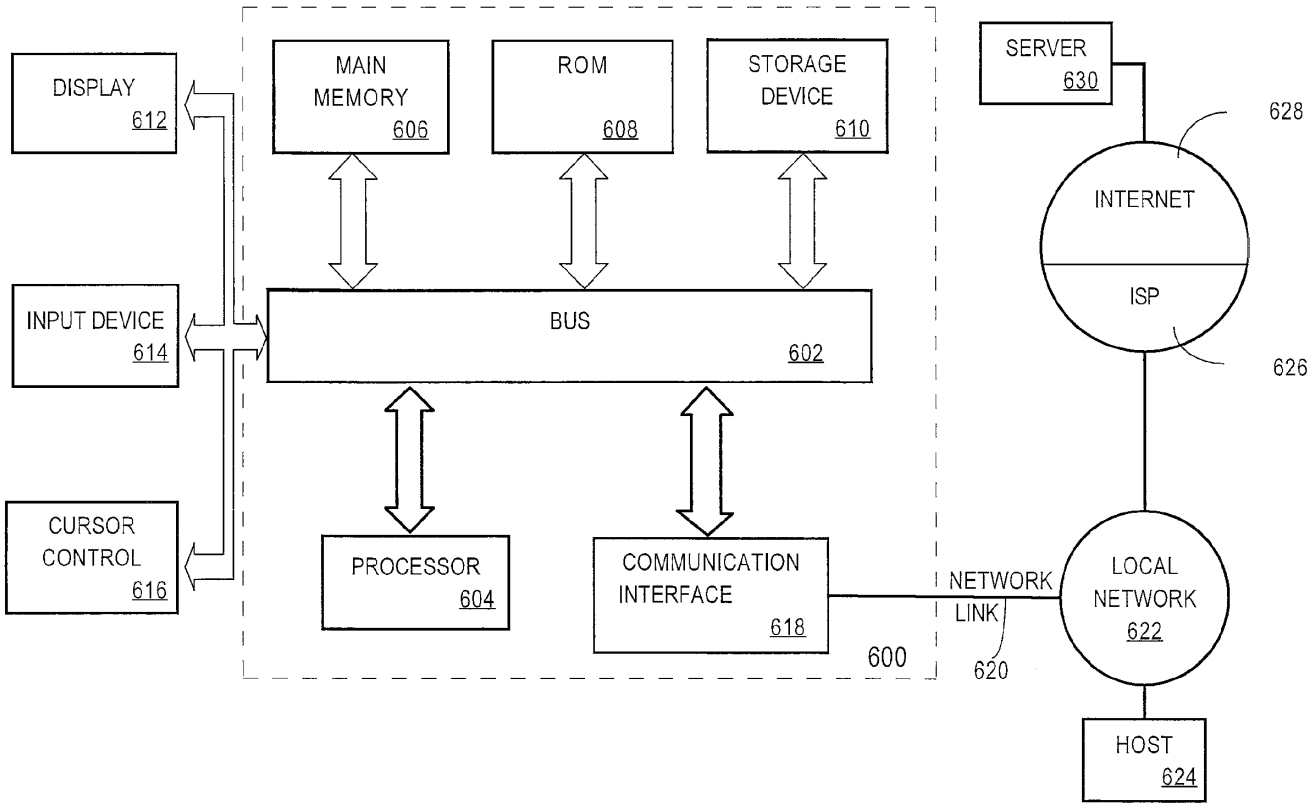

CDN
506
network
505
CDN
502
ISP
507
508
Content Aggregators
Publishing Services
504
Content Owner
501
Content Owners
503

FIG. 5

FIG. 6


DISPLAY
612
INPUT DEVICE
614
CURSOR
CONTROL
616
MAIN
MEMORY
606
ROM
608
STORAGE
DEVICE
610
BUS
602
PROCESSOR
604
COMMUNICATION
INTERFACE
618
600
NETWORK
LINK
620
SERVER
630
628
INTERNET
ISP
626
LOCAL
NETWORK
622
HOST
624

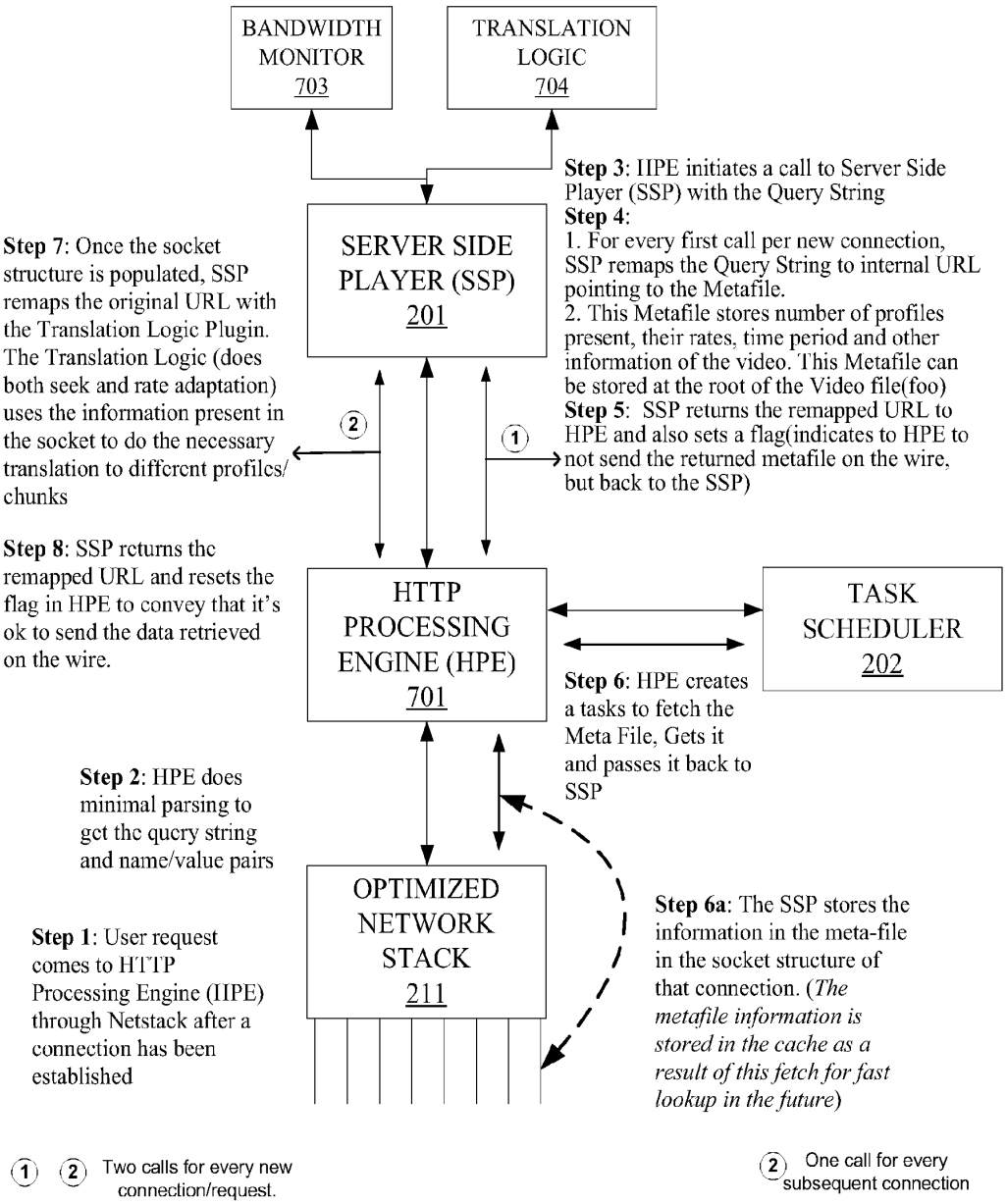


Fig. 7

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0546
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

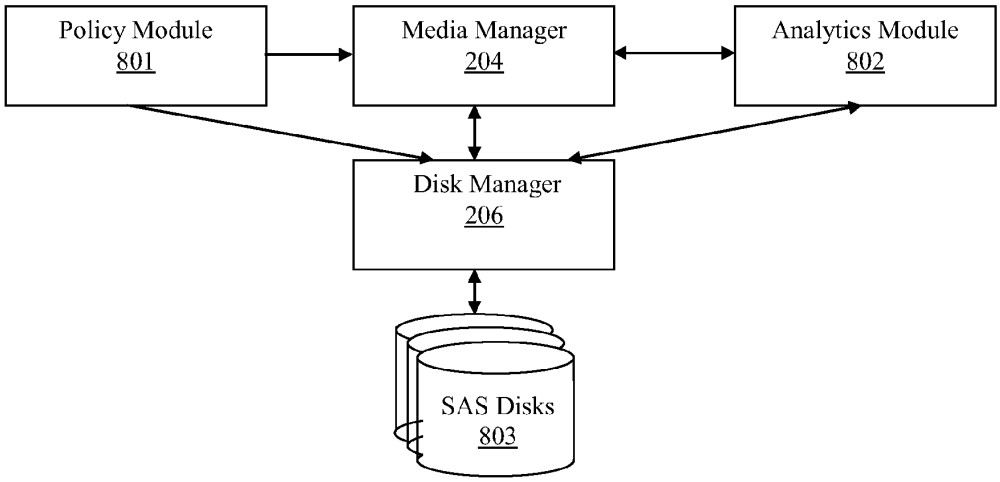


Fig. 8a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0547
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

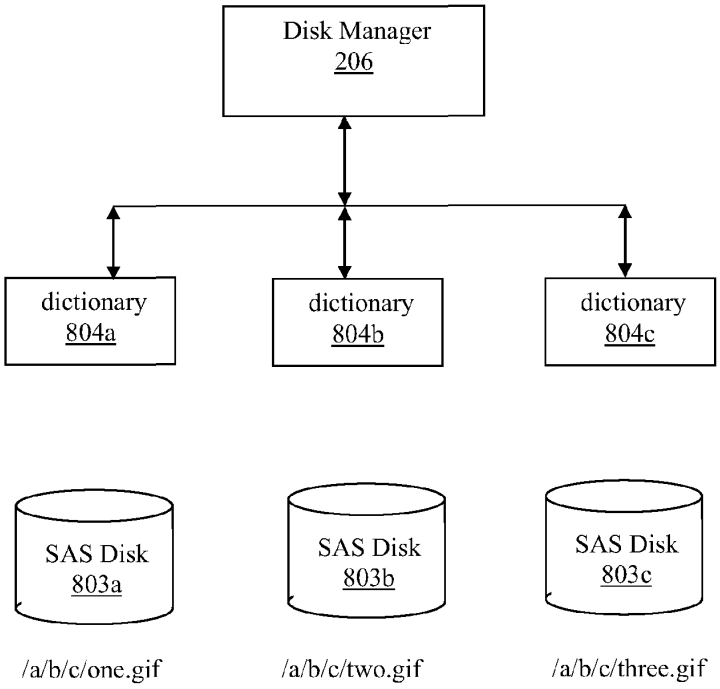


Fig. 8b

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0548
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

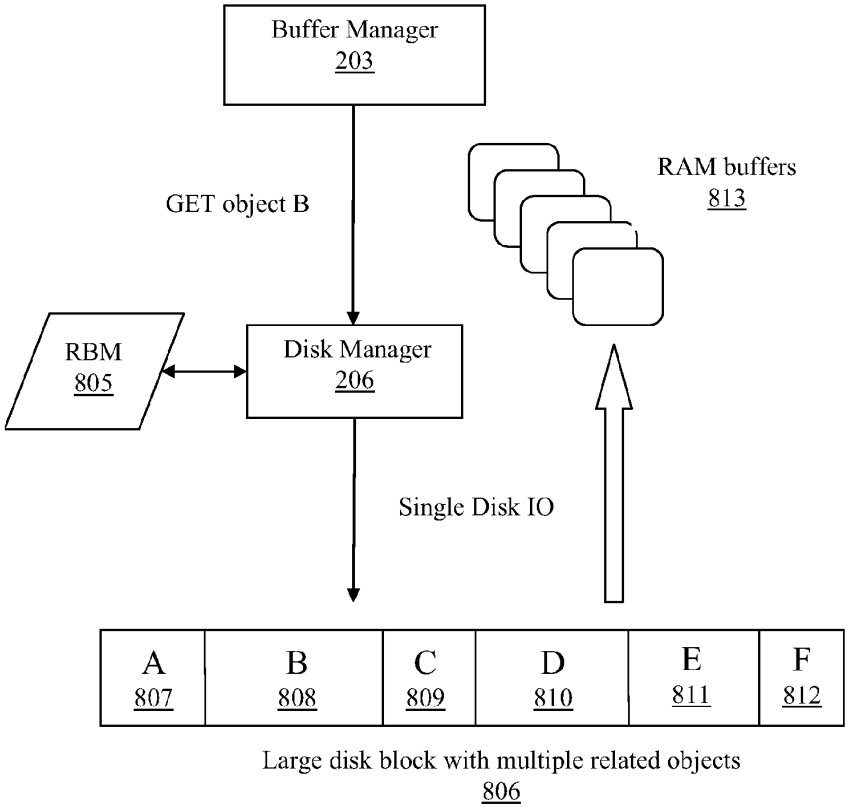


Fig. 8c

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0549
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

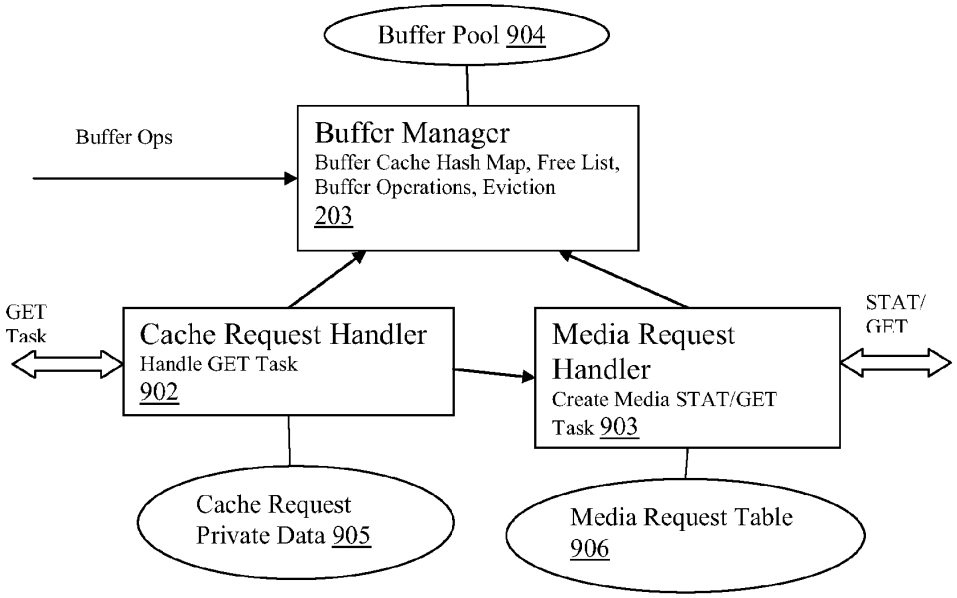


Fig. 9

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0550
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

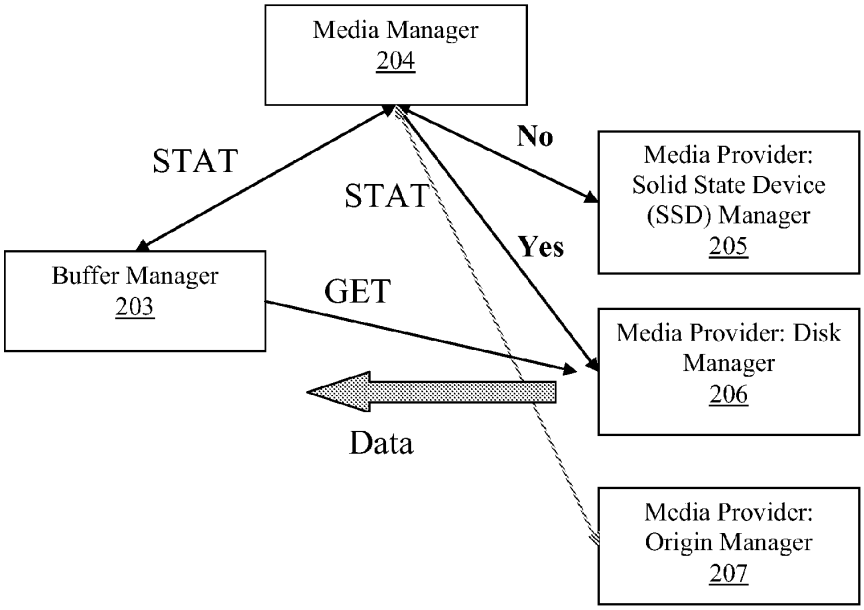


Fig. 10a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0551
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

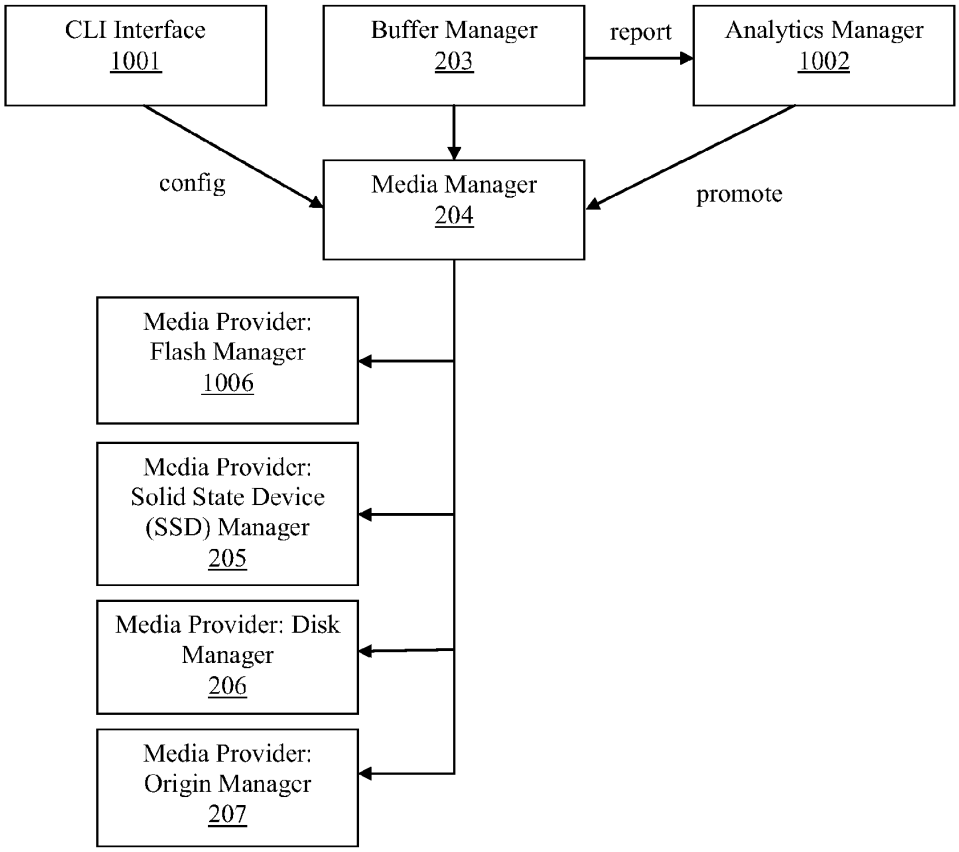


Fig. 10b

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0552
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

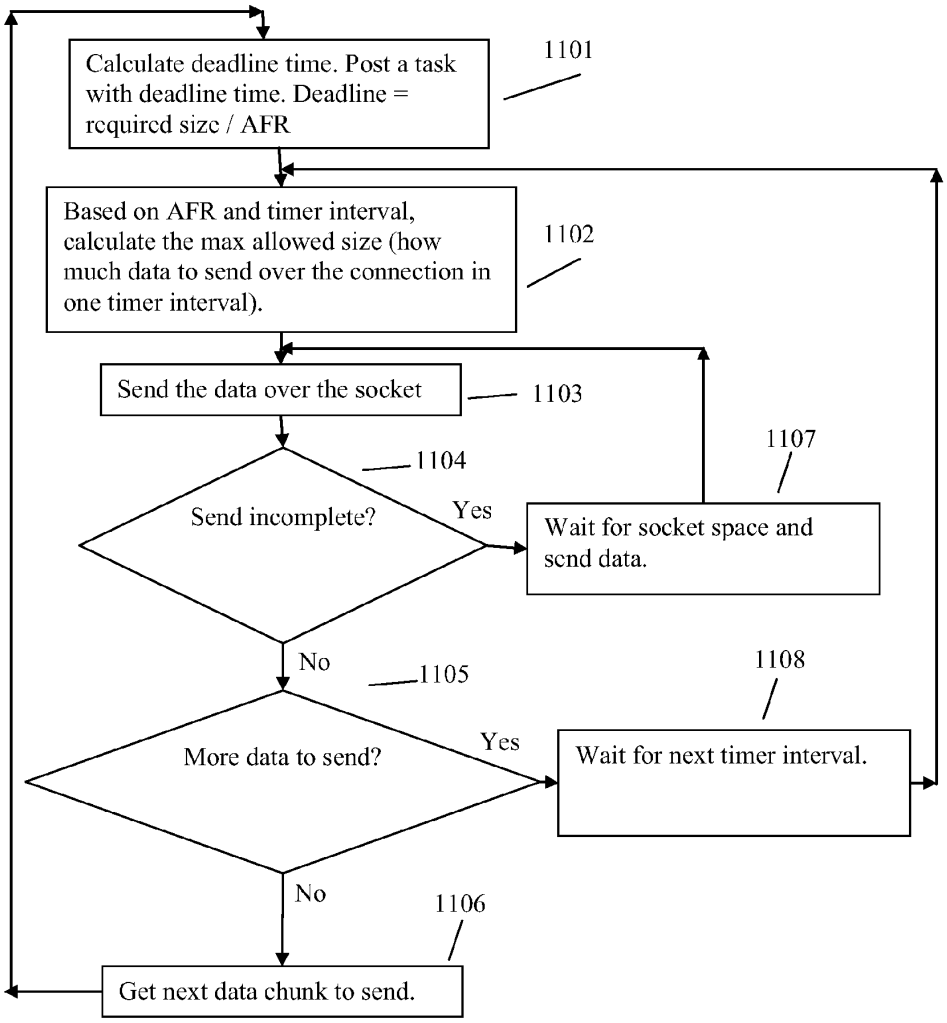


Fig. 11

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0553
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

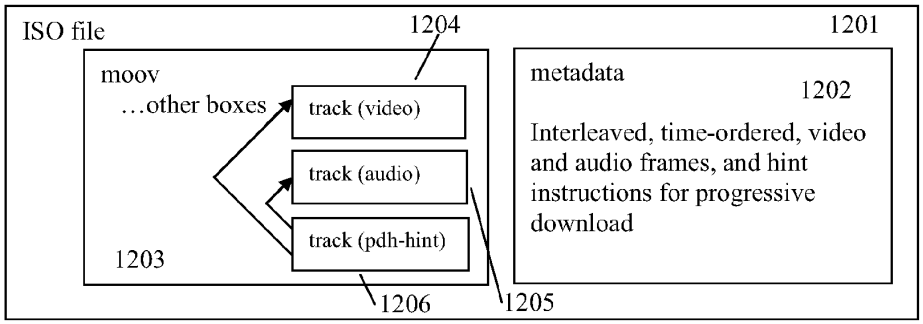


Fig. 12

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0554
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

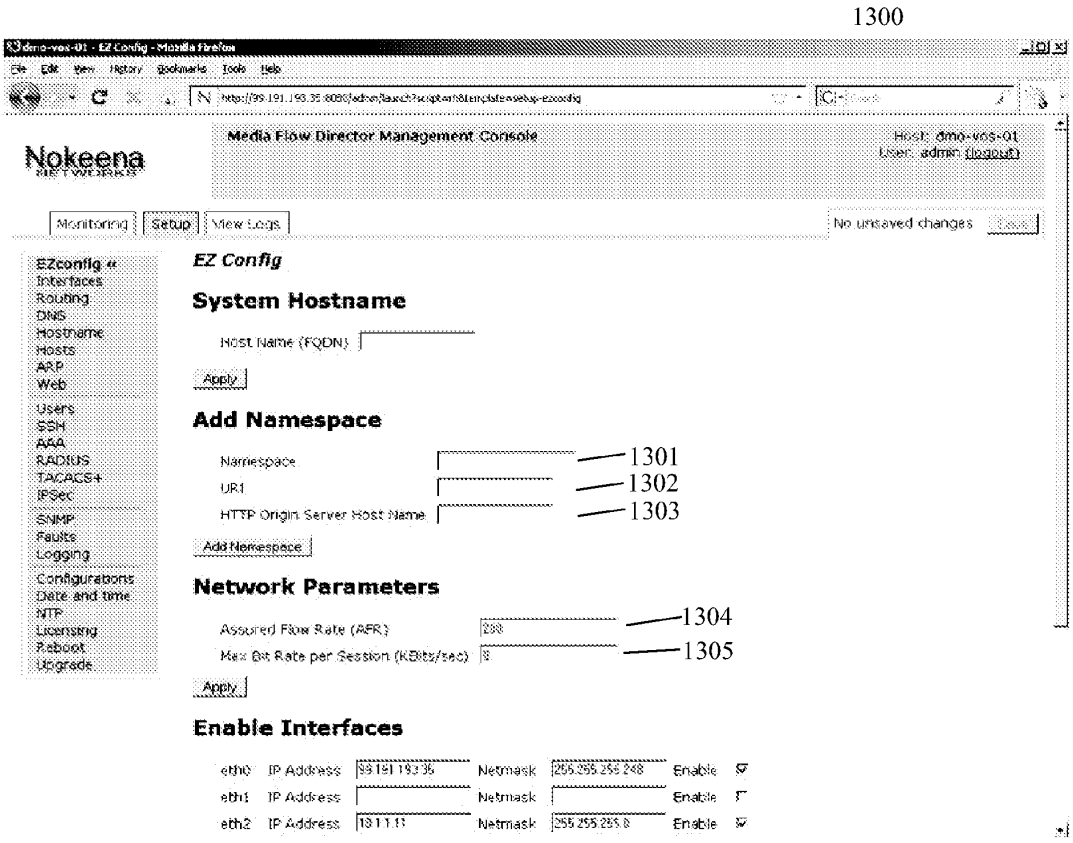


Fig. 13

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0555
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

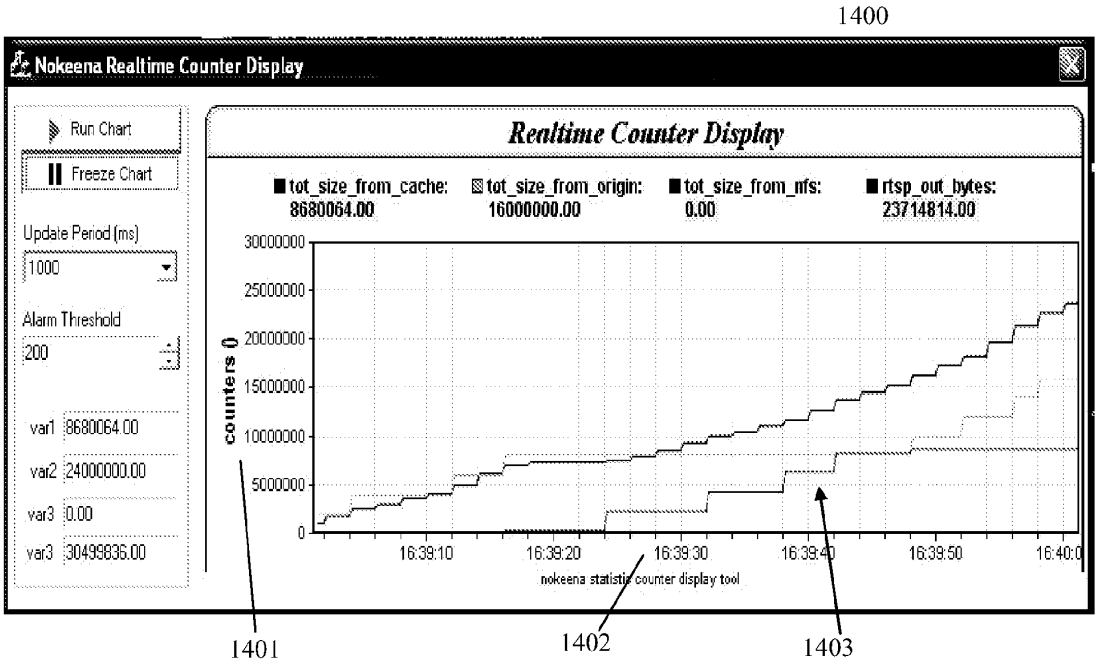


Fig. 14

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0556
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

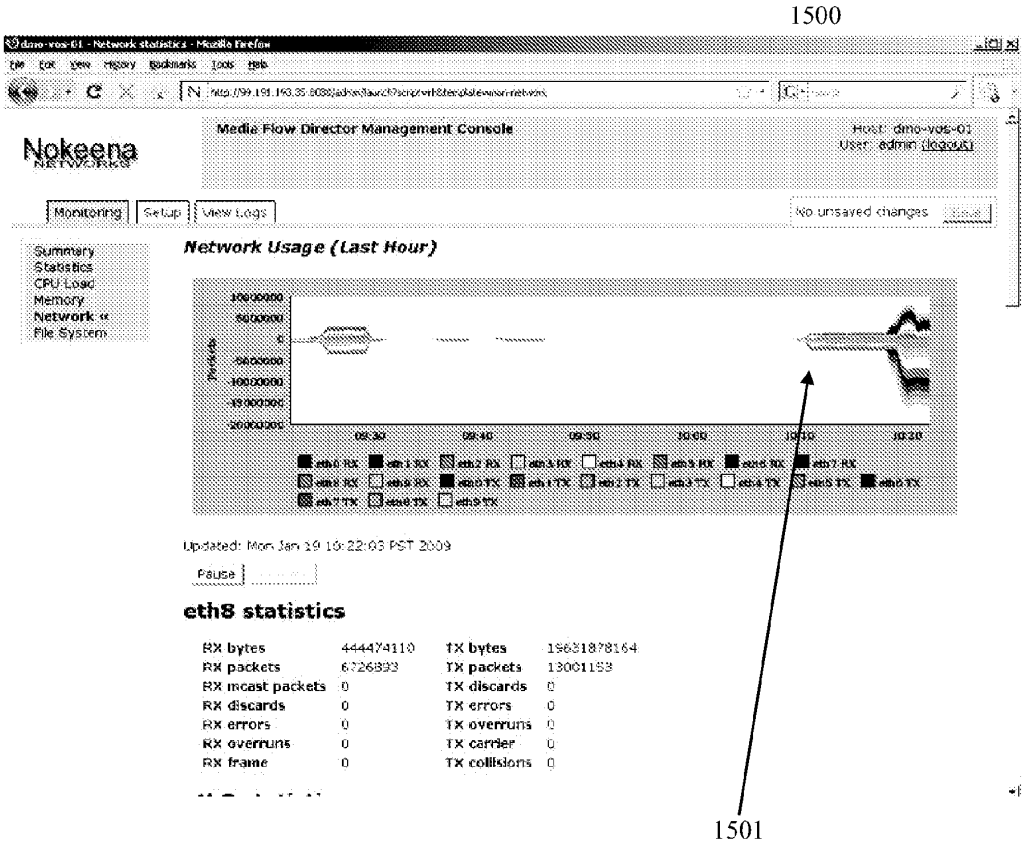


Fig. 15

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0557
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2009/065805

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04L 12/26 (2010.01)
USPC - 709/234
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8) - H04J 1/16, 3/14; H04L 12/00, 12/26 (2010.01)
USPC - 370/229, 370/230, 232, 233, 234; 709/226, 231, 232, 233, 234

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PatBase, Google Patents, MicroPatent

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2002/0029274 A1 (ALLEN) 07 March 2002 (07.03.2002) entire document	1-21, 23, 28 and 33
Y	US 6,650,618 B1 (PENG et al) 18 November 2003 (18.11.2003) entire document	1-21
Y	US 2003/0099254 A1 (RICHTER) 29 May 2003 (29.05.2003) entire document	2-4, 9-11, and 16-18
X -- Y	US 2005/0267948 A1 (MCKINLEY et al) 01 December 2005 (01.12.2005) entire document	22, 24-27, 29-32, and 34-36 ----- 5, 12, 19, 23, 28 and 33

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 March 2010	08 APR 2010
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2009/065805

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

See Extra Sheet

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☒ No protest accompanied the payment of additional search fees.

Form PCT/ISA/210 (continuation of first sheet (2)) (July 2009)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2009/065805

Contnuation of Box No. III:

This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fees must be paid.

Group I, claims 1-21, drawn to an apparatus, media delivery scheduling in a method or a computer-readable storage medium carrying a sequence of instructions for simultaneously distributing media across network to a plurality of client systems.
Group II, claims 22-36, drawn to an apparatus, a media caching subsystem in a method or a computer-readable storage medium carrying a sequence of instructions for simultaneously distributing media across network to a plurality of client systems.

The inventions listed as Groups I-II do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons: the special technical feature of the Group I invention: a storage optimization subsystem that optimizes bandwidth from each storage device among the plurality of storage devices by dynamically relocating portions of media content stored on each storage device based on usage measurements of the storage device as claimed therein is not present in the invention of Group II. The special technical feature of the Group II invention: dynamically adjusting flow rate for each outbound media flow in order to meet the target flow rate for each outbound media flow, and refusing incoming media requests when upon detecting that target flow rates for media flows may not be achieved if additional media content requests are accepted as claimed therein is not present in the invention of Groups I.

Since none of the special technical features of the Group I or II inventions are found in more than one of the inventions, unity of invention is lacking.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0560
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 October 2010 (28.10.2010)

PCT

(10) International Publication Number
WO 2010/122447 A1

(51) International Patent Classification:
H04N 5/445 (2006.01) *G06F 17/30* (2006.01)
G06K 9/00 (2006.01) *H04N 7/173* (2006.01)

(21) International Application Number:
PCT/IB2010/051588

(22) International Filing Date:
13 April 2010 (13.04.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09158248.6 20 April 2009 (20.04.2009) EP

(71) Applicant *(for all designated States except US)*: **KONINKLIJKE PHILIPS ELECTRONICS N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **FOO, Teck, W.** [MY/SG]; c/o High Tech Campus Building 44, NL-5656 AE Eindhoven (NL). **LOI, Tek, S.** [MY/MY]; c/o High Tech Campus Building 44, NL-5656 AE Eindhoven (NL).

(74) Agents: **COOPS, Peter** et al.; Philips Intellectual Property & Standards, P.O. Box 220, NL-5600 AE Eindhoven (NL)

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

---- *with international search report (Art. 21(3))*

(54) **Title**: VERIFICATION AND SYNCHRONIZATION OF FILES OBTAINED SEPARATELY FROM A VIDEO CONTENT

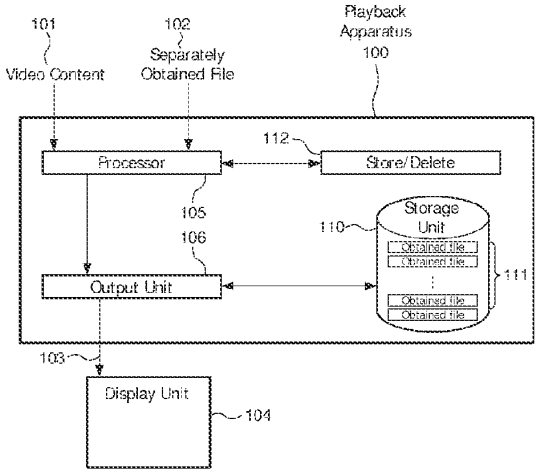


FIG. 1

(57) **Abstract**: The invention relates to a playback device for playback of a video content and of a file obtained separately from the video content. The invention is for a system wherein a file to be played with a movie is obtained from an uncontrolled source such as the Internet. Obtaining a file from such a source creates a number of issues due to the lack of control. The file obtained may not correspond to the intended movie or to the intended language and may not be synchronized with the movie. The invention proposes: 1) to verify that the separately obtained file corresponds to the movie; 2) if the separately obtained file corresponds to the movie, to determine whether it is synchronized with the movie; 3) if the separately obtained file is not synchronized with the movie, to synchronize it. The correspondence can be verified based on language, key phrases relating to the movie, or timing.

WO 2010/122447 A1

Verification and synchronization of files obtained separately from a video content

FIELD OF THE INVENTION

The present invention relates to playback of a video content and of files obtained separately.

BACKGROUND OF THE INVENTION

Subtitle files for movies can now be obtained separately from the movie with original audio track. The user that already has a movie with original audio track can search for and obtain subtitle files in other languages than those given with the movie. The same can be expected for audio files. Separate files for a movie can be downloaded from uncontrolled sources such as the Internet. Files related to a movie can be searched for and downloaded from a generic web site. Obtaining audio or subtitle files from such sources creates a number of issues due to the lack of control. The obtained files may not correspond to the movie that the user wants to watch, or to the intended language. The obtained files may also not be synchronized with the movie. To verify the correspondence between the obtained file and the movie, the user has to play the obtained file with the movie. The user has to repeat this verification for each file obtained until the intended file is found. Once the intended file is found, the user has to check the synchronization of the obtained file with the movie and to synchronize the obtained file manually if it is not synchronized with the movie. These are tedious and time consuming tasks for the user.

SUMMARY OF THE INVENTION

The inventors of the present invention have appreciated that for files obtained separately from a video content, the solution to the above mentioned problems and possibly other problems is to first verify if the separately obtained files correspond to the video content and then to synchronize the files if needed.

Accordingly, the invention preferably seeks to mitigate, alleviate or eliminate one or more of the above mentioned disadvantages singly or in any combination.

The object of the present invention is obtained in a first aspect of the invention by providing a playback apparatus comprising:

- Means for playback of a video content including a video file and an audio file, and of a file obtained separately from the video content; and
- Means for determining whether the separately obtained file has a predefined correspondence to the video content; and
- Means for determining, when it has been determined that the separately obtained file has the predefined correspondence to the video content, whether an element of the separately obtained file is synchronized with a corresponding element of the audio file of the video content; and
- Means for synchronizing, if the element of the corresponding separately obtained file is not synchronized with the corresponding element in the audio file of the video content, the element of the separately obtained file with the corresponding element of the audio file of the video content.

The invention is particularly, but not exclusively, advantageous for use in connection with playback of multimedia content in the DVD format, DivX format, Blu-Ray format, and MKV format. The MKV format is the Video version of the Matroska Multimedia Container that is an open standard free container format that can hold an unlimited number of video, audio, picture or subtitle tracks inside a single file. Matroska file types are .MKV for video (with subtitles and audio), .MKA for audio-only files and .MKS for subtitles only.

It is an advantage of the present invention, that an effective way to handle files obtained separately is provided by first determining whether the separately obtained file has a predefined correspondence to the video content before proceeding further, so that the user need not to playback in vain each file obtained separately to determine a correspondence. Second, elements of the separately obtained file with the audio file of the video content are analyzed to determine and to synchronize non-synchronized elements, so that the user need not to search for another file or edit manually the obtained file to synchronize it with the video content.

In a second aspect, a method for verification and synchronization of a file obtained separately from a video content in accordance with the first aspect is provided. The method of the second aspect may in a third aspect be implemented in an integrated circuit (IC) for operating a playback apparatus in accordance with the first aspect.

In a fourth aspect, the invention relates to a computer program product being adapted to enable a computer system comprising at least one computer having data storage means associated therewith to control a playback apparatus according to the first aspect of the invention.

This aspect of the invention is particularly, but not exclusively, advantageous in that the present invention may be implemented by a computer program product enabling a computer system to perform the operations of the second aspect of the invention. Thus, it is contemplated that a playback apparatus may be changed to operate according to the present invention by installing a computer program product on a computer system controlling the playback apparatus. Such a computer program product may be provided on any kind of computer readable medium, e.g. magnetically or optically based medium, or through a computer based network, e.g. the Internet.

The various aspect of the present invention may each be combined with any of the other aspects. These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

BRIEF DESCRIPTION OF THE FIGURES

The present invention will now be explained, by way of example only, with reference to the accompanying Figures, where:

Fig. 1 is a schematic view of a preferred implementation of a playback apparatus according to the present invention.

Fig. 2 is a flow-chart of a preferred implementation for operating a playback apparatus for playback of a video content and a separately obtained file.

Fig. 3 is a flow-chart for determining, based on a language analysis, a predefined correspondence of a separately obtained file with the video content.

Fig. 4 is a flow-chart for determining, based on a retrieved key phrase, a predefined correspondence of a separately obtained file with the video content.

Fig. 5 is a flow-chart for determining, based on timing, a predefined correspondence of a separately obtained file with the video content.

Fig. 6A is a flow-chart for determining whether an element of a separately obtained file is synchronized with a corresponding element of the audio file of the video content.

Fig. 6B is a flow-chart of a preferred implementation for synchronization based on modification of a timecode of a non-synchronized element of the separately obtained file.

Fig. 6C is a flow-chart of a preferred implementation for synchronization based on attribution of a timecode.

DETAILED DESCRIPTION OF AN EMBODIMENT

Referring to Figure 1, a playback apparatus is shown therein, the playback apparatus being generally referred to by reference number 100. The playback apparatus 100 accepts a video content 101 and a separately obtained file 102. The video content includes a video file, one or more audio files (e.g. original audio file and audio files in other languages) and possibly one or more closed captions such as subtitle files. These files include elements marked with timecodes where elements are for example text elements for a subtitle file, or sound elements for an audio file. Elements of e.g. a subtitle file are individually synchronized to corresponding elements in an included audio file. The user may select to play the video content with the original audio file and a subtitle file that is in a language (e.g. French) different from the language (e.g. English) in the original audio file. The user may also select to play the video content with an audio file that is in a language (e.g. French) different from the language (e.g. English) in the original audio file.

The user may also decide to search for files that are not included in the video content in the user's possession. For example, the user may want to see the video content with subtitle files or audio files in a language (e.g. Italian) which is not included in the video content. The user would then search for and obtain separately subtitle files or audio files in the desired language (e.g. Italian) for the video content to be viewed.

The separately obtained file can be a subtitle file, a closed caption file or an audio file. The playback apparatus 100 comprises an output unit 106 that outputs a video and audio signal 103 to a viewing means, such as a display unit 104. The display unit 104 can be disposed remotely from the playback apparatus 100 as illustrated, or integrated with the playback apparatus 100. The playback apparatus has a processor 105 for controlling components thereof and carrying out instructions contained on hardware or software in the playback apparatus 100 or remote there from. The playback apparatus has a storage unit 110, such as a hard drive, for storing files thereon. The stored files comprise a plurality of files 111 obtained separately from the video content 101 through various sources such as Internet. The storage unit 110 is under the control of a store/delete agent 112 which instructs the storage unit to store or delete a given file. The store/delete agent 112 is under the control of the processor 105, and may be integrated therewith. In addition to the typical function of a processor 105 for a typical playback apparatus, the added functionalities of the processor will be described below in detail with regards to the present invention.

Referring to Figure 2, there is a flowchart showing the main functionalities for verification and synchronization of separately obtained files. At step 201, a file is obtained

separately from the video content. At step 202, the processor 105 stores the separately obtained file on the storage unit 110. The user can instruct the playback apparatus to store the separately obtained file. At step 203, the processor 105 determines whether the separately obtained file has a predefined correspondence to the video content. Various methods for determination of predefined correspondence will be described below in detail. If it is determined that the separately obtained file does not have a predefined correspondence to the video content, the processor 105 proceeds along path 203-NO to step 204 where the separately obtained file is deleted from the storage unit 110 and another file is obtained separately (step 201). In case of a plurality of files obtained separately from the video content, the plurality of separately obtained files is stored on the storage unit 110, and the processor105 proceeds to the next separately obtained file that is stored for playback with the video content.

If it is determined that the separately obtained file has a predefined correspondence to the video content, the processor 105 proceeds along path 203-YES to step 205. At step 205, the processor 105 determines whether an element of the separately obtained file is synchronized with a corresponding element of the audio file of the video content. A preferred method for determining the synchronization between elements of the separately obtained file and corresponding elements of the audio file of the video content is described later. If it is determined that the separately obtained file does not contain any element that is not synchronized with a corresponding element of the audio file of the video content, the processor 105 proceeds along path 205-YES to end.

If it is determined that the separately obtained file contains an element of the separately obtained file that is not synchronized with a corresponding element of the audio file of the video content, the processor 105 proceeds along path 205-NO to step 206. At step 206, the processor 105 synchronizes the non-synchronized element of the separately obtained file with the corresponding element in the audio file of the video content. Synchronization relates to the time coordination of the elements in both files. A threshold of tolerance is taken into account in determining and performing synchronization of elements. Methods for synchronizing non-synchronous elements of the separately obtained file are described later.

Referring to Figure 3, there is a flowchart showing a preferred implementation of the functionality for determining a predefined correspondence of a separately obtained file based on an intended language. At step 301, the processor 105 selects a separately obtained file stored on the storage unit 110. At step 302, the user selects an intended language for which the separately obtained file should be verified. At step 303, the processor 105 analyzes

the language setting of the elements of the separately obtained file and proceeds to step 304 where it detects a predominant language present in the separately obtained file. At step 305, the processor determines whether the predominant language detected is the same as the intended language. If it is determined that the predominant language detected is not the same as the intended language, the processor 105 proceeds along path 305-NO to step 306 where the separately obtained file is marked as having no predefined correspondence to the video content. If it is determined that the predominant language detected is the same as the intended language, the processor 105 proceeds along path 305-YES to step 307 where the separately obtained file is marked as having the predefined correspondence to the video content.

Referring to Figure 4, there is a flowchart showing a preferred implementation of the functionality for determining a predefined correspondence of a separately obtained file based on a retrieved key phrase relating to the video content. At step 401, the processor 105 selects a separately obtained file stored on the storage unit 110. At step 402, the processor 105 retrieves a key phrase relating to the video content. The key phrase could be retrieved from a separate source of information like a website on the Internet or from any file included in the video content (e.g. audio file, subtitle file, extra features). At step 403, the processor 105 searches for the retrieved key phrase in the separately obtained file. At step 404, the processor determines whether the retrieved key phrase is present in the separately obtained file. If it is determined that the retrieved key phrase is not present in the separately obtained file, the processor 105 proceeds along path 404-NO to step 405 where the separately obtained file is marked as having no predefined correspondence to the video content. If it is determined that the retrieved key phrase is present in the separately obtained file, the processor 105 proceeds along path 404-YES to step 406 where the separately obtained file is marked as having the predefined correspondence to the video content.

Referring to Figure 5, there is a flowchart showing a preferred implementation of the functionality for determining a predefined correspondence of a separately obtained file based on timing. At step 501, the processor 105 selects a separately obtained file stored on the storage unit 110. At step 502, the processor 105 extracts timing of an element of the separately obtained file. At step 503, the processor 105 compares the extracted timing with timing of a corresponding element of the audio file of the video content. At step 504, the processor 105 determines whether the result of the comparison establishes a correspondence between the separately obtained file and the video content. For example, if the comparison result reveals a difference in timing that is within predefined limits, then the correspondence between both files is established; else the correspondence is not established. If it is

determined that the result of the comparison does not establish a correspondence between the separately obtained file and the video content, the processor 105 proceeds along path 504-NO to step 505 where the separately obtained file is marked as having no predefined correspondence to the video content. If it is determined that the result of the comparison establishes a correspondence between the separately obtained file and the video content, the processor 105 proceeds along path 504-YES to step 506 where the separately obtained file is marked as having the predefined correspondence to the video content.

Referring to Figure 6A, there is a flowchart showing a preferred implementation of the functionality for determining whether an element of a separately obtained file is synchronized with a corresponding element of the audio file of the video content. At step 601, the processor 105 accesses an element, denoted by the index $X_{obtained\_file}$, of the separately obtained file to be examined. At step 602, the processor 105 extracts the timecode $TC_{X\_obtained\_file}$ of the element of the separately obtained file. At step 603, the processor 105 extracts the timecode $TC_{X\_audio\_file}$ of the corresponding element X_{audio_file} of the audio file of the video content. At step 604, the processor 105 calculates the time difference T_{diff} between the timecode of the element of the separately obtained file and the timecode of the corresponding element of the audio file of the video content:

$$T_{diff} = TC_{X\_obtained\_file} - TC_{X\_audio\_file}$$

At step 605, the processor 105 determines whether the absolute time difference T_{diff} calculated is below a predefined threshold. If it is determined that the absolute time difference calculated is below the predefined threshold, the processor 105 proceeds along path 605-YES to step 606 where the element $X_{obtained\_file}$ of the separately obtained file is marked as synchronized to the corresponding element X_{audio_file} of the audio file of the video content. If it is determined that the absolute time difference calculated is not below the predefined threshold, the processor 105 proceeds along path 605-NO to routine 620. Routine 620 is shown in Figure 6B and 6C depicting two alternative techniques preferred for synchronizing non-synchronized elements.

At step 607, the processor 105 determines whether there are any remaining elements of the separately obtained file to be examined. If it is determined that there are remaining elements in the separately obtained file, the processor 105 proceeds along path 607-YES to step 608 where the index $X_{obtained\_file}$ of the element of the separately obtained

file is incremented by 1 to equal the index of the next element in the sequence of the separately obtained file.

Referring to Figure 6B, routine 620 is illustrated therein. At step 621, the processor 105 modifies the timecode $TC_{X\_obtained\_file}$ of the element of the separately obtained file so that time difference between $TC_{X\_obtained\_file}$ and $TC_{X\_audio\_file}$ is within the predefined threshold. The processor 105 then proceeds to step 606.

Referring to Figure 6C, an alternative routine 620 is illustrated therein. At step 631, the processor 105 detects the timecode $TC_{X\_audio\_file}$ of the element X_{audio_file} of the audio file of the video content that corresponds to the accessed element $X_{obtained\_file}$ of the separately obtained file. At step 632, the processor 105 attributes the detected timecode $TC_{X\_audio\_file}$ to the corresponding element $X_{obtained\_file}$ of the separately obtained file. The processor 105 then proceeds to step 606.

The invention can be implemented in any suitable form including hardware, software, firmware or any combination of these. The invention or some features of the invention can be implemented as computer software running on one or more data processors and/or digital signal processors. The elements and components of an embodiment of the invention may be physically, functionally and logically implemented in any suitable way. Indeed, the functionality may be implemented in a single unit, in a plurality of units or as part of other functional units. As such, the invention may be implemented in a single unit, or may be physically and functionally distributed between different units and processors.

Although the present invention has been described in connection with the specified embodiments, it is not intended to be limited to the specific form set forth herein. Rather, the scope of the present invention is limited only by the accompanying claims. In the claims, the term "comprising" does not exclude the presence of other elements or steps. Additionally, although individual features may be included in different claims, these may possibly be advantageously combined, and the inclusion in different claims does not imply that a combination of features is not feasible and/or advantageous. In addition, singular references do not exclude a plurality. Thus, references to "a", "an", "first", "second" etc. do not preclude a plurality. Furthermore, reference signs in the claims shall not be construed as limiting the scope.

CLAIMS:

1. A Playback apparatus comprising:

- Means for playback of a video content including a video file and an audio file, and of a file obtained separately from the video content; and

- Means for determining whether the separately obtained file has a predefined correspondence to the video content; and

- Means for determining, when it has been determined that the separately obtained file has the predefined correspondence to the video content, whether an element of the separately obtained file is synchronized with a corresponding element of the audio file of the video content; and

- Means for synchronizing, when it has been determined that the element of the corresponding separately obtained file is not synchronized with the corresponding element in the audio file of the video content, the element of the separately obtained file with the corresponding element of the audio file of the video content.

2. The apparatus according to claim 1, wherein the separately obtained file from the video content is a subtitle file, a closed caption file, or an audio file.

3. The apparatus according to claim 1, wherein the means for determining whether the separately obtained file has a predefined correspondence to the video content comprise:

- Means for analyzing a language setting of the separately obtained file; and

- Means for detecting a language predominantly present in the separately obtained file.

4. The apparatus according to claim 3, wherein the means for detecting a language predominantly present in the separately obtained file further comprise means for detecting whether an intended language is the language detected as predominantly present in the separately obtained file.

5. The apparatus according to claim 1, wherein the means for determining whether the separately obtained file has a predefined correspondence to the video content comprise:

- Means for comparing a timing of an element of the separately obtained file with a timing of a corresponding element in the audio file of the video content; and
- Means for analyzing results of the comparison to establish a correspondence between the separately obtained file and the video content.

6. The apparatus according to claim 1, wherein the means for determining whether the separately obtained file has a predefined correspondence to the video content comprise:

- Means for retrieving a key phrase relating to the video content; and
- Means for detecting whether the retrieved key phrase is present in the separately obtained file.

7. The apparatus according to claim 1, wherein the means for determining whether an element of the separately obtained file is not synchronized with a corresponding element of the audio file of the video content comprise:

- Means for calculating a time difference between a timecode of the element of the separately obtained file and a timecode of the corresponding element of the audio file of the video content; and
- Means for comparing the time difference calculated with a predefined threshold, wherein if the absolute time difference calculated is not below the predefined threshold then the element of the separately obtained file is determined to be non-synchronized with the corresponding element of the audio file of the video content.

8. The apparatus according to claim 1, wherein the means for synchronizing a non-synchronized element of the separately obtained file with a corresponding element of the audio file of the video content, comprise:

- Means for calculating a time difference between a timecode of the non-synchronized element of the separately obtained file and a timecode of the corresponding element of the audio file of the video content; and
- Means for modifying the timecode of the non-synchronized element of the separately obtained file so that the time difference is within a predefined limit.

9. The apparatus according to claim 1, wherein the means for synchronizing a non-synchronized element of the separately obtained file with a corresponding element of the audio file of the video content, comprise:

- Means for detecting a timecode of the element of the audio file of the video content; and
- Means for attributing the detected timecode of the element of the audio file of the video content to the corresponding element of the separately obtained file.

10. Method for verification and synchronization of a file obtained separately from a video content including a video file and an audio file, the method comprising:

- determining whether the separately obtained file has a predefined correspondence to the video content; and
- if it has been determined that the separately obtained file has the predefined correspondence to the video content, determining whether an element of the separately obtained file is synchronized with a corresponding element of the audio file of the video content; and
- if it has been determined that the element of the corresponding separately obtained file is not synchronized with the corresponding element in the audio file of the video content, synchronizing the element of the separately obtained file with the corresponding element of the audio file of the video content.

11. Integrated circuit (IC) for operating a playback apparatus in accordance with claim 10.

12. A computer program product embodied in a computer-readable medium for operating a playback apparatus in accordance with claim 10.

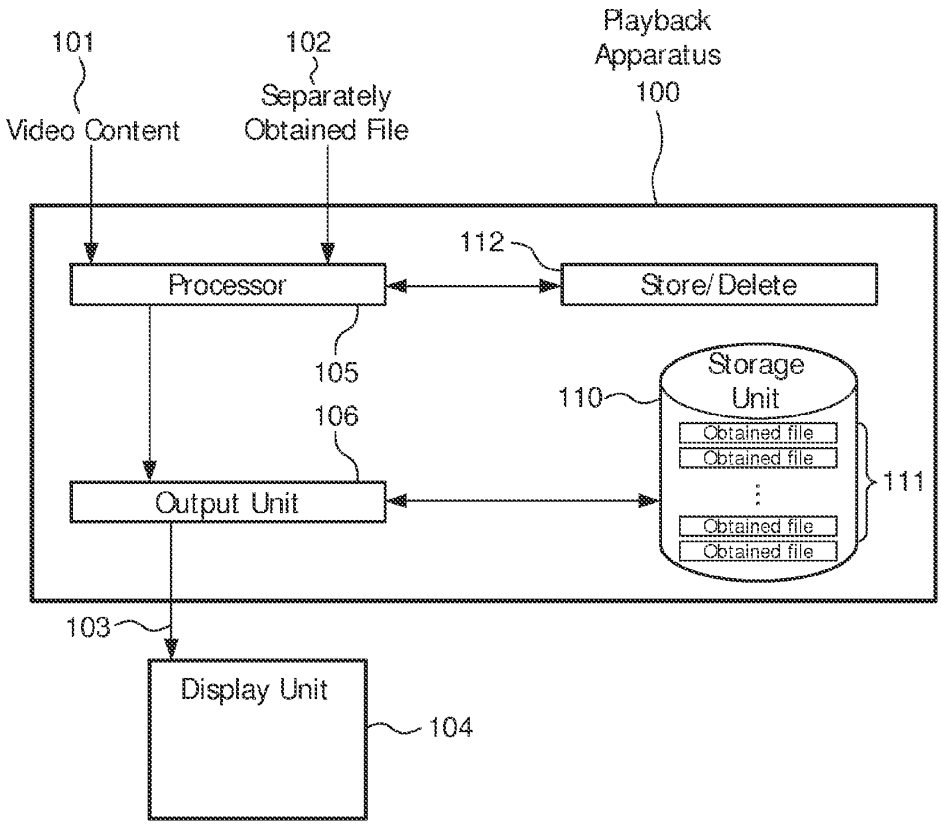

101
Video Content
102
Separately
Obtained File
Playback
Apparatus
100
112
Processor
Store/Delete
105
106
Storage
Unit
110
Obtained file
Obtained file
Obtained file
Obtained file
111
Output Unit
103
Display Unit
104

FIG. 1

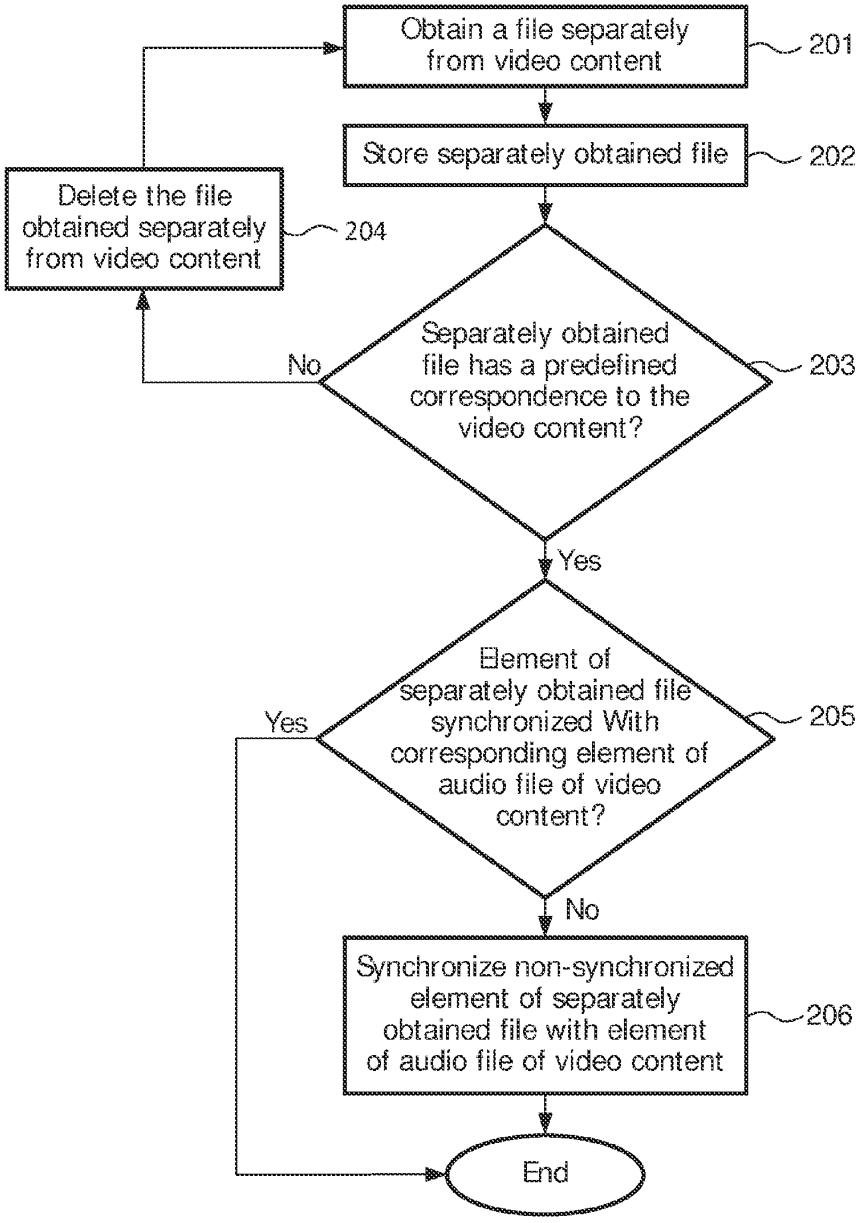

Obtain a file separately from video content
201
Store separately obtained file
202
Delete the file obtained separately from video content
204
Separately obtained file has a predefined correspondence to the video content?
203
No
Yes
Element of separately obtained file synchronized With corresponding element of audio file of video content?
205
Yes
No
Synchronize non-synchronized element of separately obtained file with element of audio file of video content
206
End

FIG. 2

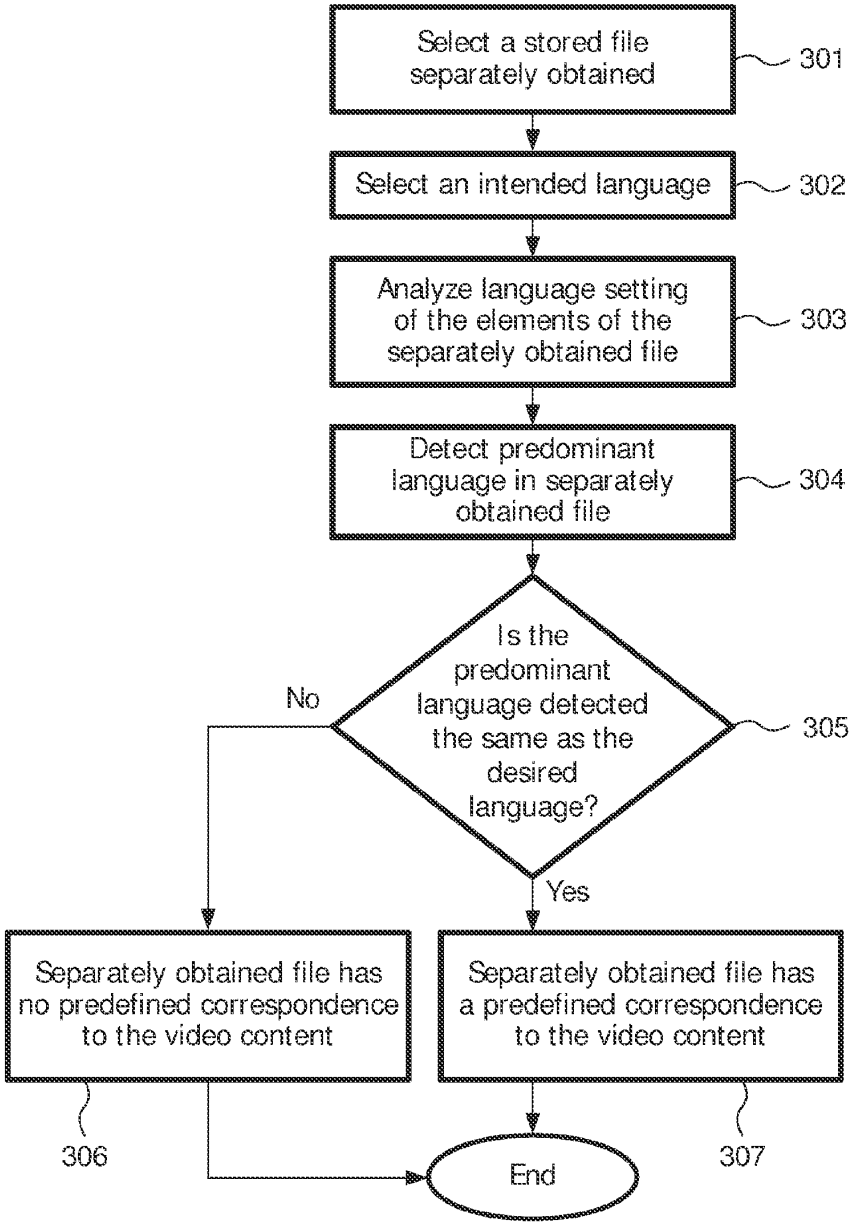

Select a stored file separately obtained
301
Select an intended language
302
Analyze language setting of the elements of the separately obtained file
303
Detect predominant language in separately obtained file
304
Is the predominant language detected the same as the desired language?
305
No
Yes
Separately obtained file has no predefined correspondence to the video content
306
Separately obtained file has a predefined correspondence to the video content
307
End

FIG. 3

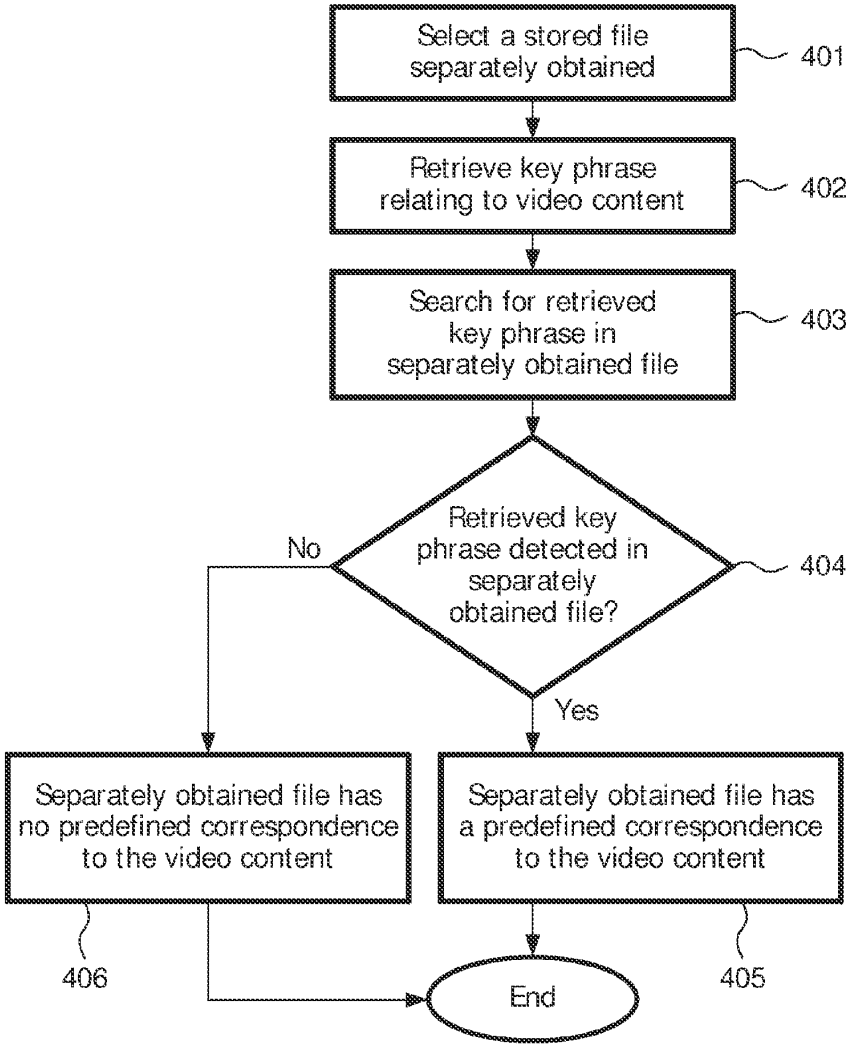

Select a stored file separately obtained
401
Retrieve key phrase relating to video content
402
Search for retrieved key phrase in separately obtained file
403
Retrieved key phrase detected in separately obtained file?
404
No
Yes
Separately obtained file has no predefined correspondence to the video content
406
Separately obtained file has a predefined correspondence to the video content
405
End

FIG. 4

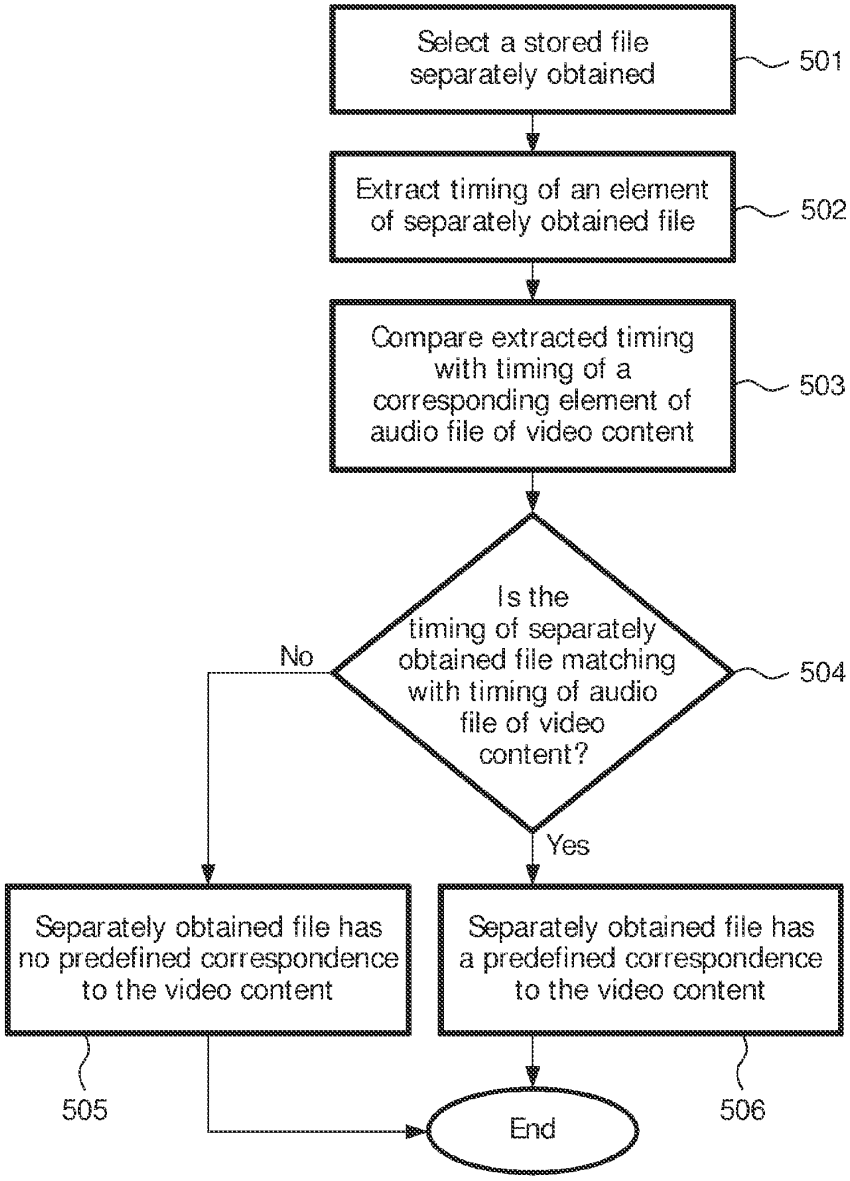

Select a stored file separately obtained
501
Extract timing of an element of separately obtained file
502
Compare extracted timing with timing of a corresponding element of audio file of video content
503
Is the timing of separately obtained file matching with timing of audio file of video content?
504
No
Yes
Separately obtained file has no predefined correspondence to the video content
505
Separately obtained file has a predefined correspondence to the video content
506
End

FIG. 5

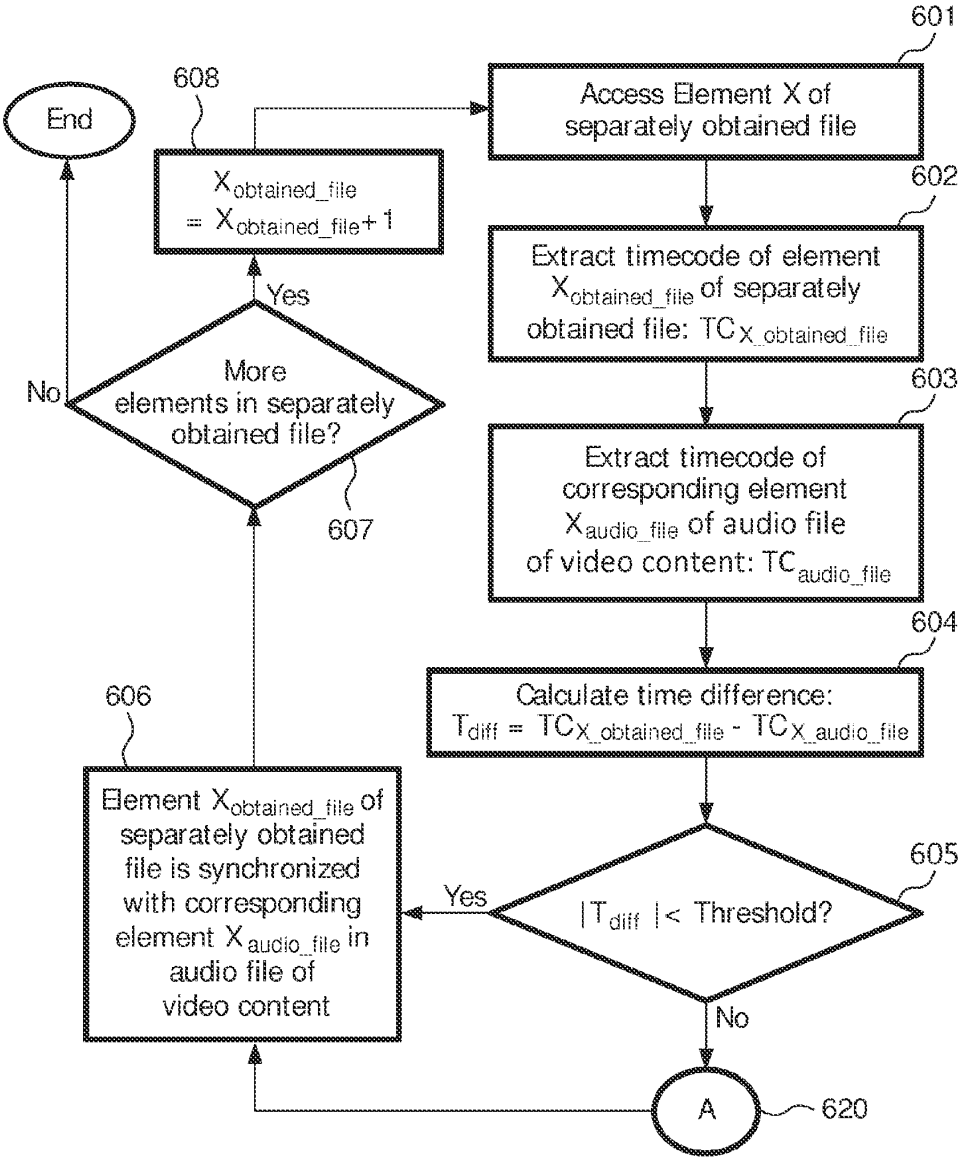

601
Access Element X of separately obtained file
602
Extract timecode of element $X_{obtained\_file}$ of separately obtained file: $TC_{X\_obtained\_file}$
603
Extract timecode of corresponding element X_{audio_file} of audio file of video content: TC_{audio_file}
604
Calculate time difference: $T_{diff} = TC_{X\_obtained\_file} - TC_{X\_audio\_file}$
605
$|T_{diff}| <$ Threshold?
Yes
No
606
Element $X_{obtained\_file}$ of separately obtained file is synchronized with corresponding element X_{audio_file} in audio file of video content
607
More elements in separately obtained file?
Yes
No
608
$X_{obtained\_file} = X_{obtained\_file} + 1$
End
A
620

FIG. 6A

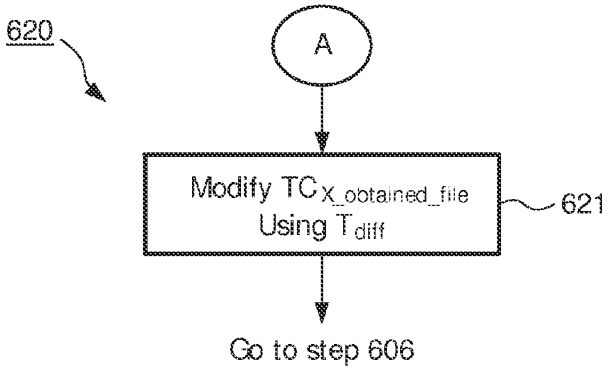

620
A
Modify $TC_{X\_obtained\_file}$ Using T_{diff}
621
Go to step 606

FIG. 6B

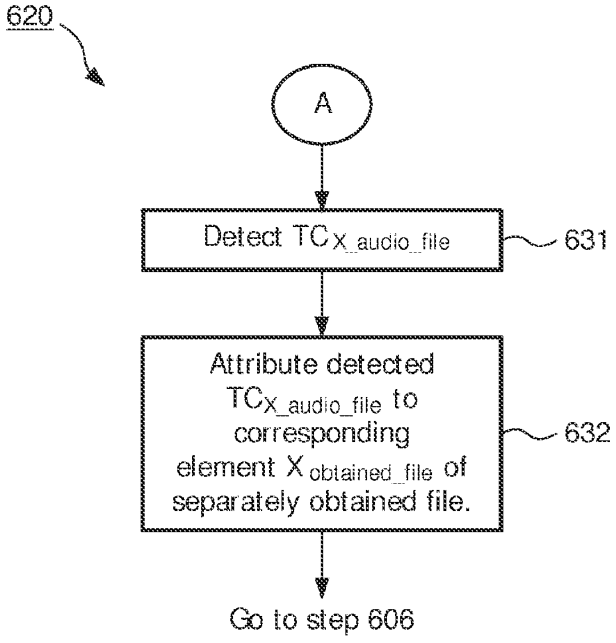

620
A
Detect $TC_{X\_audio\_file}$
631
Attribute detected $TC_{X\_audio\_file}$ to corresponding element $X_{obtained\_file}$ of separately obtained file.
632
Go to step 606

FIG. 6C

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2010/051588

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N5/445 G06K9/00 G06F17/30 H04N7/173
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N G06K G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 01/18678 A2 (LIBERATE TECHNOLOGIES L L C [US]; BOUCHER ANTOINE [CA]; MCRAE PAUL E [) 15 March 2001 (2001-03-15) page 2, line 14 - page 3, line 32 page 41, line 12 - page 43, line 7	1-12
X	US 2004/220791 A1 (LAMKIN ALLAN B [US] ET AL) 4 November 2004 (2004-11-04) the whole document	1-12
X	US 2008/005130 A1 (LOGAN JAMES D [US] ET AL) 3 January 2008 (2008-01-03) paragraphs [0006], [0040] - [0043], [0182] - [0187], [0206], [0229], [0231], [0249]	1-12

-/--

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 June 2010	05/07/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Brandenburg, Jörg

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2010/051588

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2004/055630 A2 (SCIENTIFIC ATLANTA [US]) 1 July 2004 (2004-07-01) page 3, line 26 - page 4, line 23; claims 1-35; figures 2-6	1-12
X	US 6 289 165 B1 (ABECASSIS MAX [US]) 11 September 2001 (2001-09-11)	1,10-12
A	the whole document	2-9
A	US 2003/174160 A1 (DEUTSCHER JOHN [US] ET AL) 18 September 2003 (2003-09-18) the whole document	1-12
A	US 2008/092045 A1 (CANDELORE BRANT L [US]) 17 April 2008 (2008-04-17) the whole document	1-12

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/IB2010/051588

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0118678 A2	15-03-2001	AU 765232 B2	11-09-2003
		AU 7122200 A	10-04-2001
		CA 2384674 A1	15-03-2001
		EP 1395912 A2	10-03-2004
		JP 2004508605 T	18-03-2004
US 2004220791 A1	04-11-2004	NONE	
US 2008005130 A1	03-01-2008	US 2009198357 A1	06-08-2009
		US 2009198358 A1	06-08-2009
		US 2009177303 A1	09-07-2009
		US 2009248182 A1	01-10-2009
		US 2008155616 A1	26-06-2008
WO 2004055630 A2	01-07-2004	NONE	
US 6289165 B1	11-09-2001	US 2002097984 A1	25-07-2002
		US 6408128 B1	18-06-2002
		US 6504990 B1	07-01-2003
US 2003174160 A1	18-09-2003	US 2006288389 A1	21-12-2006
US 2008092045 A1	17-04-2008	NONE	

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 June 2011 (09.06.2011)

PCT

(10) International Publication Number
WO 2011/068668 A1

(51) International Patent Classification:
H04N 7/167 (2011.01)

(21) International Application Number:
PCT/US2010/056733

(22) International Filing Date:
15 November 2010 (15.11.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/266,982 4 December 2009 (04.12.2009) US

(71) Applicant *(for all designated States except US)*: **DIVX, LLC** [US/US]; 4780 Eastgate Mall, San Diego, CA 92121 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CHAN, Francis, Yee-dug** [US/US]; 4639 Executive Drive, Unit 63, San Diego, CA 92121 (US). **SOROUSHIAN, Kourosh** [US/US]; 3758 Brookshire Street, San Diego, CA 92111 (US). **WOOD, Andrew, Jeffrey** [US/US]; 2753 Mobley Street, San Diego, CA 92123 (US).

(74) Agent: **BAILEY, David, J.**; Kauth, Pomeroy, Peck & Bailey LLP, 2875 Michelle Drive, Suite 110, Irvine, CA 92606 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: ELEMENTARY BITSTREAM CRYPTOGRAPHIC MATERIAL TRANSPORT SYSTEMS AND METHODS

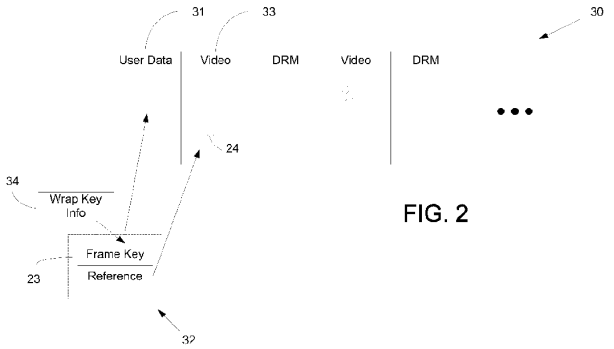


FIG. 2

(57) Abstract: Systems and methods for providing multimedia content from one process or component to another process or component over an unsecured connection are provided. One embodiment includes obtaining the cryptographic information, extracting the at least partially encrypted video data from the container file to create an elementary bitstream, enciphering the cryptographic information, inserting the cryptographic information in the elementary bitstream, providing the elementary bitstream to a video decoder, extracting the cryptographic information from the elementary bitstream at the video decoder, deciphering the cryptographic information, decrypting the elementary bitstream with the cryptographic information and decoding the elementary bitstream for rendering on a display device using the video decoder.

WO 2011/068668 A1

ELEMENTARY BITSTREAM CRYPTOGRAPHIC MATERIAL TRANSPORT SYSTEMS AND METHODS

BACKGROUND

[0001] The present invention generally relates to digital multimedia distribution systems and more specifically to digital transmission of encrypted multimedia content over an unsecured connection.

[0002] Providers of multimedia content can digitize content for distribution via digital communication networks. An important issue faced by a content distribution system is enabling only those customers that have purchased the content to play the content and compartmentalize access to all the stakeholders in the content distribution chain. One approach is to encrypt portions of the content and to issue encryption keys to authorized users that enable encrypted portions of the content to be unencrypted. Layers of keys and protection policies can be used so a single encryption key alone is insufficient for the user to access the content. In a number of systems, users purchase players that possess specified decryption capabilities. Content providers can distribute content to user's owning such a player in an encryption format supported by the player. Complying with a specified protection policy typically involves using an encryption key specified by the manufacturer of the players. In many instances the manufacturer of the players will not reveal the encryption keys used in the specified encryption scheme and likewise the content provider does not want to share the content keys to the manufacturer of the players.

[0003] Communications between components or processes of players or playback systems are typically trustworthy and secured. However, when communication or the transporting of information becomes unsecured or untrustworthy, such gaps need to be accounted for and filled. This has become more evident with advent and popularity of open multimedia frameworks. Bi-directional communication requirements and/or run time challenges and authentication requests to fill such gaps have proved to be less than adequate.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0584
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0004] There are many ways of securing communication, including ciphering and encryption.

[0005] Ciphering is a procedure used to secure data that typically involves using a series of steps to scramble and render the data readable only to the intended audience. The procedure itself does not require an outside source, such as a key, in order to encipher or decipher the data. Rather, data can be properly deciphered by the intended audience so long as deciphering exactly follows the enciphering steps to unravel the data. Encryption is a procedure used to secure data. That typically involves the use of an external input for at least one step in the procedure, such as a key, in order to secure and/or access the data. The external data is used to intentionally manipulate at least one step in the encryption or decryption process, changing the way the data processing for encryption occurs. Generally, without the external data or a corresponding decryption key in an encryption process, a step in a corresponding decryption process cannot properly be executed and the data cannot be properly decrypted.

[0006] In the context of digital media, encoding is a procedure by which digital media is represented in a digital format. The format is typically selected to obtain specific benefits during the transportation, playback and storage of the digital media format used. For example, representing the media using fewer bits may be beneficial to transfer data in order to minimize bandwidth usage or storage space. In another example, a media player may only decode or read media in a certain format and therefore the digital media may first be in that format in order to be decoded by that media player.

[0007] Decoding is a procedure by which digital media in a format is translated into a format readable by a media player for rendering on a display device. Often, decoding may also reverse processes associated with encoding such as compression. In instances where encryption and/or enciphering have been applied to encoded media, the enciphering process or encryption process typically must be reversed before the encoded media can be decoded.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0585
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

SUMMARY OF THE INVENTION

[0008] Systems and methods are described for taking cryptographic material from a container file and inserting the cryptographic material in an elementary bitstream, where the cryptographic information can then be used to decrypt the elementary bitstream for playback

[0009] A number of embodiments include obtaining the cryptographic information, extracting the at least partially encrypted video data from the container file to create an elementary bitstream, enciphering the cryptographic information, inserting the cryptographic information in the elementary bitstream, providing the elementary bitstream to a video decoder, extracting the cryptographic information from the elementary bitstream at the video decoder, deciphering the cryptographic information, decrypting the elementary bitstream with the cryptographic information and decoding the elementary bitstream for rendering on a display device using the video decoder.

[0010] In a further embodiment, the cryptographic information is obtained from the container file.

[0011] In another embodiment, the cryptographic information includes key information and information concerning at least a portion of the at least partially encrypted video data that is encrypted using the key information.

[0012] In an additional embodiment, information concerning at least a portion of the at least partially encrypted video data is a reference to a block of encrypted data within an encoded frame of video that is encrypted using the key information.

[0013] In a still further embodiment, the cryptographic information inserted in the elementary bitstream is delimited by an identifier and the cryptographic information is inserted before the at least partially encrypted video data encrypted using the key information.

[0014] In a still other embodiment, the cryptographic information is extracted using the identifier.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0586
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0015] In a still additional embodiment, the decrypting process is performed by using the key information to identify the encrypted portion of video data and decrypting the encrypted video data using the key information.

[0016] In a yet further embodiment, cryptographic information inserted in different locations within the elementary bitstream includes different key information.

[0017] In a yet other embodiment, the at least partially encrypted video data includes frames of encoded video. In addition, the at least partially encrypted video data includes at least a portion of a plurality of the encoded frames that is encrypted.

[0018] In a yet further additional embodiment, the enciphering process and the deciphering process are synchronized such that a delay in excess of a predetermined time between enciphering and deciphering results in the cryptographic information being unrecoverable.

[0019] In a still further embodiment again, the enciphering process enciphers data by using a sequence of scrambling processes to scramble data.

[0020] In a still other embodiment again, the deciphering process deciphers data by performing the inverse sequence of scrambling processes to the sequence used to scramble the data.

[0021] Many embodiments include a demultiplexer configured to extract the at least partially encrypted video data from the container file to create an elementary bitstream, a video decoder configured to decrypt the elementary bitstream using the cryptographic information and decode the elementary bitstream for rendering on a display device. Additionally, the demultiplexer is configured to encipher the cryptographic information and insert the enciphered cryptographic information in the elementary bitstream and the decoder is configured to extract enciphered cryptographic information from an elementary bitstream and to decipher the cryptographic information.

[0022] In a further embodiment, the cryptographic information is obtained from the container file.

[0023] In another embodiment, the cryptographic information includes

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0587
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

key information and information concerning at least a portion of the at least partially encrypted video data that is encrypted using the key information.

[0024] In an additional embodiment, the information concerning at least a portion of the at least partially encrypted video data is a reference to a block of encrypted data within an encoded frame of video that is encrypted using the key information.

[0025] In a further embodiment again, the demultiplexer is configured to insert the cryptographic information delimited by an identifier in the elementary bitstream and insert the cryptographic information before the at least partially encrypted video data encrypted using the key information.

[0026] In another embodiment again, the decoder is configured to extract the cryptographic information using the identifier.

[0027] In an additional embodiment again, the decoder is configured to decrypt the portion of the video data encrypted using the key information by identifying the encrypted portion of video data and decrypting the encrypted video data using the key information.

[0028] In a still further embodiment again, cryptographic information inserted in different locations within the elementary bitstream includes different key information.

[0029] In still another embodiment again, the at least partially encrypted video data includes frames of encoded video. Additionally, at least a portion of a plurality of the encoded frames is encrypted.

[0030] In a still additional embodiment, both the demultiplexer and the decoder are configured to be synchronized such that a delay in excess of a predetermined time between enciphering and deciphering results in the cryptographic information being unrecoverable.

[0031] In a yet further embodiment, the demultiplexer is configured to encipher data by using a sequence of scrambling processes to scramble data.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0588
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0032] In a yet other embodiment, the decoder is configured to decipher data by performing the inverse sequence of scrambling processes to the sequence used to scramble the data.

[0033] Numerous embodiments include obtaining the cryptographic information. In addition, the cryptographic information is obtained from the container file. Also, the at least partially encrypted video data includes frames of encoded video and at least a portion of a plurality of the encoded frames is encrypted. Additionally, the cryptographic information includes key information and information concerning at least a portion of the least partially encrypted video data that is encrypted using the key information. Furthermore, the information concerning at least a portion of the at least partially encrypted video data is a reference to a block of encrypted data within an encoded frame of video that is encrypted using the key information and the cryptographic information inserted in different locations within the elementary bitstream includes different key information.

[0034] Several embodiments include extracting the at least partially encrypted video data from the container file to create an elementary bitstream. In addition, the cryptographic information inserted in the elementary bitstream is delimited by an identifier and the cryptographic information is inserted before the at least partially encrypted video data encrypted using the key information.

[0035] Many embodiments include enciphering the cryptographic information and inserting the cryptographic information in the elementary bitstream. In addition, the cryptographic information is extracted using the identifier.

[0036] A number of embodiments include providing the elementary bitstream to a video decoder, extracting the cryptographic information from the elementary bitstream at the video decoder and deciphering the cryptographic information. In addition, the enciphering process and the deciphering process are synchronized such that a delay in excess of a predetermined time between enciphering and deciphering results in the cryptographic information being unrecoverable. Also, the enciphering process

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0589
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

enciphers data by using a sequence of scrambling processes to scramble data. Furthermore, the deciphering process deciphers data by performing the inverse sequence of scrambling processes in the sequence used to unscramble data.

[0037] Several embodiments include decrypting the elementary bitstream with the cryptographic information. In addition, the decrypting process is performed by using the key information to identify the encrypted portion of video data and decrypting the encrypted video data using the key information.

[0038] Many embodiments include decoding the elementary bitstream for rendering on a display device using the video decoder.

BRIEF DESCRIPTION OF THE DRAWINGS

[0039] FIG. 1 illustrates a graphical representation of a multimedia container file in accordance with various embodiments of the present invention.

[0040] FIG. 2 illustrates a graphical representation of a bitstream with cryptographic material in accordance with various embodiments of the present invention.

[0041] FIG. 3 is a block diagram of a multimedia cryptographic bitstream transport system in accordance with various embodiments of the present invention.

[0042] FIG. 4 is a flow diagram of a demultiplex and authentication process in accordance with various embodiments of the present invention.

[0043] FIG. 5 is a flow diagram of a decoder and decipher process in accordance with various embodiments of the present invention.

[0044] FIG. 6 is a block diagram of a multimedia cryptographic bitstream transport system in accordance with various embodiments of the present invention.

[0045] FIG. 7 is a flow diagram of a wrap key generation process in accordance with various embodiments of the present invention.

[0046] FIG. 8 is a flow diagram of a bitstream insertion process in accordance with various embodiments of the present invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0590
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DETAILED DESCRIPTION

[0047] Systems and methods for providing multimedia content from one process or component to another process or component over an unsecured connection are provided. In several embodiments, the transmission occurs between a demultiplexer and a decoder over an unsecured connection where traditionally such connections are secured. In many embodiments, the transmission occurs on a bi-directional communication path. Embodiments of the present invention do not secure the transmission but rather secure the data being transmitted via the unsecured connection. The transmitted data in a number of embodiments includes an encrypted multimedia bitstream and associated cryptographic material in the bitstream for transmission to a decoder for decryption. In various embodiments, a bi-directional communication path between a demultiplexer and the decoder is not used. Additionally, by allowing the decryption to occur on the decoder the bitstream is protected even if the connection is compromised and an unauthorized component or process intercepts the bitstream.

[0048] In various embodiments, frame keys are used to decrypt the bitstream. For example, in the manner described in U.S. Patent 7,295,673 to Grab et al. the disclosure of which is incorporated by reference herein in its entirety. In several embodiments, the frame keys are protected by a cryptographic wrap algorithm that uses a separate series of newly generated keys. The wrapped frame keys are inserted into the encrypted bit stream for deciphering and decoding by the decoder. The cryptographic information in various embodiments includes information to decrypt a video frame or a portion of the video frame. In various embodiments, a time indicator in the form of a frame sequence is also utilized to ensure connection between the demultiplexer and decoder is not being intercepted or spied upon.

[0049] The cryptographic information inserted into the elementary bitstream can take any of a variety of forms. In many embodiments, the cryptographic information includes a frame key and/or a reference to a block of encrypted video data. In several

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0591
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

embodiments, the cryptographic information contains an index to a frame key or a separate reference to both a frame key and an encrypted block. A number of embodiments provide for first inserting a table of possible keys and still further embodiments provide for sending multiple keys where different keys are used to encrypt different portions of the video.

[0050] Turning now to the drawings, FIG. 1 represents a multimedia container file 20 including encrypted content, e.g., video. The multimedia container file includes a digital rights management portion 21 preceding associated video portions or chunks 22. The digital rights management portion includes at least one frame key 23 or an index to a frame key in a separately provided table of frame keys, which in many embodiments is encrypted in a way that only enables playback by a particular device and/or user. The digital rights management portion also points to or identifies a specified portion of or an entire video frame within the video chunk 24 that is encrypted. Without first decrypting this encrypted portion of the video chunk, the video content cannot be decoded or displayed. The multimedia container file is supplied to a demultiplexer.

[0051] The demultiplexer parses the multimedia container file and transmits portions or chunks of data, e.g., video or audio, to a decoder. However, prior to transmitting the video data, the demultiplexer incorporates or attaches cryptographic material to the video data.

[0052] FIG. 2 graphically illustrates the generated multimedia bitstream sent to the decoder. The bitstream 30 includes a header or user data 31 that includes cryptographic material 32. In accordance with many embodiments of the invention, the material includes the frame key 23 from the multimedia container file, which is encrypted using a wrap key, and wrap key information 34 to provide synchronization of the demultiplexer to the decoder in order to decipher the cryptographic material. As is discussed below, the wrap key information can take any of a variety of different forms depending upon the specific application including but not limited to information

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0592
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

enabling synchronization of wrap key factories and/or the direct transfer of the wrap keys themselves. The associated video data 33 follows.

[0053] Referring now to FIG. 3, a demultiplexer 10 that receives a multimedia container file that includes video and audio data, portions of which are encrypted, is shown. In one embodiment, the multimedia file conforms to a specific format such as audio video interleave (AVI) or Matroska (MKV). The multimedia file is provided via a disc, flash memory device or another tangible storage medium or streamed or otherwise transmitted to the demultiplexer. The demultiplexer separates portions of the received multimedia data including but not limited to video, audio and encryption data that is supplied to an upstream digital rights management component 15. In various embodiments, the connection between the demultiplexer 10 and the digital rights management component 15 can be secure although need not be depending upon the requirements of the application. The digital rights management component 15 generates cryptographic material and the multimedia bitstream transport that is supplied to a decoder 20. In particular, the demultiplexer 10 transmits video data with cryptographic material to the decoder 20.

[0054] The connection between the demultiplexer and the decoder is typically secured. However, in the illustrated embodiment, the connection is not secured. Typically, the multimedia file is authorized and decrypted in a demultiplexer and then transmitted downstream unencrypted to the decoder via an inter-communication data channel. This however can present a security problem due to the high value of the unencrypted but still encoded bitstream that can be captured during transmission. This bitstream is considered high-value since the encoded data can be easily multiplexed back into a container for unprotected and unauthorized views and/or distribution with no loss in the quality of the data. In the illustrated embodiment, the video provided to the decoder 20 by the demultiplexer 10 is at least partially encrypted and the decoder 20 communicates with a downstream digital rights management component 25 that deciphers the cryptographic material. Utilizing the deciphered cryptographic material,

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0593
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the digital rights management component is able to access the encryption data and thereby decrypt and decode the video data for playback.

[0055] The general processes of the demultiplexer and the decoder are now described. In FIG. 4, the demultiplexer and authentication process is illustrated in which a multimedia container file is received and portions of which are identified or separated (101). If encryption data is identified, cryptographic packets or material are generated (102) and stored in a temporary buffer (103). However, if video data is identified (104), the cryptographic material stored in the temporary buffer is combined with the video data (105) and then transmitted to a video decoder (106). If audio data is identified (107), the audio data is transmitted (108) to the audio decoder. It should be appreciated that audio or other types of data may also include encryption data and thus associated cryptographic material is generated and combined with the associated data and transmitted to the respective decoder. Also, other types of data may be included in the container file without encryption data and thus is transmitted directly to the associated decoder.

[0056] In FIG. 5, a decoder and decipher process is illustrated in which the decoder receives video and/or audio data sent from the demultiplexer (201). The decoder deciphers the cryptographic material supplied with the associated data (202). Utilizing the deciphered material, the encrypted data is decrypted (203) and decoded (204) by the decoder for playback.

[0057] To further elaborate on the demultiplexer and decoder processes and the bitstream transport system, a more detailed representation of the demultiplexer's and decoder's associated digital rights manager along with the associated processes are illustrated in the remaining figures.

[0058] Referring to FIG. 6, the upstream digital rights manager 15 of the demultiplexer 10 includes an authentication engine 16, a bit stream inserter 17, a payload builder 18 and a wrap key factory 19. The downstream digital rights manager 25 of the decoder includes a decrypt engine 26, a bit stream decoder 27, a payload

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0594
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

parser 28 and a wrap key factory 29. The authentication engine prepares cryptographic material utilizing the encryption data from the container file and the video data in conjunction with the payload builder 18 and the wrap key factory 19.

[0059] The payload builder 18 provides discrete units of cryptographic material in the bitstream delimited by an identifier. On the decoder, the payload parser 28 utilizes the identifiers to extract the discrete units, which are then processed by the decrypt engine 26. In many embodiments, the cryptographic material in one embodiment includes a bitstream frame header along with a cryptographic payload. The cryptographic payload, however, is not dependent on the format of the header of the elementary bitstream, e.g., MPEG-4 or H.264.

[0060] In one embodiment, the payload builder 18 inserts a reserved start code identifier along with a cryptographic payload at the front of each video chunk that is demultiplexed. By utilizing a reserved start code, the decrypt engine 26 can pass the entire video data including the inserted cryptographic material to the decoder 20 that simply discards or ignores the cryptographic material. For example, a MPEG-4 compliant decoder discards frames that contain a reserved start code identifier that is included in the bitstream. Accordingly, removal of any of the cryptographic material from the bitstream is not needed to decode the associated data.

[0061] The cryptographic payload in one embodiment includes three different packet types: a wrap key, a synchronization payload and a frame payload. The frame payload indicates that the current frame is encrypted and includes key information and a reference to at least a portion of the encoded frame that is encrypted. The frame payload can be used to decrypt the video frame. The synchronization payload is the first packet sent to synchronize the authentication engine of the demultiplexer to the decrypt engine of the decoder. This synchronization ensures that data transmitted from the demultiplexer to the decoder is not being intercepted. The wrap key includes information to unwrap or decipher the transmitted data from the demultiplexer.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0595
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0062] The bit stream inserter 17 packages the cryptographic material for transport with the video data. Conversely, the bit stream decoder 27 of the decoder unpacks the cryptographic material from the bitstream. In one embodiment, frame keys are transported in the bitstream and are sent when a key index change is detected by the authentication engine of the demultiplexer. In many embodiments, the decrypt engine of the decoder stores only one frame key and thus frame encryption information sent by the demultiplexer applies to the current frame. If the decrypt engine receives a new frame key from the demultiplexer, the decrypt engine stores the new frame key and uses it to decrypt the next frame. In a number of embodiments, a key table is transmitted and stored in the decrypt engine for reference by subsequent encryption information. In several embodiments, the decoder does not enforce key rotation. In many embodiments, however, the decoder expects a new frame key after a predetermined number of frames in the sequence of frames. In this way, the decrypt engine can identify when supplied frame information is unreliable and terminate the decoding of the multimedia bitstream.

[0063] The wrap key factory 19 encrypts or wraps the cryptographic material for transport on the bitstream to the decoder. In one embodiment, the wrap key factory uses a key wrap process based on the Advanced Encryption Standard (AES) and uses the ECB Cipher Mode to provide cryptographic security for wrapping small blocks of data using chaining and cipher feedback loop. The key wrap process is stateless. A corresponding wrap key factory is included with the decoder to unwrap the cryptographic material. Synchronization with the corresponding wrap key factory 29 is used to allow unwrapping of the material without communication back to the demultiplexer (i.e., bi-directional communication) and to prevent unauthorized decoding of the content by, for example, a rogue process intercepting or copying the transmitted content.

Wrap Key Factory

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0596
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0064] In one embodiment, each of the authentication and decryption blocks (digital rights managers 15, 25) construct a series of predictable transform number sequences using a common heuristic. Subsequently, those numbers are combined with a random value for additional entropy used to contribute toward key material for wrapping keys.

[0065] A flow diagram of a wrap key generation process 300 in accordance with an embodiment of the invention is illustrated in FIG. 7. A selected heuristic (302) is combined with key material (304) to create a wrap key (306).

[0066] In accordance with various embodiments, one such heuristic (302) may combine the use of a predictable number sequence generator such that identical transform sequences can be generated by different heuristics even though no information is exchanged. If both authentication and decrypt blocks are created such that the output of the common heuristic are identical, the key material (304) generated from such heuristic will be identical. This may apply in situations where a wrapped key (306) and a selected heuristic (302) are provided. Any process for generating identical encryption keys without exchange of key material can be used as an appropriate heuristic to generate wrapping keys (306) in accordance with embodiments of the invention. Although, some information exchange to enable synchronization between the two wrap key factories can be utilized in accordance with embodiments of the invention.

[0067] The two wrap key factories use the same transform sequence. To synchronize the wrap key factories, the sender's wrap key factory selects one heuristic (302) from a predetermined set of heuristics to generate the key material for the next wrap key. The decoder factory will receive a known payload that has been encrypted with the sender's wrap key (306) generated using selected heuristic (302) from the known set of heuristics. The receiver then attempts to decrypt and verify the contents of the payload using each of the predetermined heuristics. If the material matches what is expected, then the receiver has identified the correct heuristic (302). If all the heuristics are exhausted, then this is considered a fatal error and decryption cannot continue.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0597
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0068] Initially, the synchronization payload is used to assist the decrypt block in identifying the appropriate transform sequence quickly. Once the decrypt block locates the proper heuristic (302), the decrypt block wrap key factory utilizes that transform sequence for all subsequent transforms. In several embodiments, once a heuristic has exhausted all values, that heuristic will deterministically choose the next heuristic to use.

[0069] Run time synchronization is maintained through monotonically incrementing a wrap number that is incremented for each wrap key generated. If an error occurs using a particular wrap key (i.e. unallowable data present in the cryptographic payload), the wrap key factory will regenerate a new wrap key and subsequently increment the wrap number. In one embodiment, the frame payload received by the decrypt block contains a wrap number element. On the decrypt block, this wrap number element is compared with the internal wrap number of the decrypt block to determine if the current wrap key needs to be skipped. In one embodiment, the wrap key includes data fed into a cryptographic digest. The resulting bytes from the digest are then used to create an AES key. A new wrap key will be generated for each payload that is wrapped.

Bitstream Data Insertion

[0070] A flow diagram of a bitstream insertion process 400 utilized with respect to video data extracted from an AVI container in accordance with an embodiment of the invention is illustrated in FIG. 8. In the demultiplexer, a caller begins extraction (402) of a relevant AVI chunk and requests (404) the DRM for the maximum expected bitstream payload. The demultiplexer then uses the information from the DRM to allocate (406) space in a buffer and passes (408) the buffer to the DRM. Next on the DRM, the video DD info is cached (410). The video DD info may be a data segment in a file container describing the data contained in a single block of container data, such as all of the video frame data in a single AVI chunk. Encrypted frames may have a DD info which contains information relating to the security features of the frame. The MPEG4 reserved start

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0598
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

code is inserted (412) into the buffer and then the cryptographic payload header is inserted (414) into the buffer. A decision (416) is then made as to whether the chunk is the first frame. If the chunk is the first frame, then a Sync() payload is inserted (418) and a FrameInfo() payload is inserted (420). The Sync() payload may include the wrap key synchronization payload to synchronize the wrap keys. The FrameInfo() payload may include the cryptographic offset and length of information relating to data security in the video data, possibliy as part of the DD Info data. If the chunk is not the first frame, then only the FrameInfo() payload is inserted (420). Then, a decision (422) is made as to whether the key index is greater than the current key index. If the key index is greater than the current key index, a FrameKey() payload is inserted (424) in the buffer and then the number of bytes inserted into the buffer is returned (426) to the caller by the DRM. The FrameKey() payload may include the payload containing the next frame key. If the key index is not lower than the current key index, then the DRM returns (426) the number of bytes inserted in the buffer to the caller. Next, the demultiplexer, is ready to extract (428) the AVI chunk. Through this process, DD info awareness occurs before the demultiplexer extracts the video chunk into the buffer for transmission to the decoder.

[0071] In various embodiments, bitstream data insertion occurs in the authentication block of the demultiplexer. The digital rights manager in one embodiment first receives the container's encryption data and temporarily stores or caches the information. The cached encryption data contains the information for the next video chunk. From this information, the digital rights manager can determine the proper bitstream payload to insert, if any. To reduce memory copies, the digital rights manager inserts the bitstream payload before extracting the chunk from the container.

[0072] Based on the cached encryption data chunk, the digital rights manager can detect frame key changes. If the frame key index has not changed since the last cached

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0599
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

encryption data, no key material is sent. In one embodiment, the encryption data is always transported if there is cached encryption data in the digital rights manager. On the first payload, there will be a synchronization payload to allow the decrypt block to synchronize the wrap sequence. The frame information payloads in one embodiment follow the synchronization payload. It should be appreciated that not all payloads are required to appear in each decrypt block. Furthermore, the processes similar to those described above with reference to FIG. 8 can also be used with respect to other container formats including but not limited to MKV container files.

[0073] Although the present invention has been described in certain specific aspects, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that the present invention may be practiced otherwise than specifically described, including various changes in the size, shape and materials, without departing from the scope and spirit of the present invention. Thus, embodiments of the present invention should be considered in all respects as illustrative and not restrictive.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0600
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS CLAIMED IS:

1. A method of playing back encrypted video on a playback device, where at least partially encrypted video data is contained within a container file and can be used to create an elementary bitstream using cryptographic information, which can be used to decrypt the at least partially encrypted video data, the method comprising:

obtaining the cryptographic information;

extracting the at least partially encrypted video data from the container file to create an elementary bitstream;

enciphering the cryptographic information;

inserting the cryptographic information in the elementary bitstream;

providing the elementary bitstream to a video decoder;

extracting the cryptographic information from the elementary bitstream at the video decoder;

deciphering the cryptographic information;

decrypting the elementary bitstream with the cryptographic information; and

decoding the elementary bitstream for rendering on a display device using the video decoder.

2. The method of claim 1, wherein the cryptographic information is obtained from the container file.

3. The method of claim 1, wherein the cryptographic information comprises:

key information; and

information concerning at least a portion of the at least partially encrypted video data that is encrypted using the key information.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0601
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

4. The method of claim 3, wherein the information concerning at least a portion of the at least partially encrypted video data is a reference to a block of encrypted data within an encoded frame of video that is encrypted using the key information.

5. The method of claim 4, wherein

the cryptographic information inserted in the elementary bitstream is delimited by an identifier; and

the cryptographic information is inserted before the at least partially encrypted video data encrypted using the key information.

6. The method of claim 5, wherein the cryptographic information is extracted using the identifier.

7. The method of claim 6, wherein the decrypting process is performed by using the key information to identify the encrypted portion of video data and decrypting the encrypted video data using the key information.

8. The method of claim 7, wherein cryptographic information inserted in different locations within the elementary bitstream includes different key information.

9. The method of claim 1, wherein the at least partially encrypted video data comprises:

frames of encoded video;

wherein at least a portion of a plurality of the encoded frames is encrypted.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0602
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

10. The method of claim 1, wherein the enciphering process and the deciphering process are synchronized such that a delay in excess of a predetermined time between enciphering and deciphering results in the cryptographic information being unrecoverable.

11. The method of claim 10, wherein the enciphering process enciphers data by using a sequence of scrambling processes to scramble data.

12. The method of claim 11, wherein the deciphering process deciphers data by performing the inverse sequence of scrambling processes to the sequence used to scramble the data.

13. A system for playback of encrypted video, where at least partially encrypted video data is contained within a container file and can be used to create an elementary bitstream using cryptographic information, which can be used to decrypt the at least partially encrypted video data, comprising:

a demultiplexer configured to extract the at least partially encrypted video data from the container file to create an elementary bitstream; and

a video decoder configured to decrypt the elementary bitstream using the cryptographic information and decode the elementary bitstream for rendering on a display device;

wherein the demultiplexer is configured to encipher the cryptographic information and insert the enciphered cryptographic information in the elementary bitstream; and

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0603
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

wherein the decoder is configured to extract enciphered cryptographic information from an elementary bitstream and to decipher the cryptographic information.

14. The system of claim 13, wherein the cryptographic information is obtained from the container file.

15. The system of claim 13, wherein the cryptographic information comprises

key information; and

information concerning at least a portion of the at least partially encrypted video data that is encrypted using the key information.

16. The system of claim 15, wherein the information concerning at least a portion of the at least partially encrypted video data is a reference to a block of encrypted data within an encoded frame of video that is encrypted using the key information.

17. The system of claim 16, wherein the demultiplexer is configured to:

insert the cryptographic information delimited by an identifier in the elementary bitstream; and

insert the cryptographic information before the at least partially encrypted video data encrypted using the key information.

18. The system of claim 17, wherein the decoder is configured to extract the cryptographic information using the identifier.

19. The system of claim 18, wherein the decoder is configured to decrypt the portion of the video data encrypted using the key information by identifying

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0604
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the encrypted portion of video data and decrypting the encrypted video data using the key information.

20. The system of claim 19, wherein cryptographic information inserted in different locations within the elementary bitstream includes different key information.

21. The system of claim 13, wherein the at least partially encrypted video data comprises:

frames of encoded video;

wherein at least a portion of a plurality of the encoded frames is encrypted.

22. The system of claim 13, wherein both the demultiplexer and the decoder are configured to be synchronized such that a delay in excess of a predetermined time between enciphering and deciphering results in the cryptographic information being unrecoverable.

23. The system of claim 22, wherein the demultiplexer is configured to encipher data by using a sequence of scrambling processes to scramble data.

24. The system of claim 23, wherein the decoder is configured to decipher data by performing the inverse sequence of scrambling processes to the sequence used to scramble the data.

25. A method of playing back encrypted video on a playback device, where at least partially encrypted video data is contained within a container file and

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0605
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

can be used to create an elementary bitstream using cryptographic information, which can be used to decrypt the at least partially encrypted video data, the method comprising:

obtaining the cryptographic information,

wherein the cryptographic information is obtained from the container file;

wherein the at least partially encrypted video data comprises:

frames of encoded video; and

wherein at least a portion of a plurality of the encoded frames is encrypted;

wherein the cryptographic information comprises:

key information; and

information concerning at least a portion of the least partially encrypted video data that is encrypted using the key information;

wherein the information concerning at least a portion of the at least partially encrypted video data is a reference to a block of encrypted data within an encoded frame of video that is encrypted using the key information; and

wherein cryptographic information inserted in different locations within the elementary bitstream includes different key information;

extracting the at least partially encrypted video data from the container file to create an elementary bitstream,

wherein the cryptographic information inserted in the elementary bitstream is delimited by an identifier; and

wherein the cryptographic information is inserted before the at least partially encrypted video data encrypted using the key information;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0606
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

enciphering the cryptographic information;

inserting the cryptographic information in the elementary bitstream,

wherein the cryptographic information is extracted using the identifier;

providing the elementary bitstream to a video decoder;

extracting the cryptographic information from the elementary bitstream at the video decoder;

deciphering the cryptographic information,

wherein the enciphering process and the deciphering process are synchronized such that a delay in excess of a predetermined time between enciphering and deciphering results in the cryptographic information being unrecoverable;

wherein the enciphering process enciphers data by using a sequence of scrambling processes to scramble data; and

wherein the deciphering process deciphers data by performing the inverse sequence of scrambling processes in the sequence used to unscramble data;

decrypting the elementary bitstream with the cryptographic information,

wherein the decrypting process is performed by using the key information to identify the encrypted portion of video data and decrypting the encrypted video data using the key information; and

decoding the elementary bitstream for rendering on a display device using the video decoder.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0607
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

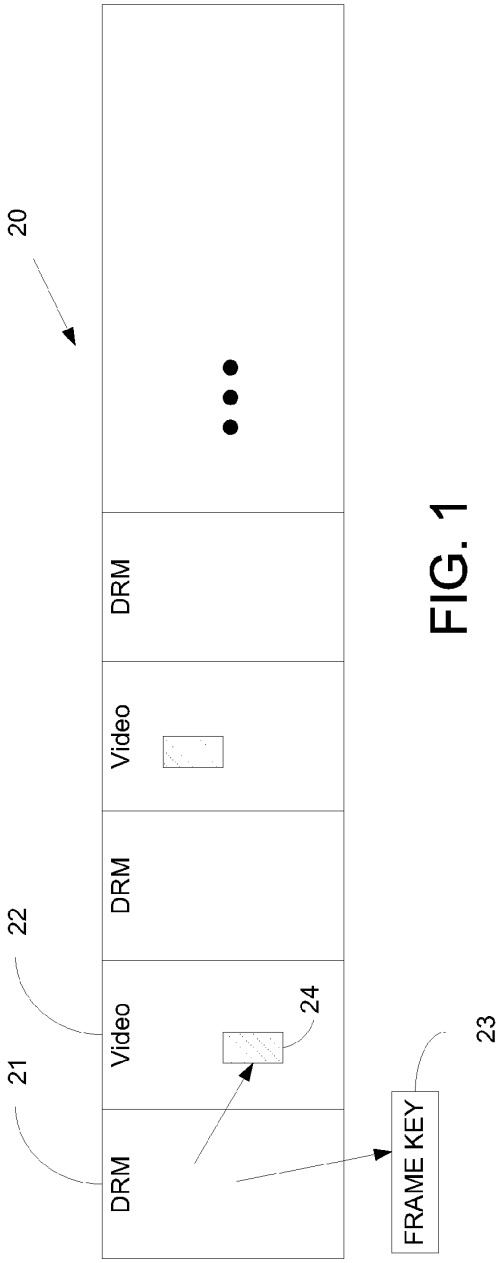


FIG. 1

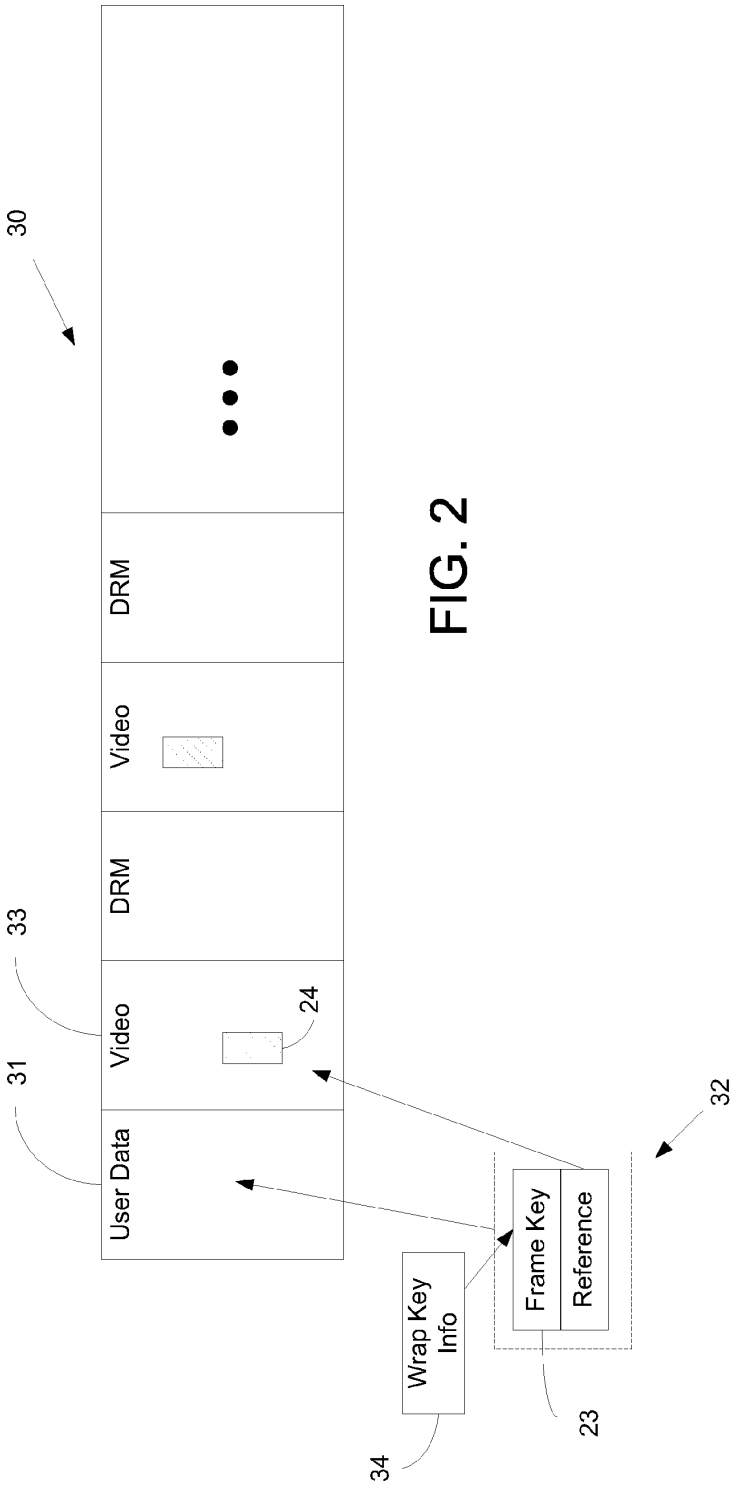

30
DRM
Video
DRM
Video
User Data
33
31
24
32
Frame Key
Reference
Wrap Key Info
23
34

FIG. 2

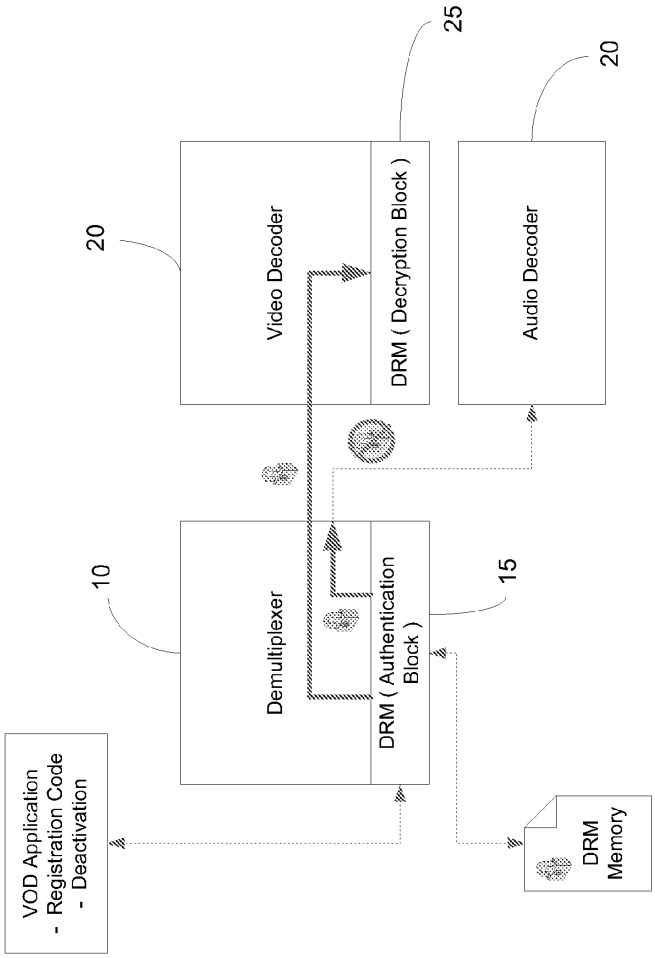


FIG. 3

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0610
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

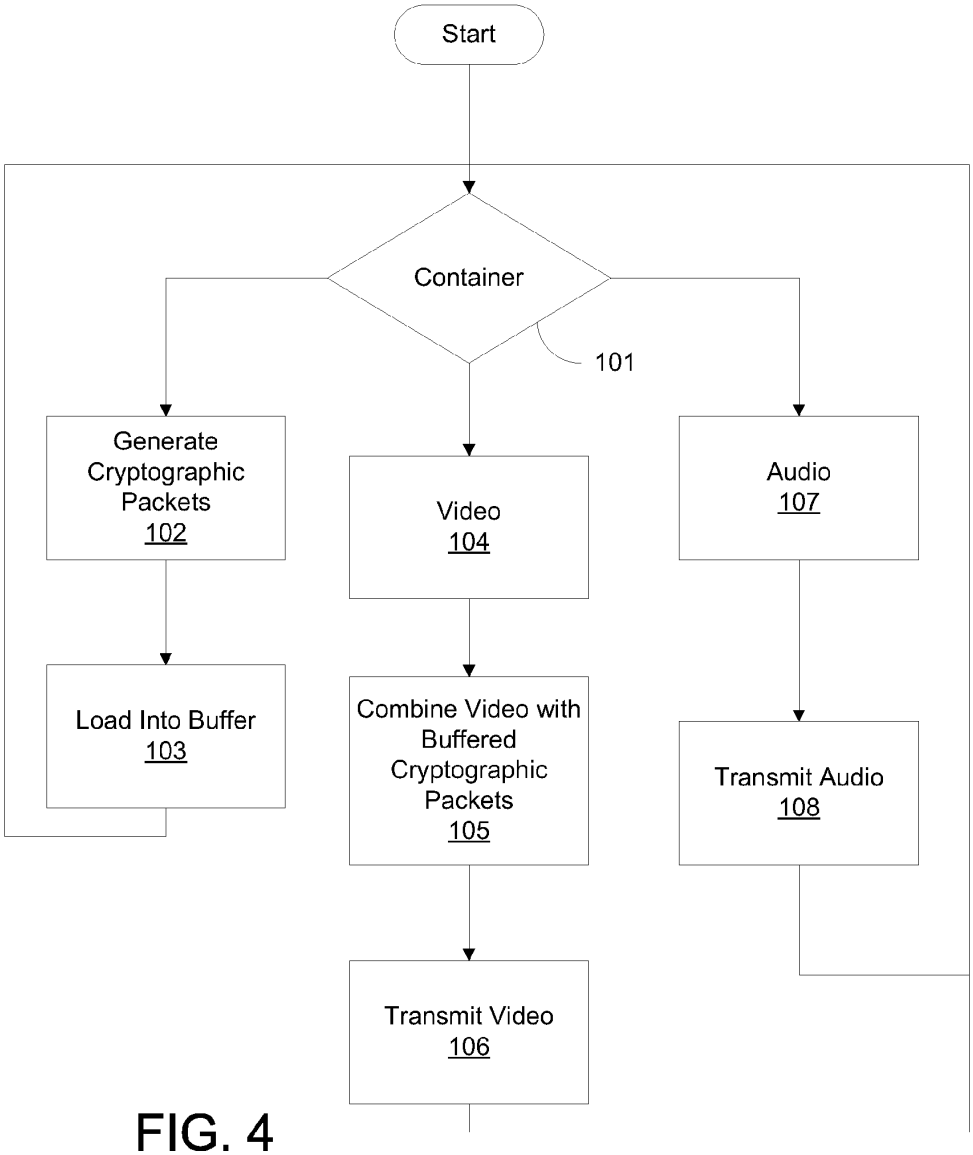

Start
Container
101
Generate Cryptographic Packets 102
Load Into Buffer 103
Video 104
Combine Video with Buffered Cryptographic Packets 105
Transmit Video 106
Audio 107
Transmit Audio 108

FIG. 4

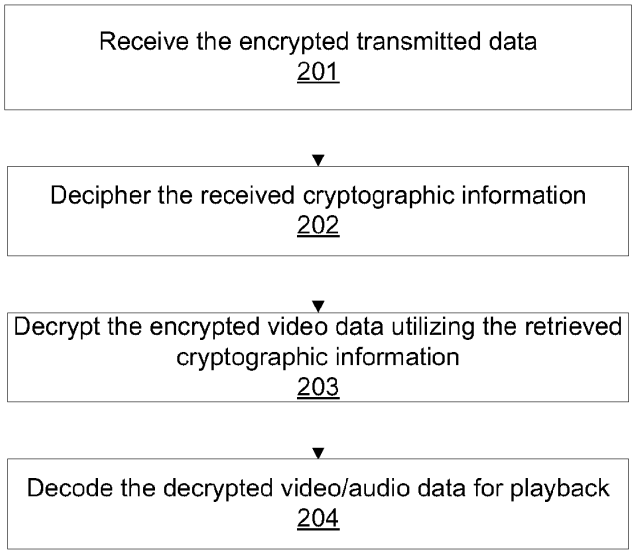

Receive the encrypted transmitted data
201
Decipher the received cryptographic information
202
Decrypt the encrypted video data utilizing the retrieved cryptographic information
203
Decode the decrypted video/audio data for playback
204

FIG. 5

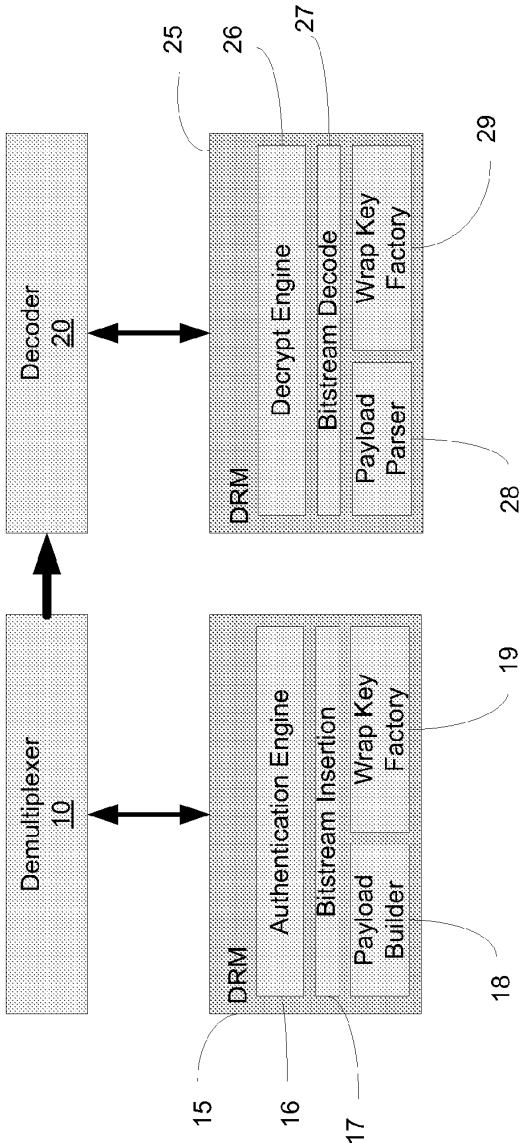

Demultiplexer
10
Decoder
20
DRM
Authentication Engine
Bitstream Insertion
Payload Builder
Wrap Key Factory
DRM
Decrypt Engine
Bitstream Decode
Payload Parser
Wrap Key Factory
15
16
17
18
19
25
26
27
28
29

FIG. 6

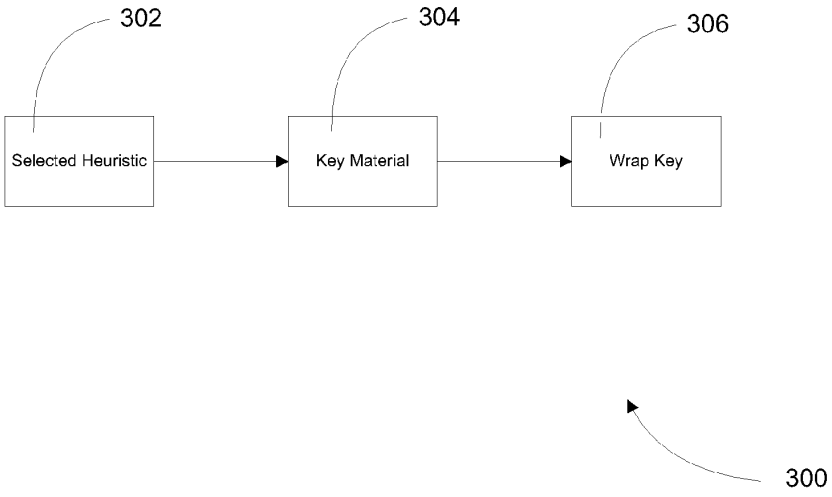


FIG. 7

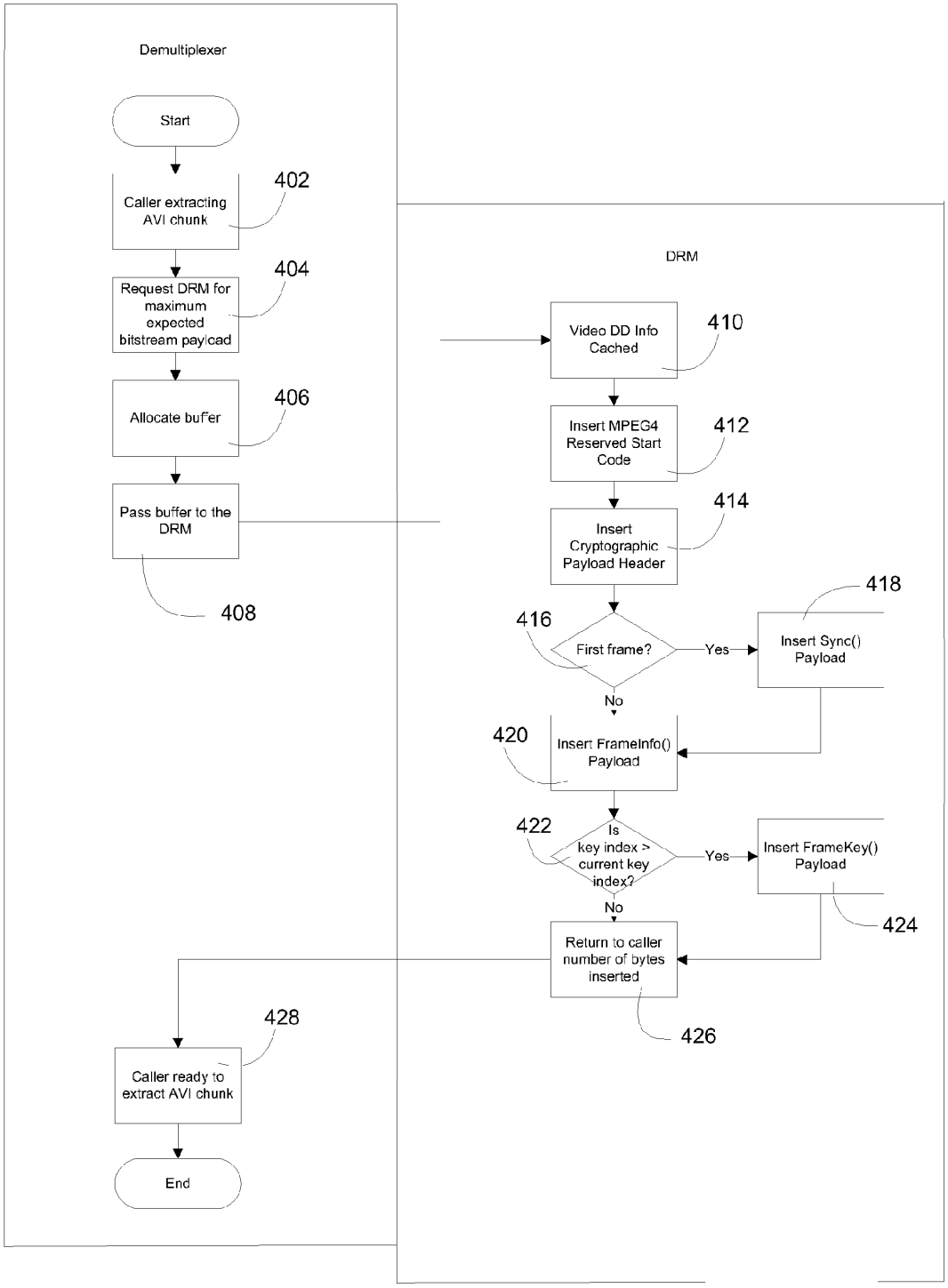

Demultiplexer
Start
Caller extracting AVI chunk
402
Request DRM for maximum expected bitstream payload
404
Allocate buffer
406
Pass buffer to the DRM
408
DRM
Video DD Info Cached
410
Insert MPEG4 Reserved Start Code
412
Insert Cryptographic Payload Header
414
First frame?
416
Yes
Insert Sync() Payload
418
No
Insert FrameInfo() Payload
420
Is key index > current key index?
422
Yes
Insert FrameKey() Payload
424
No
Return to caller number of bytes inserted
426
Caller ready to extract AVI chunk
428
End
400

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 10/56733

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04N 7/167 (2010.01)
USPC - 380/217
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
USPC - 380/217

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 380/210; 380/239; 705/51 (keyword limited; terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Dialog; Google
Transmit, sent, deliver, transport, distrib, media, multimedia, video, content, encrypt, decrypt, crypt, encipher, decipher, scramble, process, algorithm, data, stream, bitstream, container, wrapper, file, digital, rights, management, DRM, header, user, metadata, data, heu

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 2004/0081333 A1 (GRAB et al.) 29 April 2004 (29.04.2004), Para [0033], [0037]-[0042], [0047]-[0048], [0050], [0052], [0055]-[0056]	1-9 and 13-21 ---------- 10-12 and 22-25
Y	US 2003/0035546 A1 (JIANG et al.) 20 February 2003 (20.02.2003), Para [0021]	10-12 and 22-25
A	US 2003/0035545 A1 (JIANG) 20 February 2003 (20.02.2003)	10, 22, 25
A	US 2004/0105549 A1 (SUZUKI et al.) 03 June 2004 (03.06.2004)	1-25
A	US 2007/0255940 A1 (UENO) 01 November 2007 (01.11.2007)	1-25
A	US 2009/0097644 A1 (HARUKI) 16 April 2009 (16.04.2009)	1-25

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 January 2011 (03.01.2011)	14 JAN 2011
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 July 2012 (12.07.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/094181 A2

(51) International Patent Classification:
H04N 9/80 (2006.01) *H04N 5/92* (2006.01)

(21) International Application Number:
PCT/US2011/067167

(22) International Filing Date:
23 December 2011 (23.12.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/430,110 5 January 2011 (05.01.2011) US
13/224,295 1 September 2011 (01.09.2011) US
13/224,298 1 September 2011 (01.09.2011) US

(71) Applicant *(for all designated States except US)*: **DIVX, LLC.** [US/US]; 2830 De La Cruz Boulevard, Santa Clara, CA 95050 (US).

(71) Applicant *(for US only)*: **BRANESS, Jason** [US/US]; 15885 Concord Ridge Terrace, San Deigo, CA 92127 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **SOROUSHIAN, Kourosh** [US/US]; 3758 Brookshire Street, San Diego, CA 92111 (US). **DOUGLAS, Scott** [US/US]; 14934 Lone Oak Trail, Ramona, CA 92065 (US). **BRAMWELL, Steve** [US/US]; 10621 Ranch View Drive, San Diego, CA 92131 (US). **SHIFADAS, Abhishek** [IN/US]; 5562 Renaissance Ave., Unit 2, San Diego, CA 92122 (US).

(74) Agent: **BAILEY, David, J.**; Kauth, Pomeroy, Peck & Bailey LLP, 2400 E. Katella Ave., Suite 1050, Anaheim, CA 92806 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: SYSTEMS AND METHODS FOR ADAPTIVE BITRATE STREAMING OF MEDIA INCLUDING SUBTITLES

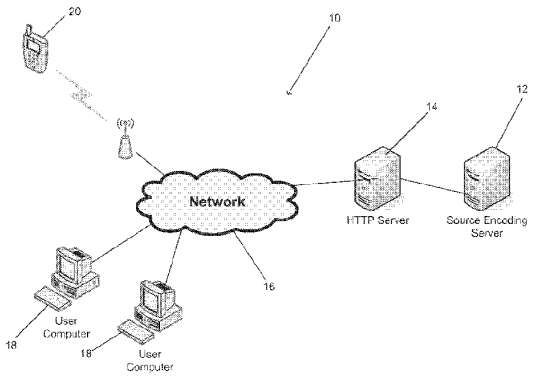


FIG. 1

(57) Abstract: Systems and methods for adaptive bitrate streaming of media including subtitles utilizing Hypertext Transfer Protocol (HTTP) in accordance with embodiments of the invention are disclosed. One embodiment of the invention includes requesting and buffering portions of video from at least one of the alternative streams using a playback device, requesting information indicative of a font utilized by a font-rendering engine to render text from a selected subtitle stream, downloading at least one font file when the font is not present on the playback device, requesting and buffering at least a portion of the selected subtitle stream, decoding the buffered portions of video using a decoder on the playback device, rendering the portions of the subtitle stream corresponding to the buffered portion of video using a font-rendering engine configured by the at least one downloaded font file, and performing synchronized playback of the decoded video and rendered subtitles using the playback device.

WO 2012/094181 A2

SYSTEMS AND METHODS FOR ADAPTIVE BITRATE STREAMING OF MEDIA INCLUDING SUBTITLES

FIELD OF THE INVENTION

[0001] The present invention generally relates to adaptive streaming and more specifically to adaptive bitrate streaming of encoded media including subtitles using Hypertext Transfer Protocol.

BACKGROUND

[0002] Presentation of textual information can be an important part of the video viewing experience. Text information may be used to represent the movie title, chapter names, specific track data, as well as subtitles. Subtitles may be used for conveying the dialogue of a video presentation in different languages, to aid those with hearing impairments, poor listening comprehension, or to suit the viewer's current listening preferences and at times to present the director or even user commentary in environments where such information is available.

[0003] Typically, embedding textual information such as subtitles with audio and video data into multimedia files involves run-length-encoding bitmap images of the subtitle text information. The run-length encoding of bitmaps provides an efficient way of storing the information, and since bitmaps are a pictorial representation of the text rather than a textual representation, there are no additional requirements to render the subtitles, such as utilizing embedded or resident font files. However, despite its advantages, because the text information is stored as bitmaps, it adapts poorly to changes in image frame size and as a result cannot be scaled with acceptable visual results to multiple sizes. In addition to the problem with scalability, bitmap representations are not easily searchable as text, which is an attractive feature from the perspective of categorization, metadata and archival activities.

[0004] The use of the actual text with respect to a known alphabet of a particular language to represent the textual information, e.g., a title, chapter names, and/or a dialogue in a movie, is one alternative to using bitmaps to represent the information. Using text in a movie typically requires the encoding of the text in a commonly acceptable representation. ASCII and Unicode are two such representations, where ASCII is typically used for encoding European languages

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0618
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

and allows a maximum of 256 symbols, and Unicode is used for representing over 100,000 characters and other symbols from a very comprehensive list of world languages.

[0005] Font files are electronic data files containing a set of representations for displaying characters or symbols. There are cases where the font for playing back subtitles may already be present on a playback device. However, often when it comes to foreign languages, or when the look-and-feel of the font is of importance from artistic, aesthetic, or readability purposes, a specific font, which has been tailored to the multimedia presentation, is provided with the multimedia content for use by the playback device. The representations or glyphs in a font file may be made in the form of individual bitmaps, drawing instructions for mathematical formulas for specifying the outline of a character, or instructions on drawing a series of lines with specific sizes and shapes. The advantage of specifying the glyph shape using drawing instructions and mathematical formulas is that the size of the character representation may be scaled to different sizes while generally maintaining their intended shape of the character. A font may also be composed of a mixture of bitmap and non-bitmap glyphs, where bitmap specifications could be used for only depicting certain character sizes.

[0006] Some languages, such as Chinese, Japanese, and Korean, utilize a unique symbol for representing each individual word in their corresponding vocabularies. The large number of words making up these different languages leads to very large font files as compared to languages which utilize combination of letters in a unique alphabet to form words. Font files for these languages may be in the 1 to 10 Mbytes range, and can sometimes be as large as 30 Mbytes. While the size of these fonts may not be an issue for processing on a personal computer, in an embedded or a consumer electronic device, such large font sizes may pose a problem especially if the fonts are expected to be dynamically available in memory for the device's rendering engine. When the size of the required representation file exceeds the resource handling capability of an embedded device, the behavior exhibited by these devices and the resulting user-experience may be non-uniform.

[0007] The term streaming media describes the playback of media on a playback device, where the media is stored on a server and continuously sent to the playback device over a network during playback. Typically, the playback device stores a sufficient quantity of media in a buffer at any given time during playback to prevent disruption of playback due to the playback

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0619
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

device completing playback of all the buffered media prior to receipt of the next portion of media. Adaptive bit rate streaming or adaptive streaming involves detecting the present streaming conditions (e.g. the user's network bandwidth and CPU capacity) in real time and adjusting the quality of the streamed media accordingly. Typically, the source media is encoded at multiple bit rates and the playback device or client switches between streaming the different encodings depending on available resources.

[0008] Adaptive streaming solutions typically utilize either Hypertext Transfer Protocol (HTTP), published by the Internet Engineering Task Force and the World Wide Web Consortium as RFC 2616, or Real Time Streaming Protocol (RTSP), published by the Internet Engineering Task Force as RFC 2326, to stream media between a server and a playback device. HTTP is a stateless protocol that enables a playback device to request a byte range within a file. HTTP is described as stateless, because the server is not required to record information concerning the state of the playback device requesting information or the byte ranges requested by the playback device in order to respond to requests received from the playback device. RTSP is a network control protocol used to control streaming media servers. Playback devices issue control commands, such as "play" and "pause", to the server streaming the media to control the playback of media files. When RTSP is utilized, the media server records the state of each client device and determines the media to stream based upon the instructions received from the client devices and the client's state.

[0009] In adaptive streaming systems, the source media is typically stored on a media server as a top level index file pointing to a number of alternate streams that contain the actual video and audio data. Each stream is typically stored in one or more container files. Different adaptive streaming solutions typically utilize different index and media containers. The Synchronized Multimedia Integration Language (SMIL) developed by the World Wide Web Consortium is utilized to create indexes in several adaptive streaming solutions including IIS Smooth Streaming developed by Microsoft Corporation of Redmond, Washington, and Flash Dynamic Streaming developed by Adobe Systems Incorporated of San Jose, California. HTTP Adaptive Bitrate Streaming developed by Apple Computer Incorporated of Cupertino, California implements index files using an extended M3U playlist file (.M3U8), which is a text file containing a list of URIs that typically identify a media container file. The most commonly used media container

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0620
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

formats are the MP4 container format specified in MPEG-4 Part 14 (i.e. ISO/IEC 14496-14) and the MPEG transport stream (TS) container specified in MPEG-2 Part 1 (i.e. ISO/IEC Standard 13818-1). The MP4 container format is utilized in IIS Smooth Streaming and Flash Dynamic Streaming. The TS container is used in HTTP Adaptive Bitrate Streaming.

[0010] The Matroska container is a media container developed as an open standard project by the Matroska non-profit organization of Aussonne, France. The Matroska container is based upon Extensible Binary Meta Language (EBML), which is a binary derivative of the Extensible Markup Language (XML). Decoding of the Matroska container is supported by many consumer electronics (CE) devices. The DivX Plus file format developed by DivX, LLC of San Diego, California utilizes an extension of the Matroska container format (i.e. is based upon the Matroska container format, but includes elements that are not specified within the Matroska format).

SUMMARY OF THE INVENTION

[0011] Systems and methods for adaptive bitrate streaming of media including subtitles utilizing Hypertext Transfer Protocol (HTTP) in accordance with embodiments of the invention are disclosed. One embodiment of the invention includes requesting and buffering portions of video from at least one of the alternative streams using a playback device, requesting information indicative of a font utilized by a font-rendering engine to render text from a selected subtitle stream, downloading at least one font file when the font is not present on the playback device, requesting and buffering at least a portion of the selected subtitle stream, decoding the buffered portions of video using a decoder on the playback device, rendering the portions of the subtitle stream corresponding to the buffered portion of video using a font-rendering engine configured by the at least one downloaded font file, and performing synchronized playback of the decoded video and rendered subtitles using the playback device.

[0012] A further embodiment includes a processor configured by a client application to request portions of files from a remote server. In addition, the client application further configures the processor to request and buffer portions of video from at least one of the alternative streams, request information indicative of a font utilized by a font-rendering engine to render text from a selected subtitle stream, download at least one font file when the font is not present on the playback device, request and buffer at least a portion of the selected subtitle

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0621
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

stream, decode the buffered portions of video, render the portions of the subtitle stream corresponding to the buffered portion of video using a font-rendering engine configured by the at least one downloaded font file, and perform synchronized playback of the decoded video and rendered subtitles.

[0013] A further embodiment of the invention includes selecting a portion of the source media using the source encoder, transcoding the selected portion of the source media into a plurality of alternative portions of encoded video using the source encoder, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame at the start of a closed group of pictures (GOP), writing each of the alternative portions of encoded video to a separate container file using the source encoder, writing a subtitle stream segment from each subtitle track corresponding to the selected portion of the source media to a separate container file using the source encoder, and associating a font file with at least one of the container files containing subtitle stream segments.

[0014] Another further embodiment includes a processor configured by a source encoding application to select a portion of the source media. In addition, the source encoding application further configures the processor to transcode the selected portion of the source media into a plurality of alternative portions of encoded video, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame at the start of a closed group of pictures (GOP), write each of the alternative portions of encoded video to a separate container file using the source encoder, write a subtitle stream segment from each subtitle track corresponding to the selected portion of the source media to a separate container file using the source encoder, and associate a font file with at least one of the container files containing subtitle stream segments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 is a network diagram of an adaptive bitrate streaming system in accordance with an embodiment of the invention.

[0016] FIG. 2 conceptually illustrates a top level index file and Matroska container files generated by the encoding of source media in accordance with embodiments of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0622
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0017] FIG. 3 conceptually illustrates a specialized Matroska container file incorporating a modified Cues element in accordance with an embodiment of the invention.

[0018] FIG. 3a conceptually illustrates a specialized Matroska container file incorporating a modified Attachments element in which metadata describing a font file is separated from the font file attached to the container file in accordance with an embodiment of the invention.

[0019] FIGS. 4a – 4c conceptually illustrate the insertion of different types of media into the Clusters element of a Matroska container file subject to various constrains that facilitate adaptive bitrate streaming in accordance with embodiments of the invention.

[0020] FIG. 4d conceptually illustrates the multiplexing of different types of media into the Clusters element of a Matroska container file subject to various constraints that facilitate adaptive bitrate streaming in accordance with an embodiment of the invention.

[0021] FIG. 4e conceptually illustrates the inclusion of a trick play track into the Clusters element of a Matroska container file subject to various constraints that facilitate adaptive bitrate streaming in accordance with an embodiment of the invention.

[0022] FIG. 5 conceptually illustrates a modified Cues element of a specialized Matroska container file, where the Cues element includes information enabling the retrieval of Cluster elements using HTTP byte range requests in accordance with an embodiment of the invention.

[0023] FIG. 5a conceptually illustrates a modified Cues element of a specialized Matroska container file in accordance with an embodiment of the invention, where the Cues element is similar to the Cues element shown in FIG. 5 with the exception that attributes that are not utilized during adaptive bitrate streaming are removed.

[0024] FIG. 6 conceptually illustrates the indexing of Cluster elements within a specialized Matroska container file utilizing modified CuePoint elements within the container file in accordance with embodiments of the invention.

[0025] FIG. 7 is a flow chart illustrating a process for encoding source media for adaptive bitrate streaming in accordance with an embodiment of the invention.

[0026] FIG. 8 conceptually illustrates communication between a playback device and an HTTP server associated with the commencement of streaming of encoded media contained within Matroska container files indexed by a top level index file in accordance with an embodiment of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0623
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0027] FIGS. 9a and 9b conceptually illustrate communication between a playback device and an HTTP server associated with switching between streams in response to the streaming conditions experienced by the playback device and depending upon the index information available to the playback device prior to the decision to switch streams in accordance with embodiments of the invention.

DETAILED DISCLOSURE OF THE INVENTION

[0028] Turning now to the drawings, systems and methods for adaptive bitrate streaming of media including subtitles utilizing Hypertext Transfer Protocol (HTTP) in accordance with embodiments of the invention are illustrated. In a number of embodiments, subtitle streams are encoded using an encoding such as (but not limited to) Unicode text and playback devices retrieve font files from a remote server when a user requests playback of subtitles that are rendered using a font that is not already present on the device. In a number of embodiments, the font file is subsetted so that it only includes the characters used within the streamed subtitles. In several embodiments, a lossless compression process is applied to the font prior to transmission and the playback device decompresses the received font file prior to utilization by a font-rendering engine on the playback device. The decoding device may query the total size of the font file (whether compressed or uncompressed) and after determining the bandwidth throughput, present a message to the user indicating the total estimated download time. After initiating the actual downloading of the font, the decoding device may present a progress notice indicating the total amount of the font file that has been downloaded, relative to the entire font size. The user may cancel the downloading of the font at any time during the download process, along with an optional message indicating that the selected subtitle cannot be displayed without the availability of the associated font.

[0029] In a number of embodiments, source media including one or more sets of subtitles or subtitles tracks synchronized to the video content is encoded as a number of alternative video streams and a separate subtitle stream for each of the sets of subtitles. Each stream is stored in a Matroska (MKV) container file. In many embodiments, the Matroska container file is a specialized Matroska container file in that the manner in which the media in each stream is encoded and stored within the container is constrained to improve streaming performance. A top

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0624
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

level index file containing an index to the streams contained within each of the container files is also generated to enable adaptive bitrate streaming of the encoded media. In many embodiments, the top level index file is a Synchronized Multimedia Integration Language (SMIL) file containing URIs for each of the Matroska container files. In other embodiments, any of a variety of file formats can be utilized in the generation of the top level index file.

[0030] In a number of embodiments, the font used to render the subtitles in a subtitle stream is included in the container file containing the subtitle stream. In several embodiments, a font file is embedded in a Matroska container file in a manner similar to that described in U.S. Patent Application Serial No. 12/480,276 entitled "Systems and Methods for Font File Optimization for Multimedia Files", to Priyadarshi et al., filed June 8, 2009. In many embodiments, the font file is referenced by the top level index file and is stored separately from the container file containing the subtitle that utilizes the font.

[0031] The performance of an adaptive bitstrate streaming system in accordance with embodiments of the invention can be significantly enhanced by encoding each portion of the source video at each bit rate in such a way that the portion of video is encoded in each stream as a single (or at least one) closed group of pictures (GOP) starting with an Instantaneous Decoder Refresh (IDR) frame. The GOP for each stream can then be stored as a Cluster element within the Matroska container file for the stream. In this way, the playback device can switch between streams at the completion of the playback of a Cluster and, irrespective of the stream from which a Cluster is obtained the first frame in the Cluster will be an IDR frame and can be decoded without reference to any encoded media other than the encoded media contained within the Cluster element. In many embodiments, the sections of the source video that are encoded as GOPs are all the same duration. In a number of embodiments each two second sequence of the source video is encoded as a GOP. The performance of the system can be further improved by subsetting fonts used to render subtitle streams accompanying the source video so that the size of the font files do not exceed the amount of memory that can be allocated to the font-rendering engine of the playback device during playback. In many embodiments, the font file is subsetted based upon the specific characters from the font present in the subtitle stream. In a number of embodiments, multiple font files are provided that each include the characters from a different portion or segment of the subtitle stream.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0625
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0032] Retrieval of media using HTTP during adaptive streaming can be improved by adding additional index information to the Matroska container files used to contain each of the encoded streams. In a number of embodiments, the index is a reduced index in that the index only points to the IDRs at the start of each cluster. In many embodiments, the index of the Matroska container file includes additional non-standard attributes (i.e. attributes that do not form part of the Matroska container file format specification) that specify the size of each of the clusters so that a playback device can retrieve a Cluster element from the Matroska container file via HTTP using a byte range request.

[0033] Adaptive streaming of source media encoded in the manner outlined above can be coordinated by a playback device in accordance with embodiments of the invention. The playback device obtains information concerning each of the available streams from the top level index file and selects one or more streams to utilize in the playback of the media. The playback device can then obtain header information from the Matroska container files containing the one or more bitstreams or streams, and the headers provide information concerning the decoding of the streams. Where the stream is a subtitle stream, the playback device can determine whether the font utilized by the subtitle stream is present on the playback device and the playback device can request the font file when the font is not present. The playback device can also request index information that indexes the encoded media stored within the relevant Matroska container files. The index information can be stored within the Matroska container files or separately from the Matroska container files in the top level index or in separate index files. The index information enables the playback device to request byte ranges corresponding to Cluster elements within the Matroska container file containing specific portions of encoded media via HTTP from the server. As the playback device receives the Cluster elements from the HTTP server, the playback device can evaluate current streaming conditions to determine whether to increase or decrease the bitrate of the streamed media. In the event that the playback device determines that a change in bitrate is necessary, the playback device can obtain header information and index information for the container file(s) containing the desired alternative stream(s) (assuming the playback device has not already obtained this information). The index information can then be used to identify the byte range of the Cluster element containing the next portion of the source media encoded at the desired bit rate and the identified Cluster element can be retrieved from the server via HTTP.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0626
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The next portion of the source media that is requested is typically identified based upon the Cluster elements already requested by the playback device and the Cluster elements buffered by the playback device. The next portion of source media requested from the alternative stream is requested to minimize the likelihood that the buffer of the playback device will underflow (i.e. run out media to playback) prior to receipt of the Cluster element containing the next portion of source media by the playback device. In this way, the playback device can achieve adaptive bitrate streaming by retrieving sequential Cluster elements from the various streams as appropriate to the streaming conditions using the top level index and index information describing the Cluster elements within each of the Matroska container files.

[0034] In a number of embodiments, variation in the bitrate between different streams of encoded video can be achieved by modifying the encoding parameters for each stream including but not limited to the bitrate, frame rate, and resolution. When different streams include different resolutions, the display aspect ratio of each stream is the same and the sample aspect ratios are modified to ensure smooth transitions from one resolution to another. The encoding of source media including subtitles for use in adaptive bitrate streaming and the playback of the encoded source media using HTTP requests to achieve adaptive bitrate streaming with synchronous display of subtitles in accordance with embodiments of the invention is discussed further below.

ADAPTIVE STREAMING SYSTEM ARCHITECTURE

[0035] An adaptive streaming system in accordance with an embodiment of the invention is illustrated in FIG. 1. The adaptive streaming system 10 includes a source encoder 12 configured to encode source media including one or more subtitle tracks as a number of alternative video streams and a separate subtitle stream for each of the subtitle tracks accompanying the video. Where a font file is provided, the source encoder can store the font file corresponding to each subtitle stream with the subtitle stream and/or can reduce the size of the font file by eliminating characters that do not form part of the subtitle stream from the original font file. In further embodiments, the source encoder generates multiple smaller font files based upon constraints including (but not limited to) memory constraints imposed by playback devices and/or associates the smaller font files with portions of the subtitle stream. In the illustrated embodiment, the source encoder is a server. In other embodiments, the source encoder can be any processing

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0627
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

device including a processor and sufficient resources to perform the transcoding of source media (including but not limited to video, audio, and/or subtitles). As is discussed further below, the source encoding server 12 generates a top level index to a plurality of container files containing the streams, at least a plurality of which are alternative video streams and one or more of which are subtitle streams that are synchronized to the video streams. Alternative streams are streams that encode the same media content in different ways. In many instances, alternative streams encode media content (such as but not limited to video) at different bitrates. In a number of embodiments, the alternative streams are encoded with different resolutions and/or at different frame rates. The top level index file and the container files are uploaded to an HTTP server 14. A variety of playback devices can then use HTTP or another appropriate stateless protocol to request portions of the top level index file and the container files via a network 16 such as the Internet.

[0036] In many embodiments, the top level index file is a SMIL file and the media is stored in Matroska container files. As is discussed further below, the media can be stored within the Matroska container file in a way that facilitates the adaptive bitrate streaming of the media. In many embodiments, the Matroska container files are specialized Matroska container files that include enhancements (i.e. elements that do not form part of the Matroska file format specification) that facilitate the retrieval of specific portions of media via HTTP during the adaptive bitrate streaming of the media.

[0037] In the illustrated embodiment, playback devices include personal computers 18 and mobile phones 20. In other embodiments, playback devices can include consumer electronics devices such as DVD players, Blu-ray players, televisions, set top boxes, video game consoles, tablets, and other devices that are capable of connecting to a server via HTTP and playing back encoded media. Although a specific architecture is shown in FIG. 1 any of a variety of architectures can be utilized that enable playback devices to request portions of the top level index file and the container files in accordance with embodiments of the invention.

FILE STRUCTURE

[0038] Files generated by a source encoder and/or stored on an HTTP server for streaming to playback devices in accordance with embodiments of the invention are illustrated in FIG. 2. The

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0628
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

files utilized in the adaptive bitrate streaming of the source media include a top level index 30 and a plurality of container files 32 that each contain at least one stream. The top level index file describes the content of each of the container files. As is discussed further below, the top level index file can take a variety of forms including an SMIL file and the container files can take a variety of forms including a specialized Matroska container file.

[0039] In many embodiments, each Matroska container file contains a single stream. For example, the stream could be one of a number of alternate video streams, an audio stream, one of a number of alternate audio streams, a subtitle stream, one of a number of alternate subtitle streams, a trick play stream, or one of a number of alternate trick play streams. In several embodiments, the Matroska container file includes multiple multiplexed streams. For example, the Matroska container could include a video stream, and one or more audio streams, one or more subtitle streams, and/or one or more trick play streams. As is discussed further below, in many embodiments the Matroska container files are specialized files. The encoding of the media and the manner in which the media is stored within Cluster elements within the Matroska container file can be subject to constraints designed to enhance the performance of an adaptive bitrate streaming system. In addition, the Matroska container file can include index elements that facilitate the location and downloading of Cluster elements from the various Matroska container files during the adaptive streaming of the media. Top level index files and Matroska container files that can be used in adaptive bitrate streaming systems in accordance with embodiments of the invention are discussed below.

TOP LEVEL INDEX FILES

[0040] Playback devices in accordance with many embodiments of the invention utilize a top level index file to identify the container files that contain the streams available to the playback device for use in adaptive bitrate streaming. In many embodiments, the top level index files can include references to container files that each include an alternative stream of encoded media. The playback device can utilize the information in the top level index file to retrieve encoded media from each of the container files according to the streaming conditions experienced by the playback device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0629
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0041] In several embodiments, the top level index file provides information enabling the playback device to retrieve information concerning the encoding of the media in each of the container files and an index to encoded media within each of the container files. In a number of embodiments, each container file includes information concerning the encoded media contained within the container file and an index to the encoded media within the container file and the top level index file indicates the portions of each container file containing this information. Therefore, a playback device can retrieve the top level index file and use the top level index file to request the portions of one or more of the container files that include information concerning the encoded media contained within the container file and an index to the encoded media within the container file. A variety of top level index files that can be utilized in adaptive bitrate streaming systems in accordance with embodiments of the invention are discussed further below.

TOP LEVEL INDEX SMIL FILES

[0042] In a number of embodiments, the top level index file utilized in the adaptive bitrate streaming of media is a SMIL file or dynamically generated SMIL data, which is XML that includes a list of URIs describing each of the streams and the container files that contain the streams. The URI can include information such as the "system-bitrate" of the stream contained within the stream and information concerning the location of specific pieces of data within the container file.

[0043] The basic structure of a SMIL file involves providing an XML declaration and a SMIL element. The SMIL element defines the streams available for use in adaptive bitrate streaming and includes a HEAD element, which is typically left empty and a BODY element that typically only contains a PAR (parallel) element. The PAR element describes streams that can be played simultaneously (i.e. include media that can be presented at the same time).

[0044] The SMIL specification defines a number of child elements to the PAR element that can be utilized to specify the streams available for use in adaptive bitrate streaming. The VIDEO, AUDIO and TEXTSTREAM elements can be utilized to define a specific video, audio or subtitle stream. The VIDEO, AUDIO and TEXTSTREAM elements can collectively be referred to as media objects. The basic attributes of a media object are the SRC attribute, which specifies the full path or a URI to a container file containing the relevant stream, and the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0630
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

XML:LANG attribute, which includes a 3 letter language code. Additional information concerning a media object can be specified using the PARAM element. The PARAM element is a standard way within the SMIL format for providing a general name value pair. In a number of embodiments of the invention, specific PARAM elements are defined that are utilized during adaptive bitrate streaming.

[0045] In many embodiments, a "header-request" PARAM element is defined that specifies the size of the header section of the container file containing the stream. The value of the "header-request" PARAM element typically specifies the number of bytes between the start of the file and the start of the encoded media within the file. In many embodiments, the header contains information concerning the manner in which the media is encoded and a playback device retrieves the header prior to playback of the encoded media in order to be able to configure the decoder for playback of the encoded media. An example of a "header-request" PARAM element is follows:

```
<param
  name="header-request"
  value="1026"
  valuetype="data" />
```

[0046] In a number of embodiments, a "mime" PARAM element is defined that specifies the MIME type of the stream. A "mime" PARAM element that identifies the stream as being an H.264 stream (i.e. a stream encoded in accordance with the MPEG-4 Advanced Video Codec standard) is as follows:

```
<param
  name="mime"
  value="V_MPEG4/ISO/AVC"
  valuetype="data" />
```

[0047] The MIME type of the stream can be specified using a "mime" PARAM element as appropriate to the encoding of a specific stream (e.g. AAC audio or UTF-8 text stream).

[0048] When the media object is a VIDEO element, additional attributes are defined within the SMIL file format specification including the systemBitrate attribute, which specifies the bitrate of the stream in the container file identified by the VIDEO element, and width and height attributes, which specify the dimensions of the encoded video in pixels. Additional attributes can

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0631
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

also be defined using the PARAM element. In several embodiments, a "vbv" PARAM element is defined that specified the VBV buffer size of the video stream in bytes. The video buffering verifier (VBV) is a theoretical MPEG video buffer model used to ensure that an encoded video stream can be correctly buffered and played back at the decoder device. An example of a "vbv" PARAM element that specifies a VBV size of 1000 bytes is as follows:

```
<param
  name="vbv"
  value="1000"
  valuetype="data" />
```

[0049] An example of VIDEO element including the attributes discussed above is as follows:

```
<video
        src="http://cnd.com/video1_620kbps.mkv"
        systemBitrate="620"
        width="480"
        height="270" >
        <param
          name="vbv"
          value="1000"
          valuetype="data" />
</video>
```

[0050] Adaptive bitrate streaming systems in accordance with embodiments of the invention can support trick play streams, which can be used to provide smooth visual search through source content encoded for adaptive bitrate streaming. A trick play stream can be encoded that appears to be an accelerated visual search through the source media when played back, when in reality the trick play stream is simply a separate track encoding the source media at a lower frame rate. In many embodiments of the system a VIDEO element that references a trick play track is indicated by the systemProfile attribute of the VIDEO element. In other embodiments, any of a variety of techniques can be utilized to signify within the top level index file that a specific stream is a trick play stream. An example of a trick play stream VIDEO element in accordance with an embodiment of the invention is as follows:

```
<video
        src="http://cnd.com/video_test2_600kbps.mkv"
        systemProfile="DivXPlusTrickTrack"
        width="480"
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0632
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
    height="240">
    <param name="vbv" value="1000" valuetype="data" />
    <param name="header-request" value="1000" valuetype="data" />
  </video>
```

[0051] In a number of embodiments of the invention, a "reservedBandwidth" PARAM element can be defined for an AUDIO element. The "reservedBandwidth" PARAM element specifies the bitrate of the audio stream in Kbps. An example of an AUDIO element specified in accordance with an embodiment of the invention is as follows:

```
<audio
   src="http://cnd.com/audio_test1_277kbps.mkv"
   xml:lang="gem"
<param
  name="reservedBandwidth"
  value="128"
  valuetype="data" />
/>
```

[0052] In several embodiments, the "reservedBandwidth" PARAM element is also defined for a TEXTSTREAM element. An example of a TEXTSTREAM element including a "reservedBandwidth" PARAM element in accordance with an embodiment of the invention is as follows:

```
<textstream
   src="http://cnd.com/text_stream_ger.mkv"
   xml:lang="gem"
<param
  name="reservedBandwidth"
  value="32"
  valuetype="data" />
/>
```

[0053] In a number of embodiments, the top level index includes a manifest listing all of the available subtitle segments, the time-ranges covered by each segment, and URIs corresponding to the segments. A decoding device can decide which segment to download based on the current time-code, while ensuring that the correct segment is available prior to presenting the associated video and audio. In several embodiments, the top level index includes a manifest listing all of the available font segments, the time-ranges covered by each segment, and a URI corresponding to

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0633
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

those segments. In other embodiments, the manifest associated with the segments of the subtitle streams and/or the font file(s) is contained within the container file that contains the subtitle stream. For both fonts and subtitles, the time-range may cover a few seconds, minutes, or the entire duration of the associated audio and video.

[0054] In other embodiments, any of a variety of mechanisms can be utilized to specify information concerning VIDEO, AUDIO, and SUBTITLE elements as appropriate to specific applications.

[0055] A SWITCH element is a mechanism defined within the SMIL file format specification that can be utilized to define adaptive or alternative streams. An example of the manner in which a SWITCH element can be utilized to specify alternative video streams at different bitrates is as follows:

```
<switch>
  <video src="http://cnd.com/video_test1_300kbps.mkv"/>
  <video src="http://cnd.com/video_test2_900kbps.mkv"/>
  <video src="http://cnd.com/video_test3_1200kbps.mkv"/>
</switch>
```

[0056] The SWTICH element specifies the URLs of three alternative video streams. The file names indicate that the different bitrates of each of the streams. As is discussed further below, the SMIL file format specification provides mechanisms that can be utilized in accordance with embodiments of the invention to specify within the top level index SMIL file additional information concerning a stream and the container file in which it is contained.

[0057] In many embodiments of the invention, the EXCL (exclusive) element is used to define alternative tracks that do not adapt during playback with streaming conditions. For example, the EXCL element can be used to define alternative audio tracks or alternative subtitle tracks. An example of the manner in which an EXCL element can be utilized to specify alternative English and French audio streams is as follows:

```
<excl>
  <audio
    src="http://cnd.com/english-audio.mkv"
    xml:lang="eng"/>
  <audio
    src="http://cnd.com/french-audio.mkv"
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0634
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
    xml:lang="fre"/>
</excl>
```

[0058] An example of a top level index SMIL file that defines the attributes and parameters of two alternative video levels, an audio stream and a subtitle stream in accordance with an embodiment of the invention is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
  <head>
  </head>
  <body>
    <par>
      <switch>
        <video
          src="http://cnd.com/video_test1_300kbps.mkv"
          systemBitrate="300"
          vbv="600"
          width="320"
          height="240" >
          <param
            name="vbv"
            value="600"
            valuetype="data" />
          <param
            name="header-request"
            value="1000"
            valuetype="data" />
        </video>
        <video
          src="http://cnd.com/video_test2_600kbps.mkv"
          systemBitrate="600"
          vbv ="900"
          width="640"
          height="480">
          <param
            name="vbv"
            value="1000"
            valuetype="data" />
          <param
            name="header-request"
            value="1000"
            valuetype="data" />
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0635
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
        </video>
      </switch>
      <audio
        src="http://cnd.com/audio.mkv"
        xml:lang="eng">
        <param
          name="header-request"
          value="1000"
          valuetype="data" />
        <param name="reservedBandwidth" value="128" valuetype="data" />
      </audio>
      <textstream
        src="http://cnd.com/subtitles.mkv"
        xml:lang="eng">
        <param
          name="header-request"
          value="1000"
          valuetype="data" />
        <param name="reservedBandwidth" value="32" valuetype="data" />
      </textstream>
    </par>
  </body>
</smil>
```

[0059] The top level index SMIL file can be generated when the source media is encoded for playback via adaptive bitrate streaming. Alternatively, the top level index SMIL file can be generated when a playback device requests the commencement of playback of the encoded media. When the playback device receives the top level index SMIL file, the playback device can parse the SMIL file to identify the available streams. The playback device can then select the streams to utilize to playback the content and can use the SMIL file to identify the portions of the container file to download to obtain information concerning the encoding of a specific stream and/or to obtain an index to the encoded media within the container file.

[0060] Although top level index SMIL files are described above, any of a variety of top level index file formats can be utilized to create top level index files as appropriate to a specific application in accordance with an embodiment of the invention. In addition, top level indexes in accordance with embodiments of the invention can provide URIs indicating the location from which a font file can be downloaded. The use of top level index files to enable playback of

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0636
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

encoded media using adaptive bitrate streaming in accordance with embodiments of the invention is discussed further below.

STORING MEDIA IN MATROSKA FILES FOR ADAPTIVE BITRATE STREAMING

[0061] A Matroska container file used to store encoded video in accordance with an embodiment of the invention is illustrated in FIG. 3. The container file 32 is an Extensible Binary Markup Language (EBML) file that is an extension of the Matroska container file format. The specialized Matroska container file 32 includes a standard EBML element 34, and a standard Segment element 36 that includes a standard Seek Head element 40, a standard Segment Information element 42, and a standard Tracks element 44. These standard elements describe the media contained within the Matroska container file. The Segment element 36 also includes a standard Clusters element 46. As is described below, the manner in which encoded media is inserted within individual Cluster elements 48 within the Clusters element 46 is constrained to improve the playback of the media in an adaptive streaming system. Where the Matroska container contains encoded video, the constraints imposed upon the encoded video are consistent with the specification of the Matroska container file format and involve encoding the video so that each cluster includes at least one closed GOP commencing with an IDR frame. In addition to the above standard elements, the Segment element 36 also includes a modified version of the standard Cues element 52. As is discussed further below, the Cues element includes specialized CuePoint elements (i.e. non-standard CuePoint elements) that facilitate the retrieval of the media contained within specific Cluster elements via HTTP. In a number of instances, the Matroska file includes an Attachments element 56. In embodiments where one or more font files are included in the container file the font files are contained within the Attachments element.

[0062] The constraints imposed upon the encoding of media and the formatting of the encoded media within the Clusters element of a Matroska container file for adaptive bitrate streaming and the additional index information inserted within the container file in accordance with embodiments of the invention is discussed further below.

ENCODING MEDIA FOR INSERTION IN CLUSTER ELEMENTS

[0063] An adaptive bitrate streaming system provides a playback device with the option of selecting between different streams of encoded media during playback according to the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0637
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

streaming conditions experienced by the playback device. In many embodiments, switching between streams is facilitated by separately pre-encoding discrete portions of the source media in accordance with the encoding parameters of each stream and then including each separately encoded portion in its own Cluster element within the stream's container file. Furthermore, the media contained within each cluster is encoded so that the media is capable of playback without reference to media contained in any other cluster within the stream. In this way, each stream includes a Cluster element corresponding to the same discrete portion of the source media and, at any time, the playback device can select the Cluster element from the stream that is most appropriate to the streaming conditions experienced by the playback device and can commence playback of the media contained within the Cluster element. Accordingly, the playback device can select clusters from different streams as the streaming conditions experienced by the playback device change over time. In several embodiments, the Cluster elements are further constrained so that each Cluster element contains a portion of encoded media from the source media having the same duration. In a number of embodiments, each Cluster element includes two seconds of encoded media. The specific constraints applied to the media encoded within each Cluster element depending upon the type of media (i.e. video, audio, or subtitles) are discussed below.

[0064] A Clusters element of a Matroska container file containing a video stream in accordance with an embodiment of the invention is illustrated in FIG. 4a. The Clusters element 46 includes a plurality of Cluster elements 48 that each contains a discrete portion of encoded video. In the illustrated embodiment, each Cluster element 48 includes two seconds of encoded video. In other embodiments, the Cluster elements include encoded video having a greater or lesser duration than two seconds. The smaller the Cluster elements (i.e. the smaller the duration of the encoded media within each Cluster element), the higher the overhead associated with requesting each Cluster element. Therefore, a tradeoff exists between the responsiveness of the playback device to changes in streaming conditions and the effective data rate of the adaptive streaming system for a given set of streaming conditions (i.e. the portion of the available bandwidth actually utilized to transmit encoded media). In several embodiments, the encoded video sequences in the Cluster elements have different durations. Each Cluster element 48 includes a Timecode element 60 indicating the start time of the encoded video within the Cluster

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0638
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

element and a plurality of BlockGroup elements. As noted above, the encoded video stored within the Cluster is constrained so that the encoded video can be played back without reference to the encoded video contained within any of the other Cluster elements in the container file. In many embodiments, encoding the video contained within the Cluster element as a GOP in which the first frame is an IDR frame enforces the constraint. In the illustrated embodiment, the first BlockGroup element 62 contains an IDR frame. Therefore, the first BlockGroup element 62 does not include a ReferenceBlock element. The first BlockGroup element 62 includes a Block element 64, which specifies the Timecode attribute of the frame encoded within the Block element 64 relative to the Timecode of the Cluster element 48. Subsequent BlockGroup elements 66 are not restricted in the types of frames that they can contain (other than that they cannot reference frames that are not contained within the Cluster element). Therefore, subsequent BlockGroup elements 66 can include ReferenceBlock elements 68 referencing other BlockGroup element(s) utilized in the decoding of the frame contained within the BlockGroup or can contain IDR frames and are similar to the first BlockGroup element 62. As noted above, the manner in which encoded video is inserted within the Cluster elements of the Matroska file conforms with the specification of the Matroska file format.

[0065] The insertion of encoded audio and subtitle information within a Clusters element 46 of a Matroska container file in accordance with embodiments of the invention is illustrated in FIGS. 4b and 4c. In the illustrated embodiments, the encoded media is inserted within the Cluster elements 48 subject to the same constraints applied to the encoded video discussed above with respect to FIG. 4a. The Cluster elements within the container files containing the audio and/or subtitle streams need not, however, correspond with the start time and duration of the Cluster elements in the container files containing the alternative video streams.

MULIPLEXING STREAMS IN A SINGLE MKV CONTAINER FILE

[0066] The Clusters elements shown in FIGS. 4a – 4c assume that a single stream is contained within each Matroska container file. In several embodiments, media from multiple streams is multiplexed within a single Matroska container file. In this way, a single container file can contain a video stream multiplexed with one or more corresponding audio streams, and/or one or more corresponding subtitle streams. Storing the streams in this way can result in

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0639
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

duplication of the audio and subtitle streams across multiple alternative video streams. However, the seek time to retrieve encoded media from a video stream and an associated audio, and/or subtitle stream can be reduced due to the adjacent storage of the data on the server. The Clusters element 46 of a Matroska container file containing multiplexed video, audio and subtitle data in accordance with an embodiment of the invention is illustrated in FIG. 4d. In the illustrated embodiment, each Cluster element 48 includes additional BlockGroup elements for each of the multiplexed streams. The first Cluster element includes a first BlockGroup element 62v for encoded video that includes a Block element 64v containing an encoded video frame and indicating the Timecode attribute of the frame relative to the start time of the Cluster element (i.e. the Timecode attribute 60). A second BlockGroup element 62a includes a Block element 64a including an encoded audio sequence and indicating the timecode of the encoded audio relative to the start time of the Cluster element, and a third BlockGroup element 62s including a Block element 64s containing an encoded subtitle and indicating the timecode of the encoded subtitle relative to the start time of the Cluster element. Although not shown in the illustrated embodiment, each Cluster element 48 likely would include additional BlockGroup elements containing additional encoded video, audio or subtitles. Despite the multiplexing of the encoded video, audio, and/or subtitle streams, the same constraints concerning the encoded media apply.

INCORPORATING TRICK PLAY TRACKS IN MKV CONTAINER FILES FOR USE IN ADAPTIVE BITRATE STREAMING SYSTEMS

[0067] The incorporation of trick play tracks within Matroska container files is proposed by DivX, LLC in U.S. Patent Application No. 12/260,404 entitled "Application Enhancement Tracks", filed October 29, 2008, the disclosure of which is hereby incorporated by reference in its entirety. Trick play tracks similar to the trick play tracks described in U.S. Patent Application No. 12/260,404 can be used to provide a trick play stream in an adaptive bitrate streaming system in accordance with an embodiment of the invention to provide smooth visual search through source content encoded for adaptive bitrate streaming. A separate trick play track can be encoded that appears to be an accelerated visual search through the source media when played back, when in reality the trick play track is simply a separate track encoding the source media at a lower frame rate. In several embodiments, the tick play stream is created by generating a trick

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0640
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

play track in the manner outlined in U.S. Patent Application No. 12/260,404 and inserting the trick play track into a Matroska container file subject to the constraints mentioned above with respect to insertion of a video stream into a Matroksa container file. In many embodiments, the trick play track is also subject to the further constraint that every frame in the GOP of each Cluster element in the trick play track is encoded as an IDR frame. As with the other video streams, each Cluster element contains a GOP corresponding to the same two seconds of source media as the corresponding Cluster elements in the other streams. There are simply fewer frames in the GOPs of the trick play track and each frame has a longer duration. In this way, transitions to and from a trick play stream can be treated in the same way as transitions between any of the other encoded streams are treated within an adaptive bitrate streaming system in accordance with embodiments of the invention. Playback of the frames contained within the trick play track to achieve accelerated visual search typically involves the playback device manipulating the timecodes assigned to the frames of encoded video prior to providing the frames to the playback device's decoder to achieve a desired increase in rate of accelerated search (e.g. x2, x4, x6, etc.).

[0068] A Clusters element containing encoded media from a trick play track is shown in FIG. 4e. In the illustrated embodiment, the encoded trick play track is inserted within the Cluster elements 48 subject to the same constraints applied to the encoded video discussed above with respect to FIG. 4a. However, each Block element contains an IDR. In other embodiments, the Cluster elements within the container files containing the trick play tracks need not correspond with the start time and duration of the Cluster elements in the container files containing the alternative video streams.

[0069] In many embodiments, source content can be encoded to provide a single trick play track or multiple trick play tracks for use by the adaptive bit rate streaming system. When a single trick play track is provided, the trick play track is typically encoded at a low bitrate. When multiple alternative trick play tracks are provided, adaptive rate streaming can also be performed with respect to the trick play tracks. In several embodiments, multiple trick play tracks are provided to support different rates of accelerated visual search through the encoded media.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0641
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

INCORPORATING SUBTITLE STREAMS AND FONT FILES IN MKV CONTAINERS

[0070] The exact representation of subtitles and fonts in any specific multimedia container standard may differ greatly. The Matroska container file format supports subtitles and has provisions for attaching font files and allowing pre-defined elements to specify the association of those attachments with the subtitle stream. Fonts are stored as attachments, and are explicitly associated with a specific subtitle TrackEntry by use of an AttachmentLink element in the TrackEntry that references the Attachment FileUID. The same identification mechanism (AttachmentLink) in a Track element may be repeated multiple times to associate multiple fonts with the Track containing the subtitle data. Additionally, an AttachmentStartTime field and AttachmentEndTime field may be added to the description of each individual attachment (AttachedFile) to denote the start and end times for which a font shall be utilized for the rendering of textual elements of a particular subtitle track. Some embodiments may associate a single font file for use by the movie, where AttachmentStartTime may be set to zero and AttachmentEndTime may be set to the time reflecting the entire duration of the movie. For transmission efficiency, a lossless compression scheme may be applied to font files prior to storage and/or transmission, and the font files can be decompressed by the playback device prior to utilization by a font-rendering engine.

[0071] Typically, subtitle font metadata and one or more associated font files are stored in a Matroska container file under an Attachment tag. The metadata associated with Font Files include (but are not limited to) FileDescription, FileName, FileMimeType, FileData and FileUID. The FileName tag identifies the font type stored within the attachment tag. For example, a FileName ARIALUNI.TTF, identifies the font type stored in the attachment as ARIAL. A parser can determine the FileName from the Attachment tag only after downloading the entire Attachment tag. Downloading attachment tags can be expensive, because the entire font information is present in it. An attachment tag for international language such as Chinese, Japanese and Hindi can be 2 MB and several seconds to downloading causing a significant delay. In the event that the playback device is already possesses the font, the delay can be unnecessary. In several embodiments, a non-standard Matroska file format is used to split the metadata and the font file into separate components. The metadata section can contain metadata used to identify the font file such as (but not limited to) FileDescription, FileMimeType, FileName, FileUID,

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0642
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

along with two additional non-standard tags FileLocation, and FileSize. The FileLocation tag can point to the location within the Matroska container file where the font file data is stored and the FileSize is the size of the font file data. By splitting the metadata and actual information, a streaming system can intelligently download the font data only if a particular font is unavailable in its system. A Matroska container file 32 including an Attachments element 54 including a separate AttachedFile element 56 containing metadata pointing to the location of font file data 57 (i.e. the File Location 58 and FileSize attributes 59) within the container file in accordance with an embodiment of the invention is illustrated in FIG. 3a.

[0072] As is noted above, the font file can be subsetted to reduce download time and to reduce the amount of memory occupied by the font file once downloaded. Processes for reducing the size of font files based upon the characters from the font utilized in a specific subtitle stream or segment of a subtitle stream are disclosed in U.S. Patent Application Serial No. 12/480,276 entitled "Systems and Methods for Font File Optimization for Multimedia Files", to Priyadarshi et al., filed June 8, 2009, which is incorporated by reference above. When a subtitle stream is segmented, the font file associated with that subtitle stream may also be segmented, such that only the characters present in the text for a particular subtitle segment are present in the corresponding font segment. When working with a segmented subtitle file, there will likely be an overlap of characters between the different segments. It is possible to scan all of the segments and determine all characters that overlap between those segments (union operation) and create a special font file that encompasses glyphs for the union of characters in all segments. Once a union font file has been created, subsequent segmented font files may be created as before, though in this case for characters in each segment that do not belong to the union of characters are represented in the segmented font. During playback, the union font file representing the union of characters is streamed first, followed by each subsequent difference or delta font file covering a specific time-range in the presentation, and corresponding to a specific subtitle segment. Start and end timecode indicators in the attachment headers for each font file (FIG. 3a) can be used to determine the time at which a font may be loaded, and the duration for which a file shall remain active. The union font, in this case, may remain active for the duration of the video and audio presentation, and other segments may remain active over the time period corresponding to a specific subtitle segment. The font renderer may be instructed to access

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0643
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

glyphs from the union font, together with the font file covering each segment of the subtitle text, according to the start and end timecode values. The top level index may create a special indicator for the union font file, such that the decoding device is made aware of the characteristics of this font file as opposed to the segment font files. When seeking to a section of the encoded video, which has not yet been played, only the new font file associated with the current range (if not already cached) is downloaded as the union font file is already cached.

[0073] It is also possible to segment the font files corresponding to each segment of the subtitle stream, such that for each segment, only the glyphs that are not included in the previously transmitted font file segments are included in the font file for the current segment. To render the subtitles, the font render is associated with all font files covering previous sections of the presentation. When seeking to a section of the movie which has not yet been played, all font files from the previous sections are downloaded first prior to playing back subtitles upon resuming from the seek. In this case, the start timecode value of the font files would coincide with the start of a subtitle segment, and the end timecode would coincide with the end of the video and audio presentation. This way, each font file would be used for the entire duration of the corresponding audio and video.

[0074] The Matroska standard does not have a specific way of associating other textual elements of the file (such as the movie title or track names) that are not part of the multimedia track hierarchy with a specific font. In one embodiment, with respect to the Matroska specification or format, the association of textual information of the file with an attached font file can be specified in the following manner:

```
+Tags
    +Tag
        +SimpleTag
            TagName=Font
            TagLanguage=jpn
            TagBinary=Subsetted Font
        +Tag
            +SimpleTag
                TagName=SUMMARY
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0644
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

TagLanguage=jpn
TagString="a lot of text"

[0075] In the above description, a subsetted font file is described by a hierarchy of a base Tag element associating a subsetted font with a series of textual elements, which use the existing Matroska Tag mechanism. In this scheme, the font file is described by the first SimpleTag element as shown above, and the actual binary data of the font can be encapsulated as a TagBinary field. The subsequent Tag elements following the first SimpleTag structure can be used to host all the textual elements related to this particular font description. A second Tag element appearing as the immediate child of the parent Tags may be used to host a second font file and associated textual elements, following the same hierarchical structure. Although specific examples are discussed above, any of a variety of techniques can be utilized to incorporate one or more font files within a container file format including (but not limited) to a container file format in accordance with embodiments of the invention that can be requested by a playback device as needed to render a specific subtitle stream.

INCORPORATING INDEXING INFORMATION WITHIN MKV CONTAINER FILES

[0076] The specification for the Matroska container file format provides for an optional Cues element that is used to index Block elements within the container file. A modified Cues element 52 that can be incorporated into a Matroska container file in accordance with an embodiment of the invention to facilitate the requesting of clusters by a playback device using HTTP is illustrated in FIG. 5. The modified Cues element 52 includes a plurality of CuePoint elements 70 that each include a CueTime attribute 72. Each CuePoint element includes a CueTrackPositions element 74 containing the CueTrack 76 and CueClusterPosition 78 attributes. In many embodiments, the CuePoint element is mainly configured to identify a specific Cluster element as opposed to a specific Block element within a Cluster element. Although, in several applications the ability to seek to specific BlockGroup elements within a Cluster element is required and additional index information is included in the Cues element.

[0077] The use of a modified Cues element to index encoded media within a Clusters element of a Matroska file in accordance with an embodiment of the invention is illustrated in

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0645
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

FIG. 6. A CuePoint element is generated to correspond to each Cluster element within the Matroska container file. The CueTime attribute 72 of the CuePoint element 70 corresponds to the Timecode attribute 60 of the corresponding Cluster element 48. In addition, the CuePoint element contains a CueTrackPositions element 74 having a CueClusterPosition attribute 78 that points to the start of the corresponding Cluster element 48. The CueTrackPositions element 74 can also include a CueBlockNumber attribute, which is typically used to indicate the Block element containing the first IDR frame within the Cluster element 48.

[0078] As can readily be appreciated the modified Cues element 52 forms an index to each of the Cluster elements 48 within the Matroska container file. Furthermore, the CueTrackPosition elements provide information that can be used by a playback device to request the byte range of a specific Cluster element 48 via HTTP or another suitable protocol from a remote server. The Cues element of a conventional Matroska file does not directly provide a playback device with information concerning the number of bytes to request from the start of the Cluster element in order to obtain all of the encoded video contained within the Cluster element. The size of a Cluster element can be inferred in a modified Cues element by using the CueClusterPosition attribute of the CueTrackPositions element that indexes the first byte of the next Cluster element. Alternatively, additional CueTrackPosition elements could be added to modified Cues elements in accordance with embodiments of the invention that index the last byte of the Cluster element (in addition to the CueTrackPositions elements that index the first byte of the Cluster element), and/or a non-standard CueClusterSize attribute that specifies the size of the Cluster element pointed to by the CueClusterPosition attribute is included in each CueTrackPosition element to assist with the retrieval of specific Cluster elements within a Matroska container file via HTTP byte range requests or a similar protocol.

[0079] The modification of the Cues element in the manner outlined above significantly simplifies the retrieval of Cluster elements from a Matroska container file via HTTP or a similar protocol during adaptive bitrate streaming. In addition, by only indexing the first frame in each Cluster the size of the index is significantly reduced. Given that the index is typically downloaded prior to playback, the reduced size of the Cues element (i.e. index) means that playback can commence more rapidly. Using the CueClusterPosition elements, a playback device can request a specific Cluster element from the stream most suited to the streaming

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0646
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

conditions experienced by the playback device by simply referencing the index of the relevant Matroska container file using the Timecode attribute for the desired Cluster element.

[0080] In some embodiments, a number of the attributes within the Cues element are not utilized during adaptive bitrate streaming. Therefore, the Cues element can be further modified by removing the unutilized attributes to reduce the overall size of the index for each Matroska container file. A modified Cues element that can be utilized in a Matroska container file that includes a single encoded stream in accordance with an embodiment of the invention is illustrated in FIG. 5a. The Cues element 52' shown in FIG. 5a is similar to the Cues element 52 shown in FIG. 5 with the exception that the CuePoint elements 70' do not include a CueTime attribute (see 72 in FIG. 5) and/or the CueTrackPositions elements 74' do not include a CueTrack attribute (76 in FIG. 5). When the portions of encoded media in each Cluster element in the Motroska container file have the same duration, the CueTime attribute is not necessary. When the Matroska contain file includes a single encoded stream, the CueTrack attribute is not necessary. In other embodiments, the Cues element and/or other elements of the Matroska container file can be modified to remove elements and/or attributes that are not necessary for the adaptive bitrate streaming of the encoded stream contained within the Matroska container file, given the manner in which the stream is encoded and inserted in the Matroska container file.

[0081] Although various modifications to the Cues element to include information concerning the size of each of the Cluster elements within a Matroska container file and to eliminate unnecessary attributes are described above, many embodiments of the invention utilize a conventional Matroska container. In several embodiments, the playback device simply determines the size of Cluster elements on the fly using information obtained from a conventional Cues element, and/or relies upon a separate index file containing information concerning the size and/or location of the Cluster elements within the MKV container file. In several embodiments, the additional index information is stored in the top level index file. In a number of embodiments, the additional index information is stored in separate files that are identified in the top level index file. When index information utilized to retrieve Cluster elements from a Matroska container file is stored separately from the container file, the Matroska container file is still typically constrained to encode media for inclusion in the Cluster elements in the manner outlined above. In addition, wherever the index information is located, the index

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0647
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

information will typically index each Cluster element and include (but not be limited to) information concerning at least the starting location and, in many instances, the size of each Cluster element.

ENCODING SOURCE MEDIA FOR ADAPTIVE BITRATE STREAMING

[0082] A process for encoding source media as a top level index file and a plurality of Matroska container files for use in an adaptive bitrate streaming system in accordance with an embodiment of the invention is illustrated in FIG. 7. The encoding process 100 commences by selecting (102) a first portion of the source media and encoding (104) the source media using the encoding parameters for each stream. When the portion of media is video, then the portion of source video is encoded as a single GOP commencing with an IDR frame. In many embodiments, encoding parameters used to create the alternative GOPs vary based upon bitrate, frame rate, encoding parameters and resolution. In this way, the portion of media is encoded as a set of interchangeable alternatives and a playback device can select the alternative most appropriate to the streaming conditions experienced by the playback device. When different resolutions are supported, the encoding of the streams is constrained so that each stream has the same display aspect ratio. A constant display aspect ratio can be achieved across different resolution streams by varying the sample aspect ratio with the resolution of the stream. In many instances, reducing resolution can result in higher quality video compared with higher resolution video encoded at the same bit rate. In many embodiments, the source media is itself encoded and the encoding process (104) involves transcoding or transrating of the encoded source media according to the encoding parameters of each of the alternative streams supported by the adaptive bitrate streaming system.

[0083] Once the source media has been encoded as a set of alternative portions of encoded media and one or more subtitle streams, each of the alternative portions of encoded media is inserted (106) into a Cluster element within the Matroska container file corresponding to the stream to which the portion of encoded media belongs. In many embodiments, the encoding process also constructs indexes for each Matroska container file as media is inserted into Cluster elements within the container. Therefore, the process 100 can also include creating a CuePoint element that points to the Cluster element inserted within the Matroska container file. The

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0648
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

CuePoint element can be held in a buffer until the source media is completely encoded. Although the above process describes encoding each of the alternative portions of encoded media sequentially in a single pass through the source media, many embodiments of the invention involve performing a separate pass through the source media to encode each of the alternative streams.

[0084] Referring back to FIG. 7, the process continues to select (102) and encode (104) portions of the source media and then insert (106) the encoded portions of media into the Matroska container file corresponding to the appropriate stream until the entire source media is encoded for adaptive bitrate streaming (108). At which point, the process can insert an index (110) into the Matroska container for each stream and create (112) a top level index file that indexes each of the encoded streams contained within the Matroska container files. As noted above, the indexes can be created as encoded media and inserted into the Matroska container files so that a CuePoint element indexes each Cluster element within the Mastroska container file. Upon completion of the encoding, each of the CuePoint elements can be included in a Cues element and the Cues element can be inserted into the Matroska container file following the Clusters element. When the source media includes subtitles, the process also includes inserting (114) one or more font files constructed in the manner outlined above into the container file as an attachment.

[0085] Following the encoding of the source media to create Matroska container files containing each of the streams generated during the encoding process, which can include the generation of trick play streams, and a top level index file that indexes each of the streams within the Matroska container files, the top level index file and the Matroska container files can be uploaded to an HTTP server for adaptive bitrate streaming to playback devices. The adaptive bitrate streaming of media including subtitles encoded in accordance with embodiments of the invention using HTTP requests is discussed further below.

ADAPTIVE BITRATE STREAMING FROM MKV CONTAINER FILES USING HTTP

[0086] When source media is encoded so that there are alternative streams contained in separate Matroska container files for at least one of video, audio, and subtitle content, adaptive streaming of the media contained within the Matroska container files can be achieved using

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0649
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

HTTP requests or a similar stateless data transfer protocol. In many embodiments, a playback device requests the top level index file resident on the server and uses the index information to identify the streams that are available to the playback device. When the playback device receives an instruction to stream subtitles in conjunction with the adaptive bitrate streaming of encoded video, the playback device can retrieve information concerning the font utilized by the subtitle stream from within the container file containing the subtitle stream. In the event that the subtitle is not resident on the playback device, the playback device can request one or more font files associated with the subtitle stream. In several embodiments, the entire subtitle stream and the corresponding font file(s) can be downloaded prior to playback of the encoded video content. In many embodiments, the subtitle stream is segmented and/or the font file(s) are subsetted according to the segment of the subtitle stream with which the font file is associated. In this way, the amount of information that is downloaded by the playback device prior to the presentation of the video and accompanying subtitle stream can be reduced.

[0087] The playback device can retrieve the indexes for one or more of the Matroska files and can use the indexes to request media from one or more of the streams contained within the Matroska container files using HTTP requests or using a similar stateless protocol. As noted above, many embodiments of the invention implement the indexes for each of the Matroska container files using a modified Cues element. In a number of embodiments, however, the encoded media for each stream is contained within a standard Matroska container file and separate index file(s) can also be provided for each of the container files. Based upon the streaming conditions experienced by the playback device, the playback device can select media from alternative streams encoded at different bitrates. When the media from each of the streams is inserted into the Matroska container file in the manner outlined above, transitions between streams can occur upon the completion of playback of media within a Cluster element. Therefore, the size of the Cluster elements (i.e the duration of the encoded media within the Cluster elements) is typically chosen so that the playback device is able to respond quickly enough to changing streaming conditions and to instructions from the user that involve utilization of a trick play track. The smaller the Cluster elements (i.e. the smaller the duration of the encoded media within each Cluster element), the higher the overhead associated with requesting each Cluster element. Therefore, a tradeoff exists between the responsiveness of the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0650
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

playback device to changes in streaming conditions and the effective data rate of the adaptive streaming system for a given set of streaming conditions (i.e. the portion of the available bandwidth actually utilized to transmit encoded media). In many embodiments, the size of the Cluster elements is chosen so that each Cluster element contains two seconds of encoded media. In other embodiments, the duration of the encoded media can be greater or less than two seconds and/or the duration of the encoded media can vary from Cluster element to Cluster element.

[0088] Communication between a playback device or client and an HTTP server during the playback of media encoded in separate streams contained within Matroska container files indexed by a top level index file in accordance with an embodiment of the invention is illustrated in FIG. 8. In the illustrated embodiment, the playback device 200 commences playback by requesting the top level index file from the server 202 using an HTTP request or a similar protocol for retrieving data. The server 202 provides the bytes corresponding to the request. The playback device 200 then parses the top level index file to identify the URIs of each of the Matroska container files containing the streams of encoded media derived from a specific piece of source media. The playback device can then request the byte ranges corresponding to headers of one or more of the Matroska container files via HTTP or a similar protocol, where the byte ranges are determined using the information contained in the URI for the relevant Matroska container files (see discussion above). The server returns the following information in response to a request for the byte range containing the headers of a Matroska container file:

ELEM("EBML")
ELEM("SEEKHEAD")
ELEM("SEGMENTINFO")
ELEM("TRACKS")

[0089] The EBML element is typically processed by the playback device to ensure that the file version is supported. The SeekHead element is parsed to find the location of the Matroska index elements and the SegmentInfo element contains two key elements utilized in playback: TimecodeScale and Duration. The TimecodeScale specifies the timecode scale for all timecodes within the Segment of the Matroska container file and the Duration specifies the duration of the Segment based upon the TimecodeScale. The Tracks element contains the information used by

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0651
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the playback device to decode the encoded media contained within the Clusters element of the Matroska file. When the Matroska container file includes a subtitle stream, the Tracks element can reference one or more font file attachments to the Matroska container file that can be utilized by the font-rendering engine of the playback device. The font files need not be downloaded by the playback device until the specific subtitle stream contained within the container file is requested for playback. At which point the font file(s) downloaded typically depend upon information within the container file relating the font file(s) to the timing of the presentation of the subtitles. In many embodiments, a single font file is provided and the playback device downloads the font file. Where multiple font files are present the playback device downloads the font files associated with the segment of the subtitle stream being requested by the playback device. In a number of embodiments, the playback device downloads a font file associated with the segment of the subtitle stream requested by the playback device. In many embodiments, an initial or union font file is downloaded and any additional font file associated with the segment of the subtitle stream requested by the playback device is also downloaded. In several embodiments, a series of font files is downloaded based upon the timing of the segment of the subtitle stream requested by the playback device relative to the start of the subtitle stream (e.g. every font file associated with the subtitle stream from the start of the subtitle stream up to the requested segment of the playback stream are requested). In a number of embodiments, the playback device presents a display to the user indicating the estimated time remaining to download the font file(s). The user can interrupt the download to commence streaming of the audio and/or video. In the event that the user interrupts downloading of the font file a message can be displayed indicating that the requested subtitle stream will not be available during playback. As noted above, adaptive bitrate streaming systems in accordance with embodiments of the invention can support different streams encoded using different encoding parameters including but not limited to frame rate, and resolution. Therefore, the playback device can use the information contained within the Matroska container file's headers to configure the decoder every time a transition is made between encoded streams.

[0090] In many embodiments, the playback device does not retrieve the headers for all of the Matroska container files indexed in the top level index file. Instead, the playback device determines the stream(s) that will be utilized to initially commence playback and requests the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0652
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

headers from the corresponding Matroska container files. Depending upon the structure of the URIs contained within the top level index file, the playback device can either use information from the URIs or information from the headers of the Matroska container files to request byte ranges from the server that contain at least a portion of the index from relevant Matroska container files. The byte ranges can correspond to the entire index. The server provides the relevant byte ranges containing the index information to the playback device, and the playback device can use the index information to request the byte ranges of Cluster elements containing encoded media using this information. When the Cluster elements are received, the playback device can extract encoded media from the Block elements within the Cluster element, and can decode and playback the media within the Block elements in accordance with their associated Timecode attributes.

[0091] In the illustrated embodiment, the playback device 200 requests sufficient index information from the HTTP server prior to the commencement of playback that the playback device can stream the entirety of each of the selected streams using the index information. In other embodiments, the playback device continuously retrieves index information as media is played back. In several embodiments, all of the index information for the lowest bitrate steam is requested prior to playback so that the index information for the lowest bitrate stream is available to the playback device in the event that streaming conditions deteriorate rapidly during playback.

SWITCHING BETWEEN STREAMS

[0092] The communications illustrated in FIG. 8 assume that the playback device continues to request media from the same streams (i.e. Matroska container files) throughout playback of the media. In reality, the streaming conditions experienced by the playback device are likely to change during the playback of the streaming media and the playback device can request media from alternative streams (i.e. different Matroska container files) to provide the best picture quality for the streaming conditions experienced by the playback device. In addition, the playback device may switch streams in order to perform a trick play function that utilizes a trick play track stream.

[0093] Communication between a playback device and a server when a playback device switches to a new stream in accordance with embodiments of the invention are illustrated in FIG.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0653
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

9a. The communications illustrated in FIG. 9a assume that the index information for the new stream has not been previously requested by the playback device and that downloading of Cluster elements from the old stream proceeds while information is obtained concerning the Matroska container file containing the new stream. When the playback device 200 detects a change in streaming conditions, determines that a higher bitrate stream can be utilized at the present streaming conditions, or receives a trick play instruction from a user, the playback device can use the top level index file to identify the URI for a more appropriate alternative stream to at least one of the video, audio, or subtitle streams from which the playback device is currently requesting encoded media. The playback device can save the information concerning the current stream(s) and can request the byte ranges of the headers for the Matroska container file(s) containing the new stream(s) using the parameters of the corresponding URIs. Caching the information in this way can be beneficial when the playback device attempts to adapt the bitrate of the stream downward. When the playback device experiences a reduction in available bandwidth, the playback device ideally will quickly switch to a lower bitrate stream. Due to the reduced bandwidth experienced by the playback device, the playback device is unlikely to have additional bandwidth to request header and index information. Ideally, the playback device utilizes all available bandwidth to download already requested higher rate Cluster elements and uses locally cached index information to start requesting Cluster elements from Matroska container file(s) containing lower bitrate stream(s).

[0094] Byte ranges for index information for the Matroska container file(s) containing the new stream(s) can be requested from the HTTP server 202 in a manner similar to that outlined above with respect to FIG. 8. At which point, the playback device can stop downloading of cluster elements from the previous streams and can commence requesting the byte ranges of the appropriate Cluster elements from the Matroska container file(s) containing the new stream(s) from the HTTP server, using the index information from the Matroska container file(s) to identify the Cluster element(s) containing the encoded media following the encoded media in the last Cluster element retrieved by the playback device. As noted above, the smooth transition from one stream to another is facilitated by encoding each of the alternative streams so that corresponding Cluster elements start with the same Timecode element and an IDR frame.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0654
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0095] When the playback device caches the header and the entire index for each stream that has be utilized in the playback of the media, the process of switching back to a previously used stream can be simplified. The playback device already has the header and index information for the Matroska file containing the previously utilized stream and the playback device can simply use this information to start requesting Cluster elements from the Matroska container file of the previously utilized stream via HTTP. Communication between a playback device and an HTTP server when switching back to a stream(s) for which the playback device has cached header and index information in accordance with an embodiment of the invention is illustrated in FIG. 9b. The process illustrated in FIG. 9b is ideally performed when adapting bitrate downwards, because a reduction in available resources can be exacerbated by a need to download index information in addition to media. The likelihood of interruption to playback is reduced by increasing the speed with which the playback device can switch between streams and reducing the amount of overhead data downloaded to achieve the switch.

[0096] Although the present invention has been described in certain specific aspects, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that the present invention may be practiced otherwise than specifically described, including various changes in the implementation such as utilizing encoders and decoders that support features beyond those specified within a particular standard with which they comply, without departing from the scope and spirit of the present invention. Thus, embodiments of the present invention should be considered in all respects as illustrative and not restrictive.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0655
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS CLAIMED:

1. A method of playing back encoded media, where the media is encoded as a plurality of alternative streams and at least one subtitle stream, the method comprising:

requesting and buffering portions of video from at least one of the alternative streams using a playback device;

requesting information indicative of a font utilized by a font-rendering engine to render text from a selected subtitle stream;

downloading at least one font file when the font is not present on the playback device;

requesting and buffering at least a portion of the selected subtitle stream;

decoding the buffered portions of video using a decoder on the playback device;

rendering the portions of the subtitle stream corresponding to the buffered portion of video using a font-rendering engine configured by the at least one downloaded font file; and

performing synchronized playback of the decoded video and rendered subtitles using the playback device.

2. The method of claim 1, further comprising measuring the current streaming conditions by measuring the time taken to receive requested portions of a stream from the time at which the portions were requested.

3. The method of claim 2, wherein requesting and buffering portions of video from at least one of the alternative streams using a playback device further comprises requesting and buffering portions of video from at least one of the alternative streams based upon the bitrates of the alternative streams and the measured streaming conditions using a playback device.

4. The method of claim 1, wherein the selected subtitle stream is stored within a container file that includes the font file as an attachment to the container file.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0656
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5. The method of claim 4, wherein the container file in which the selected subtitle stream is stored includes a separate element containing metadata describing the attached font file.

6. The method of claim 5, further comprising determining whether the font file is present on the playback device without downloading the font file by requesting the metadata describing the attached font file from the container file using the playback device.

7. The method of claim 5, wherein the metadata describing the attached font file includes a description of the font file and the location of the font file within the container file.

8. The method of claim 1, wherein the font file is compressed and the method further comprises decompressing the downloaded font file using the playback device.

9. The method of claim 1, wherein the selected subtitle stream is selected by the playback device in response to a user instruction.

10. The method of claim 1, further comprising displaying a user interface indicative of the estimated time to download the at least one font file using the playback device.

11. The method of claim 1, wherein the subtitle stream is encoded as Unicode text.

12. The method of claim 1, wherein requesting and buffering at least a portion of the selected subtitle stream comprises downloading and buffering the entire subtitle stream prior to the commencement of playback

13. The method of claim 1, wherein requesting and buffering at least a portion of the selected subtitle stream comprises requesting and buffering segments of the subtitle stream with timing corresponding to the timing of the portions of video requested by the playback device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0657
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

14. The method of claim 13, wherein the font used to render the subtitle track is contained within a plurality of font files and at least one of the plurality of font files corresponds to a segment of the subtitle stream, the method further comprising downloading the font file corresponding to a segment of the subtitle stream when the playback device requests the segment of the subtitle stream.

15. The method of claim 14, wherein the font used to render the subtitle track is active according to an associated start timecode and end timecode, and the start and end timecodes are independent of the subtitle segment times.

16. The method of claim 15, wherein start and end timecodes for a plurality of font segments overlap, and lead to multiple font segments being utilized for rendering subtitles at a given time.

17. The method of claim 1, wherein:

the alternative streams and the subtitle stream are stored in separate container files; and

requesting portions of a stream further comprising requesting portions of files from remote servers via Hypertext Transfer Protocol (HTTP) byte range requests using the playback device.

18. The method of claim 1, wherein the alternative streams and the subtitle stream are stored in separate Extensible Binary Markup Language (EBML) container files.

19. The method of claim 18, wherein each of the EBML container files comprises a plurality of Cluster elements, where each Cluster element contains a portion of encoded media.

20. The method of claim 19, wherein the portions of encoded media in each of the Cluster elements have the same duration.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0658
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

21. The method of claim 20, wherein the portions of encoded media in each of the Cluster elements have a 2 second duration.

22. The method of claim 18, wherein the EBML container file containing the selected subtitle stream includes at least one font file as an attachment to the container file.

23. The method of claim 22, wherein the at least one font file attached to the container file are compressed.

24. The method of claim 1, further comprising retrieving a top level index file using the playback device that identifies the alternative streams and identifies at least one subtitle stream.

25. A playback device configured to playback encoded media, where the media is encoded as a plurality of alternative streams and at least one subtitle stream, the playback device comprising:

a processor configured by a client application to request portions of files from a remote server;

wherein the client application further configures the processor to:

request and buffer portions of video from at least one of the alternative streams;

request information indicative of a font utilized by a font-rendering engine to render text from a selected subtitle stream;

download at least one font file when the font is not present on the playback device;

request and buffer at least a portion of the selected subtitle stream;

decode the buffered portions of video;

render the portions of the subtitle stream corresponding to the buffered portion of video using a font-rendering engine configured by the at least one downloaded font file; and

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0659
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

perform synchronized playback of the decoded video and rendered subtitles.

26. A method of encoding source media including at least one subtitle track as a plurality of alternative video streams and at least one subtitle stream using a source encoder, the method comprising:

selecting a portion of the source media using the source encoder;

transcoding the selected portion of the source media into a plurality of alternative portions of encoded video using the source encoder, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame at the start of a closed group of pictures (GOP);

writing each of the alternative portions of encoded video to a separate container file using the source encoder;

writing a subtitle stream segment from each subtitle track corresponding to the selected portion of the source media to a separate container file using the source encoder; and

associating a font file with at least one of the container files containing subtitle stream segments.

27. The method of claim 26, further comprising compressing the font file.

28. The method of claim 26, further comprising reducing the size of the font file by subsetting the font file, where the subsetted font file includes all of the characters within the subtitle stream segments within the container file with which the font file is associated.

29. The method of claim 26, further comprising breaking the font file into a plurality of font files, where at least one of the font files includes characters displayed during a subtitle stream segment.

30. The method of claim 29, wherein breaking the font file into a plurality of font files comprises generating font files corresponding to different subtitle stream segments.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0660
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

31. The method of claim 30, wherein the plurality of font files is assembled as attachments to a single container file.

32. The method of claim 30, wherein starting and ending timecodes are used to indicate a time period over which each font file is used for rendering subtitles.

33. The method of claim 30, further comprising generating a union font file using the plurality of font files that includes characters that appear in at least two of the plurality of font files and subsetting the plurality of font files to remove the characters that are in the union font file.

34. The method of claim 30, further comprising subsetting each font file so that characters used in a previous subtitle stream segments are removed from the font file.

35. The method of claim 26, wherein the selected portions of source media are selected based upon their duration.

36. The method of claim 35, wherein the selected portions of source media each have a duration of two seconds.

37. The method of claim 26, wherein the container files containing the alternative streams and the subtitle streams are Extensible Binary Markup Language (EBML) container files.

38. The method of claim 37, wherein:

the element of the EBML container file to which each alternative portion of encoded video is written is a Cluster element; and

each Cluster element to which each of the alternative portions of encoded video is written is assigned the same time code.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0661
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

39. The method of claim 37, wherein the element of the EBML container file to which the subtitle stream segment is written is a Cluster element containing the same time code as the Cluster elements to which the alternative portions of encoded video are written.

40. The method of claim 26, further comprising creating a top level index file that identifies each of the container files containing the alternative portions of encoded video and identifies the container files that include the at least one subtitle stream.

41. The method of claim 26, wherein the font file associated with the at least one of the container files containing subtitle stream segments is attached to the container file and the container file includes a separate element containing metadata describing the font file.

42. The method of claim 41, wherein the metadata includes the location of the font file within the container file and information describing the file.

43. A source encoder configured to encode source media including at least one subtitle track as a plurality of alternative video streams and at least one subtitle stream, the source encoder comprising:

a processor configured by a source encoding application to select a portion of the source media;

wherein the source encoding application further configures the processor to:

transcode the selected portion of the source media into a plurality of alternative portions of encoded video, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame at the start of a closed group of pictures (GOP);

write each of the alternative portions of encoded video to a separate container file using the source encoder;

write a subtitle stream segment from each subtitle track corresponding to the selected portion of the source media to a separate container file using the source encoder; and

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0662
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

associate a font file with at least one of the container files containing subtitle stream segments.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0663
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

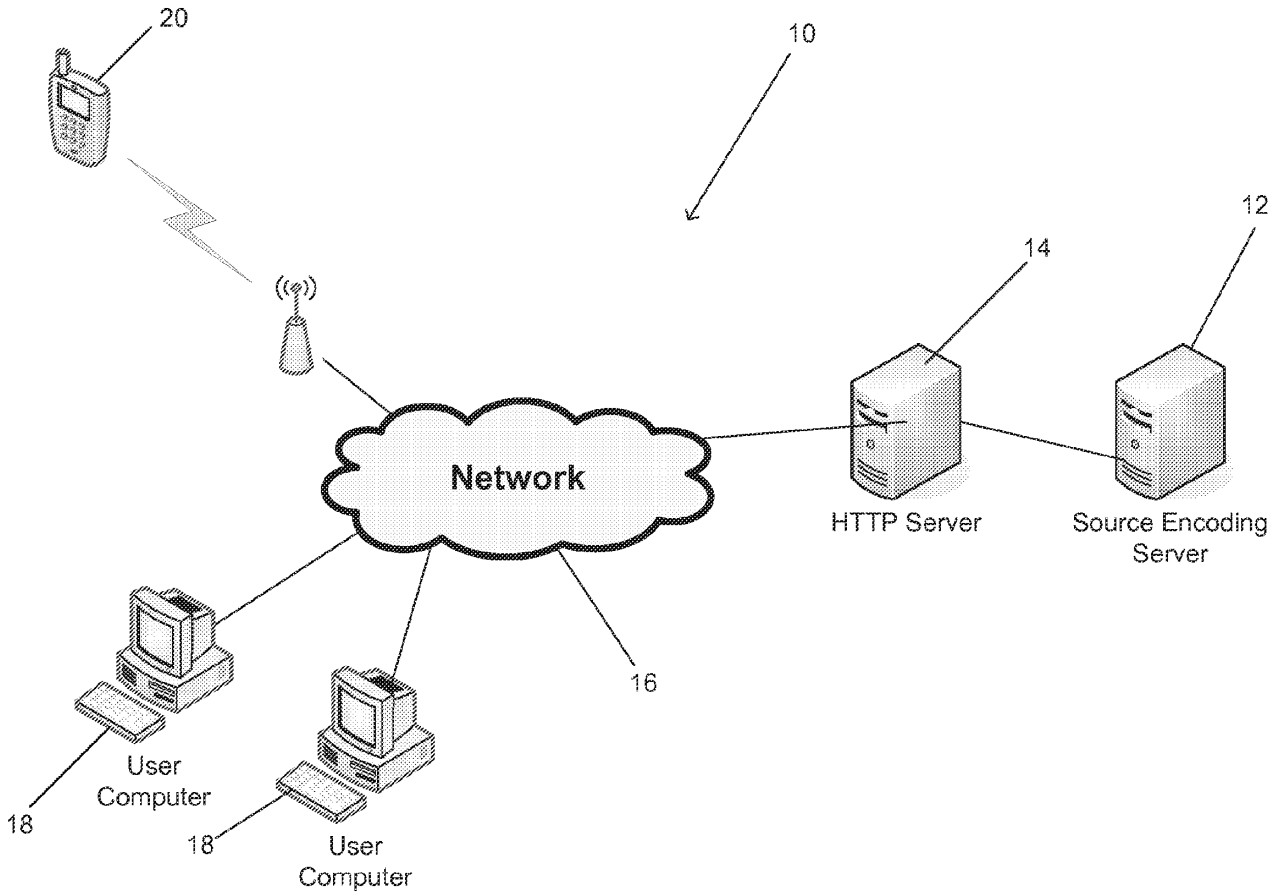


FIG. 1

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0664
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

SMIL File
30
MKVFile (Stream 1)
32
MKVFile (Stream 2)
32
MKVFile (Stream 3)
32
MKVFile (Trick Play)
32

FIG. 2

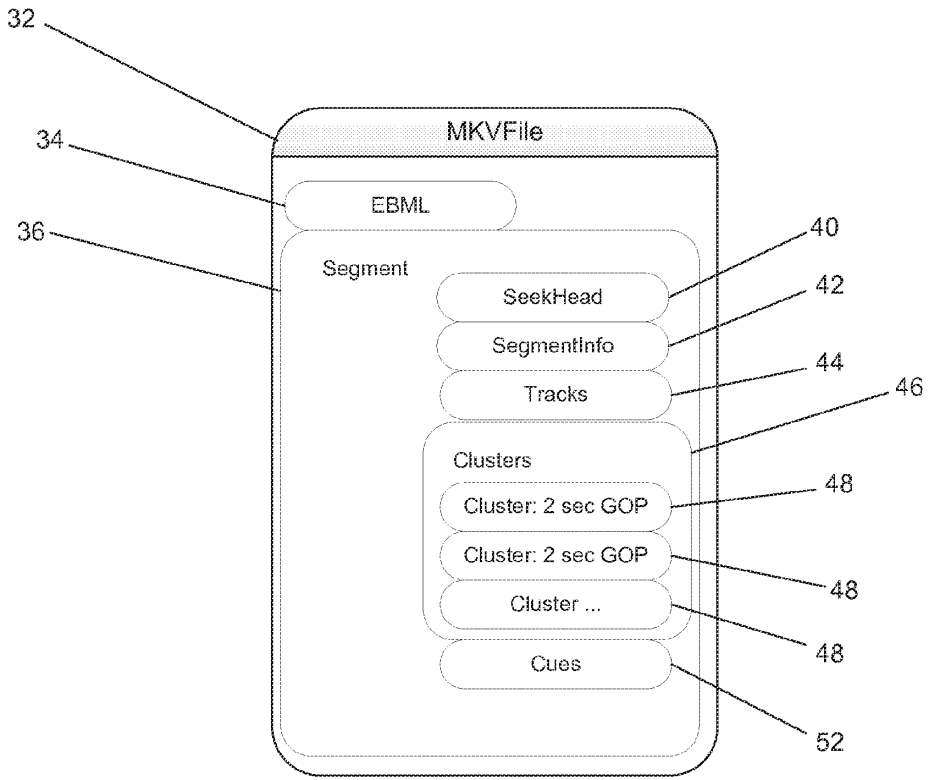

32
34
36
MKVFile
EBML
Segment
SeekHead
SegmentInfo
Tracks
Clusters
Cluster: 2 sec GOP
Cluster: 2 sec GOP
Cluster ...
Cues
40
42
44
46
48
48
48
52

FIG. 3

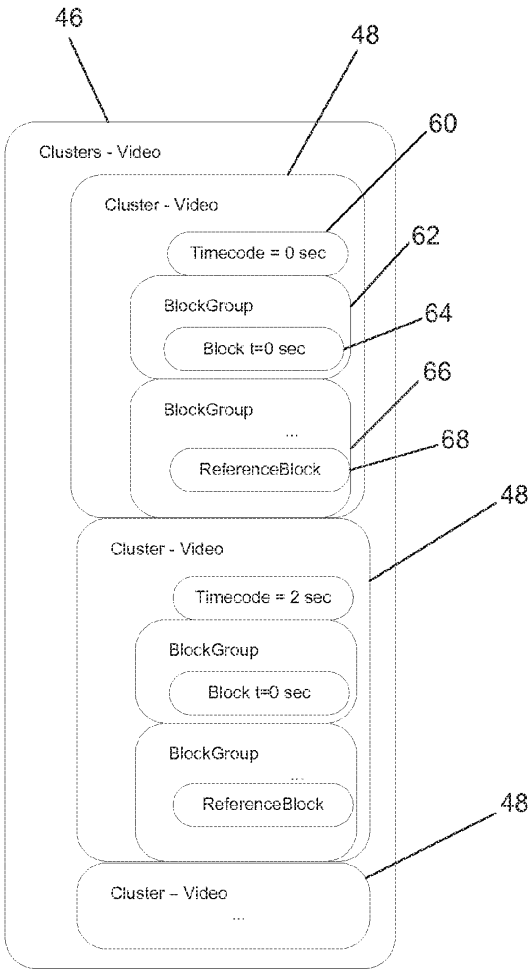


FIG. 4a

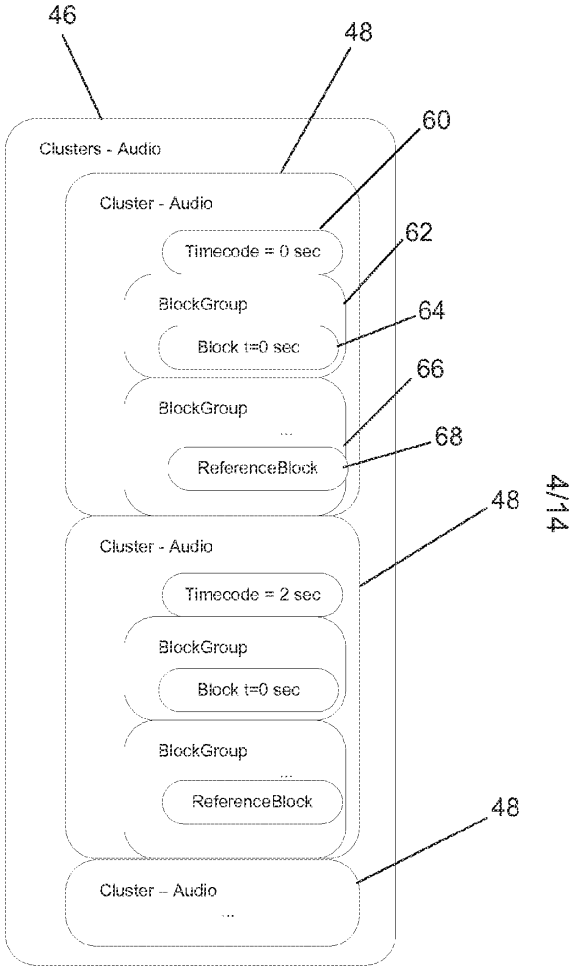


FIG. 4b

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0667
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

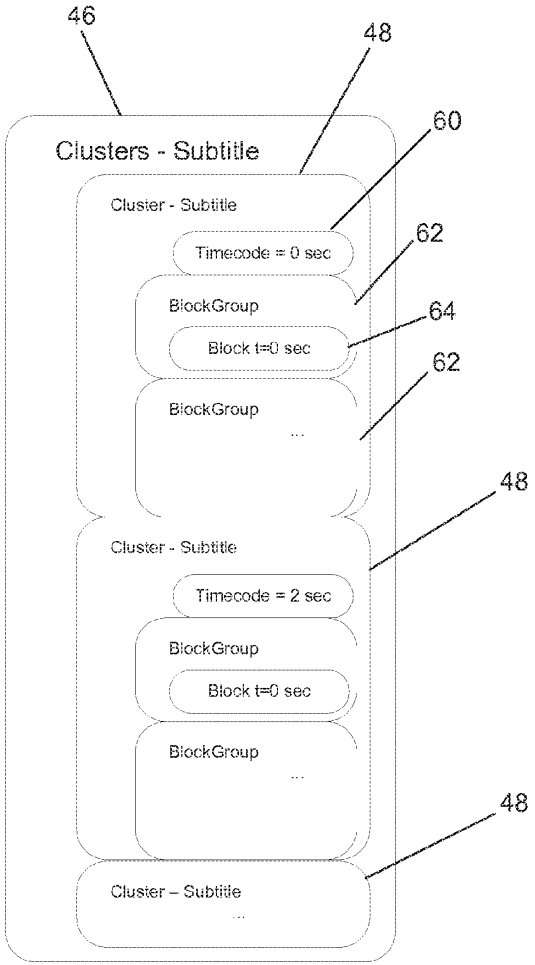


FIG. 4c

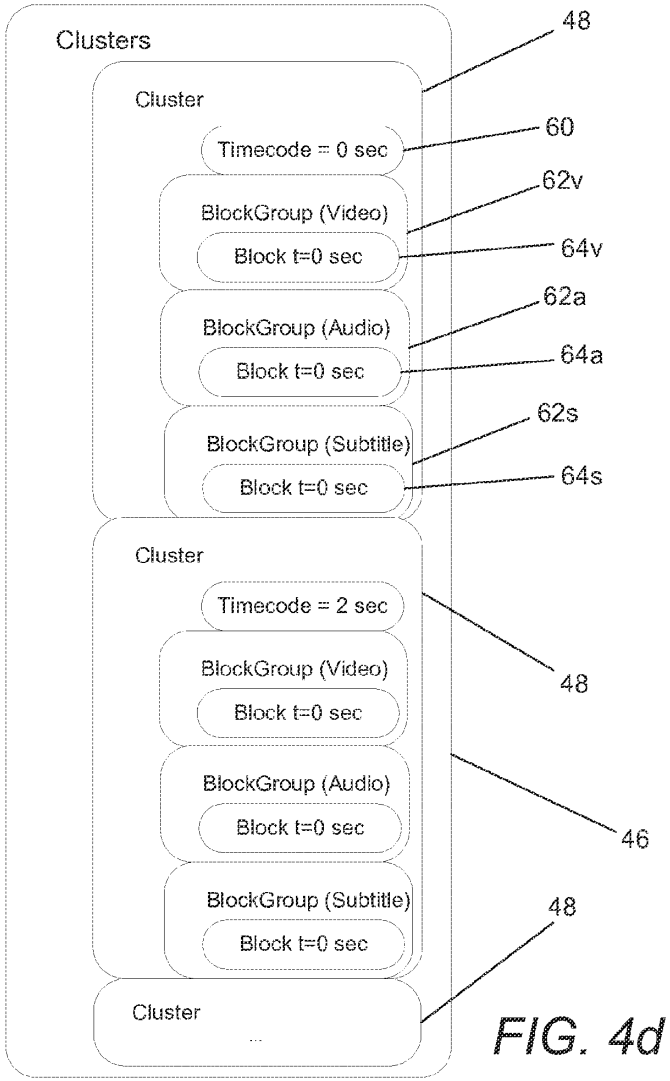


FIG. 4d

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0668
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

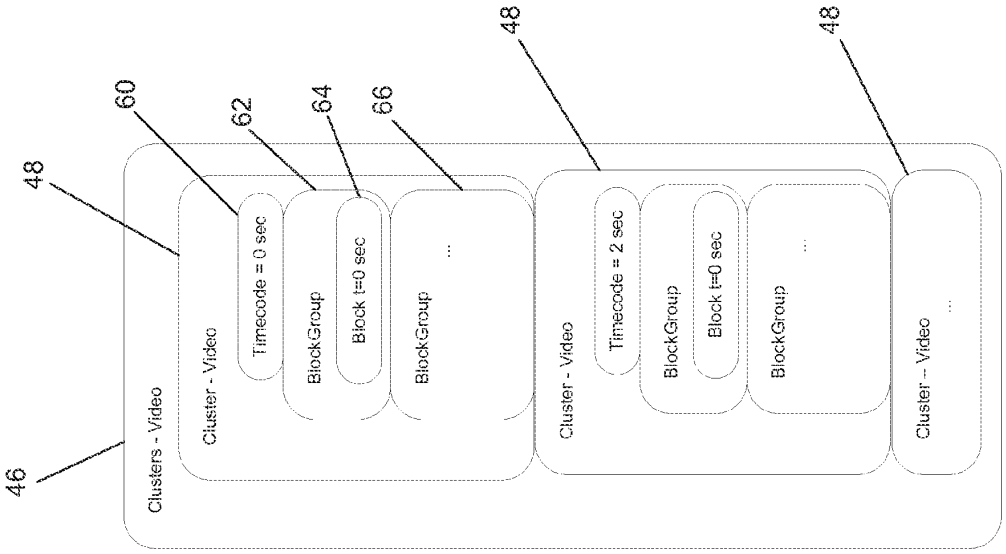

46
48
60
62
64
66
48
48
Clusters - Video
Cluster - Video
Timecode = 0 sec
BlockGroup
Block t=0 sec
BlockGroup
...
Cluster - Video
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
...
Cluster - Video
...

FIG. 4e

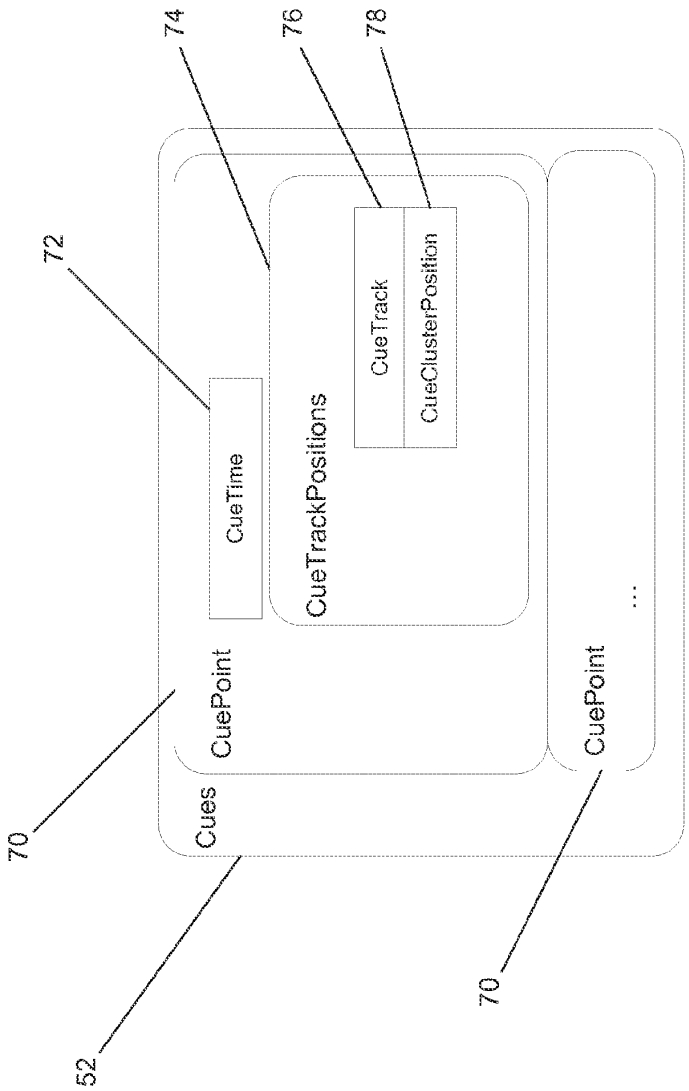

74
76
78
72
CueTime
CueTrackPositions
CueTrack
CueClusterPosition
CuePoint
...
CuePoint
Cues
70
52
70

FIG. 5

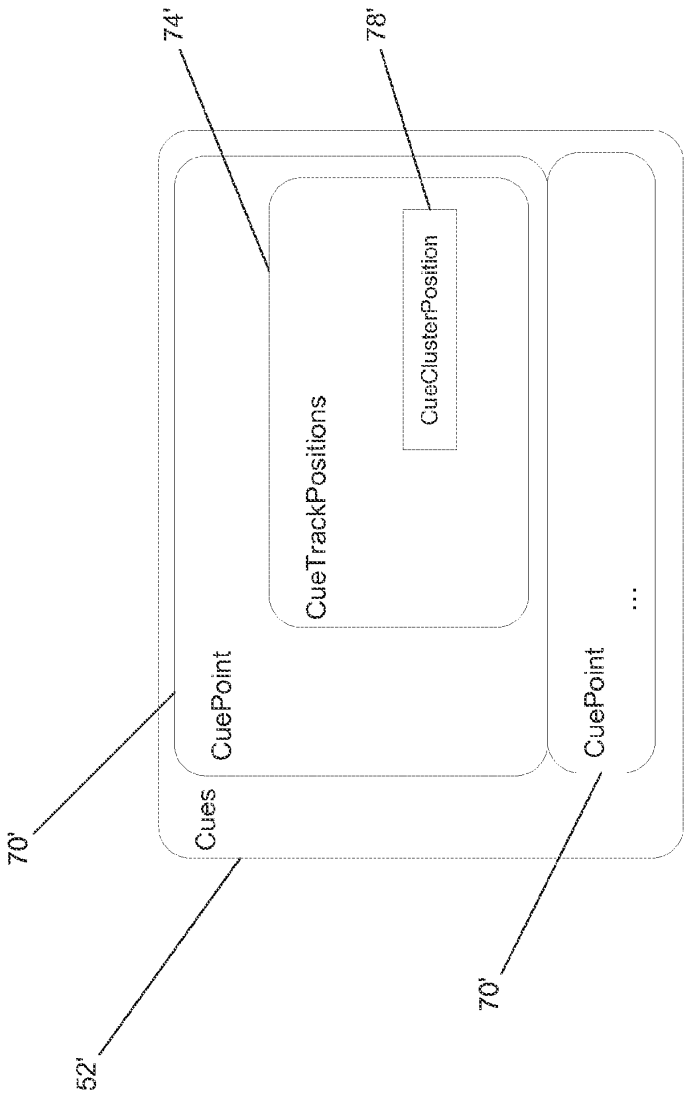

74'
78'
CueClusterPosition
CueTrackPositions
CuePoint
...
CuePoint
Cues
70'
52'
70'

FIG. 5a

70"
72"
52"
74"
Cues
CuePoint
CueTime
CueTrackPositions
78"
CueClusterPosition
79"
CueBlockPositions
CueBlockPositions
80"
CueBlockPosition
...
70"
CuePoint
...

FIG. 5b

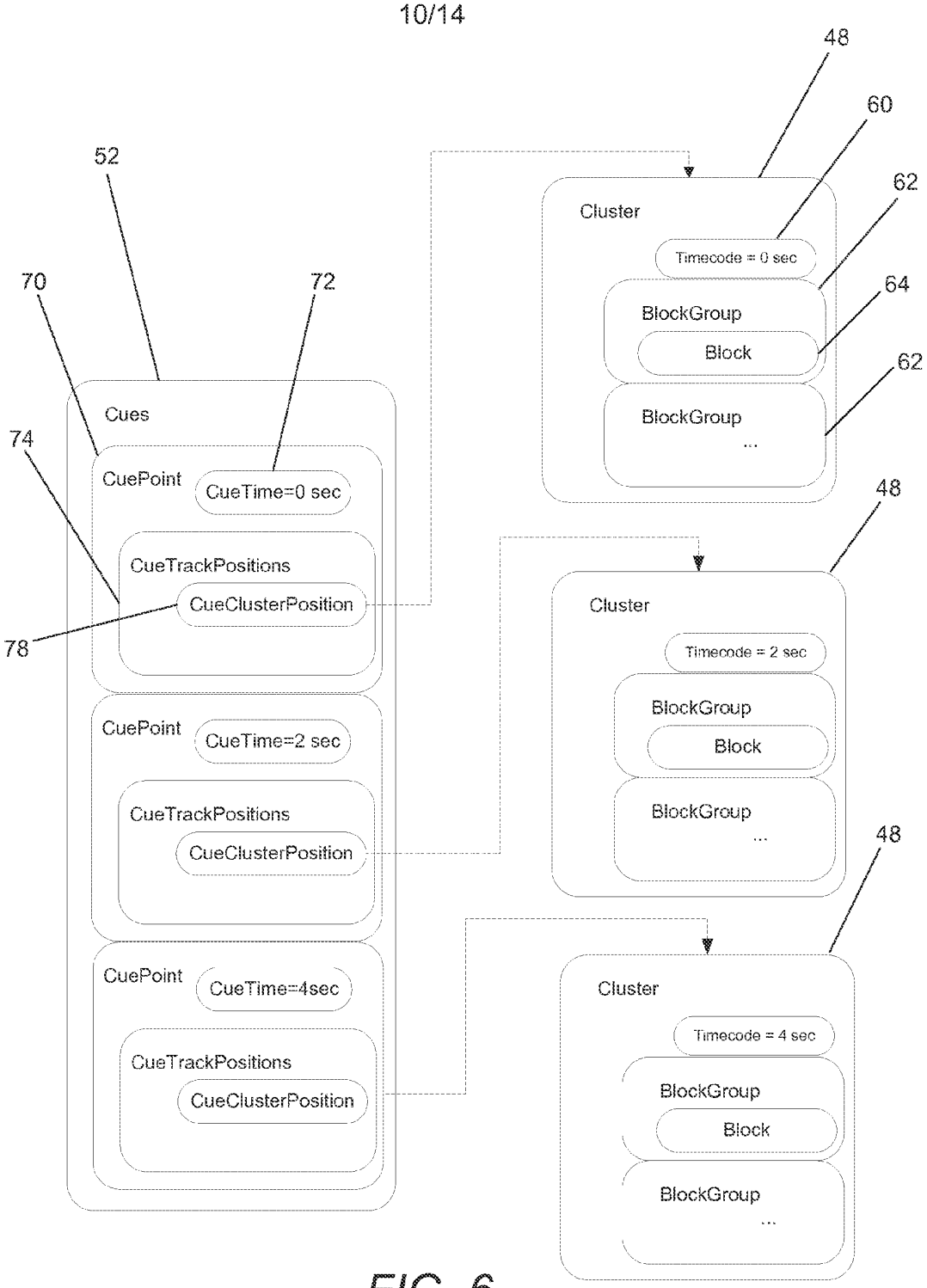

48
60
52
62
Cluster
Timecode = 0 sec
70
72
64
BlockGroup
Block
62
BlockGroup
...
Cues
74
CuePoint
CueTime=0 sec
48
CueTrackPositions
CueClusterPosition
Cluster
78
Timecode = 2 sec
CuePoint
CueTime=2 sec
BlockGroup
Block
CueTrackPositions
BlockGroup
CueClusterPosition
...
48
CuePoint
CueTime=4sec
Cluster
CueTrackPositions
Timecode = 4 sec
CueClusterPosition
BlockGroup
Block
BlockGroup
...

FIG. 6

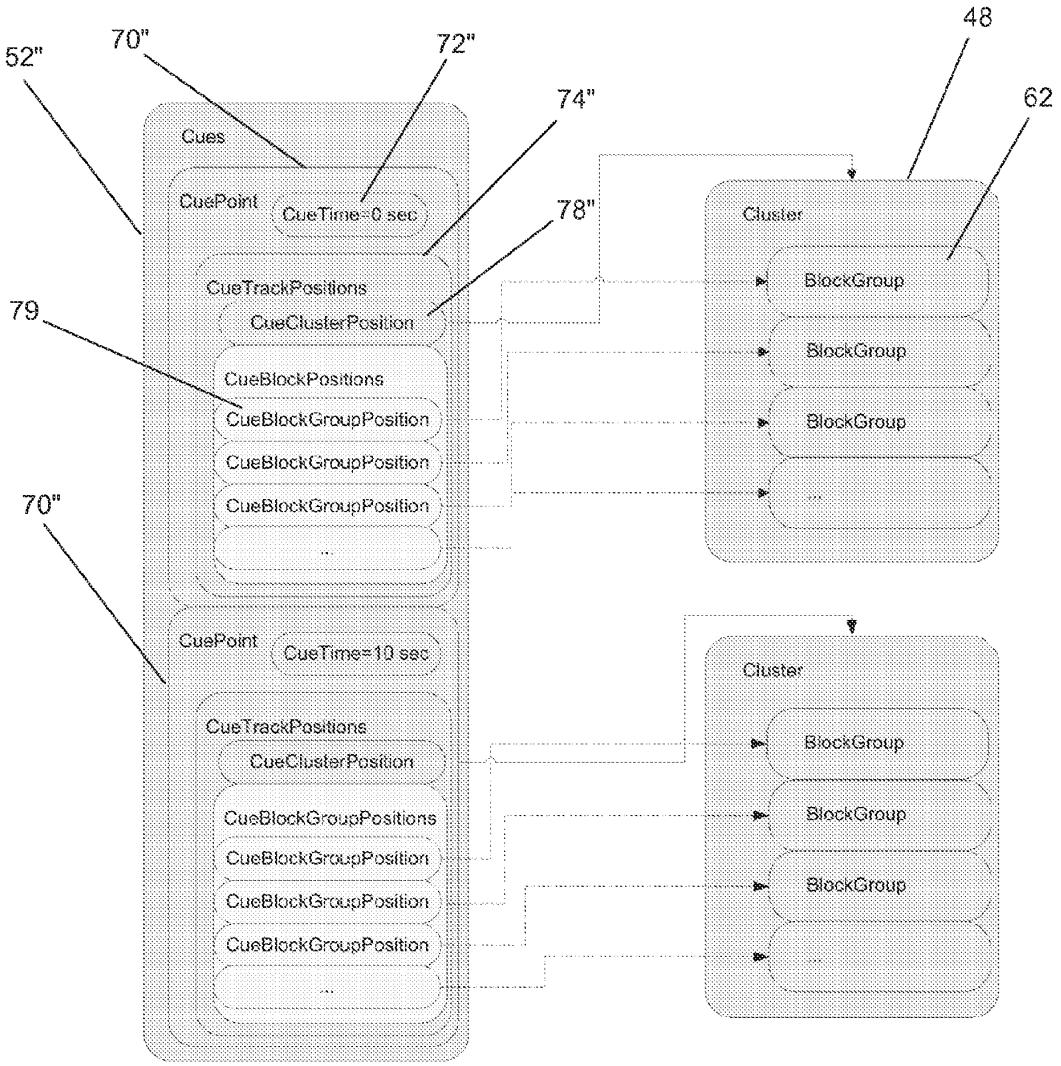

52"
70"
72"
74"
48
62
78"
79
70"
Cues
CuePoint
CueTime=0 sec
CueTrackPositions
CueClusterPosition
CueBlockPositions
CueBlockGroupPosition
CueBlockGroupPosition
CueBlockGroupPosition
...
CuePoint
CueTime=10 sec
CueTrackPositions
CueClusterPosition
CueBlockGroupPositions
CueBlockGroupPosition
CueBlockGroupPosition
CueBlockGroupPosition
...
Cluster
BlockGroup
BlockGroup
BlockGroup
...
Cluster
BlockGroup
BlockGroup
BlockGroup
...

FIG. 7

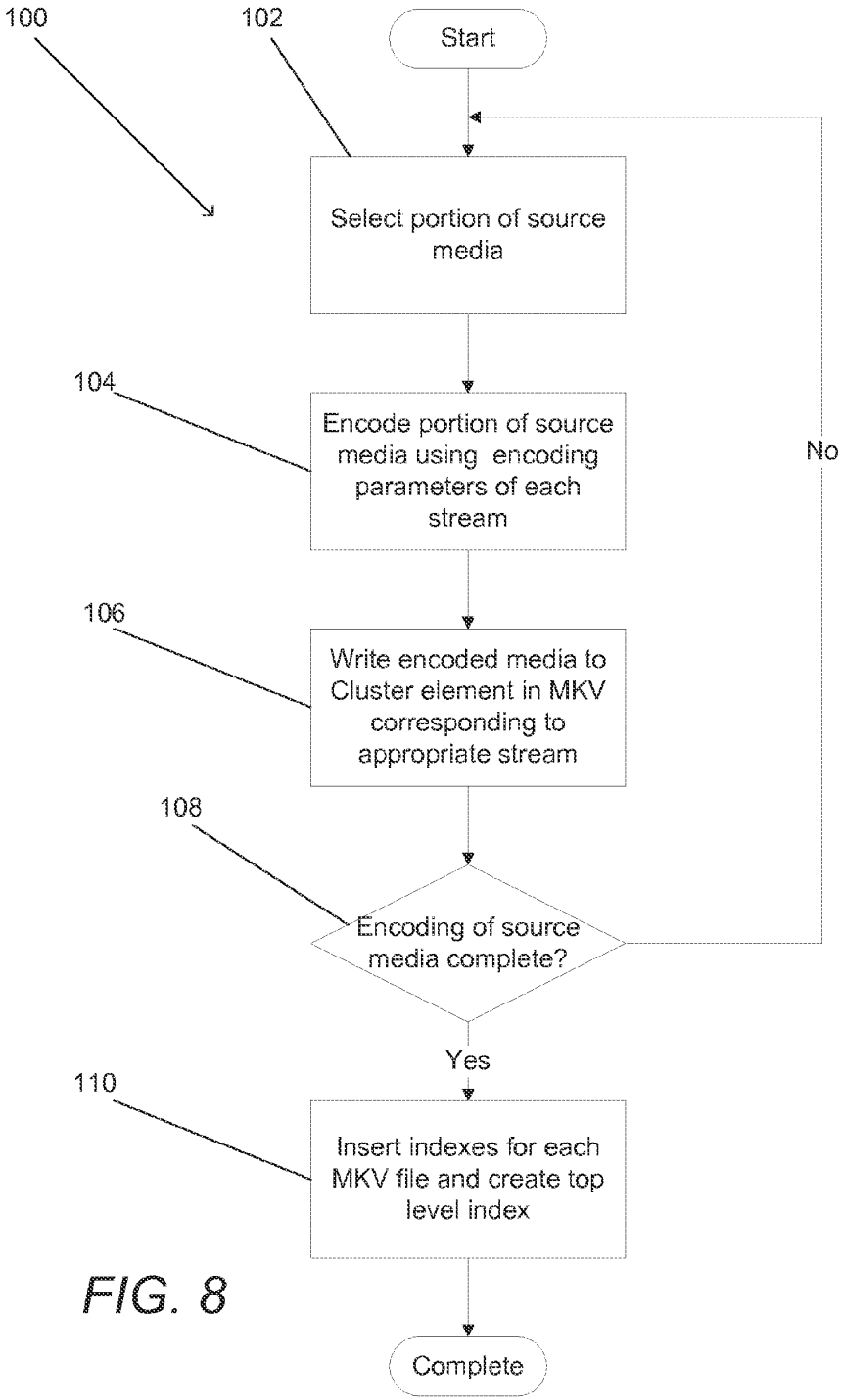

100
102
Start
Select portion of source media
104
Encode portion of source media using encoding parameters of each stream
106
Write encoded media to Cluster element in MKV corresponding to appropriate stream
108
Encoding of source media complete?
No
Yes
110
Insert indexes for each MKV file and create top level index
Complete

FIG. 8

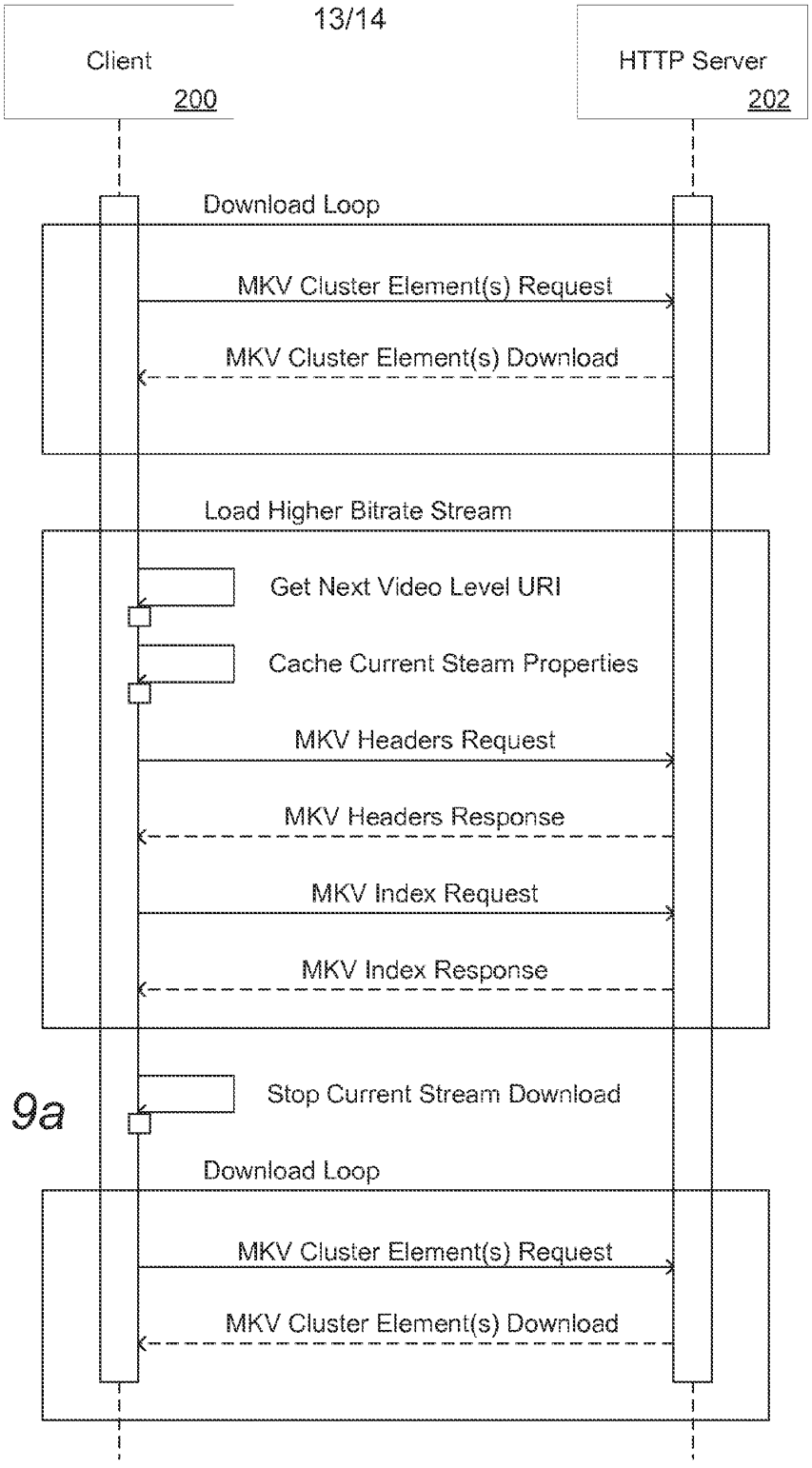

Client
200
HTTP Server
202
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download
Load Higher Bitrate Stream
Get Next Video Level URI
Cache Current Steam Properties
MKV Headers Request
MKV Headers Response
MKV Index Request
MKV Index Response
Stop Current Stream Download
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 9a

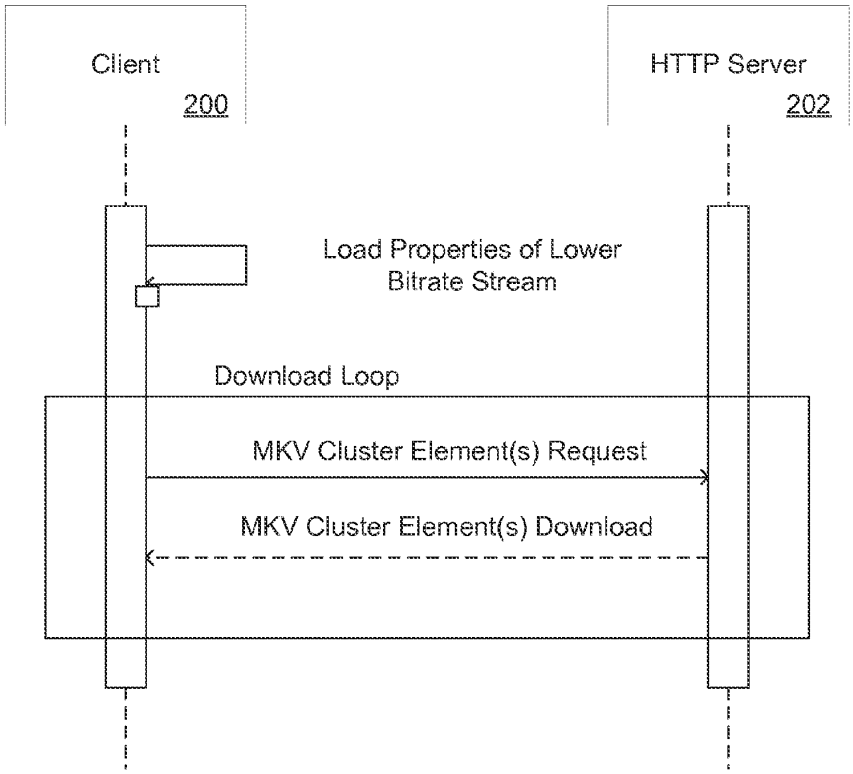

Client
200
HTTP Server
202
Load Properties of Lower Bitrate Stream
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 9b

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 July 2012 (12.07.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/094189 A1

(51) International Patent Classification:
H04N 7/12 (2006.01)

(21) International Application Number:
PCT/US2011/067243

(22) International Filing Date:
23 December 2011 (23.12.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/430,110	5 January 2011 (05.01.2011)	US
13/223,210	31 August 2011 (31.08.2011)	US
13/223,186	31 August 2011 (31.08.2011)	US

(71) Applicant *(for all designated States except US)*: **DIVX, LLC.** [US/US]; 2830 De La Cruz Boulevard, Santa Clara, CA 95050 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BRANESS, Jason** [US/US]; 15885 Concord Ridge Terrace, San Diego, CA 92127 (US). **SOROUSHIAN, Kourosh** [US/US]; 3758 Brookshire Street, San Diego, CA 92111 (US). **VAN DER SCHAAR, Auke Sjoerd** [NL/US]; 10775 Wilkins Ave., Apt. 102, Los Angeles, CA 90024 (US).

(74) Agent: **BAILEY, David J.**; Kauth, Pomeroy, Peck & Bailey LLP, 2400 E. Katella Ave., Suite 1050, Anaheim, CA 92806 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

[Continued on next page]

(54) Title: SYSTEMS AND METHODS FOR PERFORMING SMOOTH VISUAL SEARCH OF MEDIA ENCODED FOR ADAPTIVE BITRATE STREAMING USING TRICK PLAY STREAMS

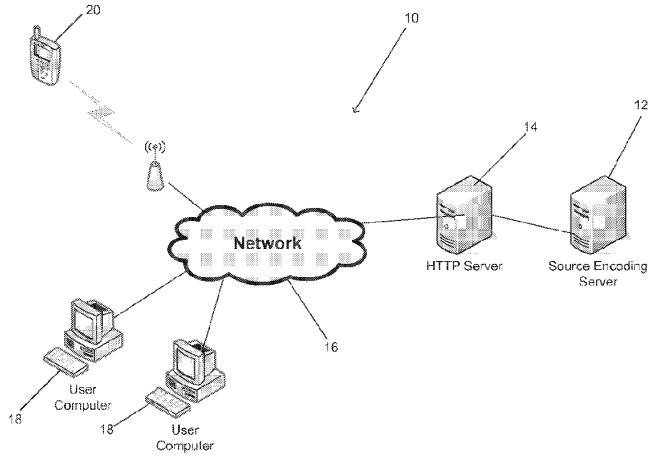


FIG. 1

(57) Abstract: Systems and methods for performing smooth visual search of media encoded for adaptive bitrate streaming using trick play streams are disclosed. One embodiment of the invention includes requesting and buffering portions of video from at least one of the alternative streams using a playback device, decoding the buffered portions of video using a decoder on the playback device, receiving at least one user instruction directing the playback device to perform a visual search of the media, requesting and buffering potions of video from the trick play stream using the playback device, and decoding the buffered portions of the trick play stream using a decoder on the playback device.

WO 2012/094189 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

D1:02357AB.PCT

SYSTEMS AND METHODS FOR PERFORMING SMOOTH VISUAL SEARCH OF MEDIA ENCODED FOR ADAPTIVE BITRATE STREAMING USING TRICK PLAY STREAMS

FIELD OF THE INVENTION

[0001] The present invention generally relates to adaptive streaming and more specifically to adaptive bitrate streaming systems that include visual search functionality.

BACKGROUND

[0002] The term streaming media describes the playback of media on a playback device, where the media is stored on a server and continuously sent to the playback device over a network during playback. Typically, the playback device stores a sufficient quantity of media in a buffer at any given time during playback to prevent disruption of playback due to the playback device completing playback of all the buffered media prior to receipt of the next portion of media. Adaptive bit rate streaming or adaptive streaming involves detecting the present streaming conditions (e.g. the user's network bandwidth and CPU capacity) in real time and adjusting the quality of the streamed media accordingly. Typically, the source media is encoded at multiple bit rates and the playback device or client switches between streaming the different encodings depending on available resources.

[0003] During streaming, a user may desire to view a different portion of the media. So called "trick play" functions are offered by many adaptive bitrate streaming systems, which enable the user to show intermittent frames of the video stream in the forward direction or the reverse direction (often with audio muted). The result is a jerky sequence of still images that the user can view to locate a desired portion of the media from which to commence normal playback.

[0004] Adaptive streaming solutions typically utilize either Hypertext Transfer Protocol (HTTP), published by the Internet Engineering Task Force and the World Wide

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0680
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Web Consortium as RFC 2616, or Real Time Streaming Protocol (RTSP), published by the Internet Engineering Task Force as RFC 2326, to stream media between a server and a playback device. HTTP is a stateless protocol that enables a playback device to request a byte range within a file. HTTP is described as stateless, because the server is not required to record information concerning the state of the playback device requesting information or the byte ranges requested by the playback device in order to respond to requests received from the playback device. RTSP is a network control protocol used to control streaming media servers. Playback devices issue control commands, such as "play" and "pause", to the server streaming the media to control the playback of media files. When RTSP is utilized, the media server records the state of each client device and determines the media to stream based upon the instructions received from the client devices and the client's state.

[0005] In adaptive streaming systems, the source media is typically stored on a media server as a top level index file pointing to a number of alternate streams that contain the actual video and audio data. Each stream is typically stored in one or more container files. Different adaptive streaming solutions typically utilize different index and media containers. The Synchronized Multimedia Integration Language (SMIL) developed by the World Wide Web Consortium is utilized to create indexes in several adaptive streaming solutions including IIS Smooth Streaming developed by Microsoft Corporation of Redmond, Washington, and Flash Dynamic Streaming developed by Adobe Systems Incorporated of San Jose, California. HTTP Adaptive Bitrate Streaming developed by Apple Computer Incorporated of Cupertino, California implements index files using an extended M3U playlist file (.M3U8), which is a text file containing a list of URIs that typically identify a media container file. The most commonly used media container formats are the MP4 container format specified in MPEG-4 Part 14 (i.e. ISO/IEC 14496-14) and the MPEG transport stream (TS) container specified in MPEG-2 Part 1 (i.e. ISO/IEC Standard 13818-1). The MP4 container format is utilized in IIS

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0681
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Smooth Streaming and Flash Dynamic Streaming. The TS container is used in HTTP Adaptive Bitrate Streaming.

[0006] The Matroska container is a media container developed as an open standard project by the Matroska non-profit organization of Aussonne, France. The Matroska container is based upon Extensible Binary Meta Language (EBML), which is a binary derivative of the Extensible Markup Language (XML). Decoding of the Matroska container is supported by many consumer electronics (CE) devices. The DivX Plus file format developed by DivX, LLC of San Diego, California utilizes an extension of the Matroska container format (i.e. is based upon the Matroska container format, but includes elements that are not specified within the Matroska format).

SUMMARY OF THE INVENTION

[0007] Systems and methods in accordance with embodiments of the invention perform smooth visual search of media encoded for adaptive bitrate streaming using trick play streams. A trick play stream encodes the source media in such a way that playback of the trick play stream with appropriate manipulation of the timestamps of the frames of the trick play stream appears to the user to be source media played back at a higher speed. In this way, the trick play stream can be utilized to perform a smooth visual search as opposed to the disjointed sequence of frames experienced when visual search is performed by jumping between intra or IDR frames in one of the streams utilized for normal playback. One embodiment of the invention includes requesting and buffering portions of video from at least one of the alternative streams using a playback device, decoding the buffered portions of video using a decoder on the playback device, receiving at least one user instruction directing the playback device to perform a visual search of the media, requesting and buffering potions of video from the trick play stream using the playback device, and decoding the buffered portions of the trick play stream using a decoder on the playback device.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0682
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0008] A further additional embodiment includes retrieving a top level index file using the playback device that identifies the alternative streams used during normal playback of the encoded video and identifies the trick play stream used during visual search of the encoded video.

[0009] Another embodiment includes a processor configured to communicate with memory, where the processor is configured by a playback client application stored in the memory. In addition, the playback client application configures the processor to: request portions of video from at least one of the alternative streams from a remote server and buffer the requested portions of video in memory; decode the buffered portions of video; receive at least one user instruction directing the playback device to perform a visual search of the media; request potions of video from the trick play stream from a remote server and buffer requested portions of video from the trick play stream; and decode the buffered portions of the trick play stream.

[0010] A still further embodiment of the invention includes repeatedly selecting a portion of the source video using the source encoder, transcoding the selected portion of the source video into a plurality of alternative portions of encoded video using the source encoder, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed group of pictures (GOP), transcoding the selected portion of the source video into a portion of trick play stream video for inclusion in a trick play stream using the source encoder, where the portion of trick play stream video is encoded at a lower frame rate and a lower resolution than the source video, writing each of the alternative portions of encoded video to a separate container file using the source encoder, writing the trick play stream to a separate container file using the source encoder, and adding an entry to an index that identifies the location of each frame in the portion of trick play stream video within the container file containing the trick play stream.

[0011] Another further embodiment includes a processor configured to communicate with memory, the processor configured by a source encoder application

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0683
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

stored in memory. In addition, the source encoder application configures the processor to: repeatedly select a portion of the source video using the source encoder; transcode the selected portion of the source video into a plurality of alternative portions of encoded video, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed group of pictures (GOP); transcode the selected portion of the source video into a portion of trick play stream video for inclusion in a trick play stream using the source encoder, where the portion of trick play stream video is encoded at a lower frame rate and a lower resolution than the source video; write each of the alternative portions of encoded video to a separate container file using the source encoder; write the trick play stream to a separate container file using the source encoder; and add an entry to an index that identifies the location of each frame in the portion of trick play stream video within the container file containing the trick play stream.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 is a network diagram of an adaptive bitrate streaming system in accordance with an embodiment of the invention.

[0013] FIG. 2 conceptually illustrates a top level index file and Matroska container files generated by the encoding of source media in accordance with embodiments of the invention.

[0014] FIG. 3 conceptually illustrates a specialized Matroska container file incorporating a modified Cues element in accordance with an embodiment of the invention.

[0015] FIGS. 4a – 4c conceptually illustrate the insertion of different types of media into the Clusters element of a Matroska container file subject to various constrains that facilitate adaptive bitrate streaming in accordance with embodiments of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0684
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0016] FIG. 4d conceptually illustrates the multiplexing of different types of media into the Clusters element of a Matroska container file subject to various constraints that facilitate adaptive bitrate streaming in accordance with an embodiment of the invention.

[0017] FIG. 4e conceptually illustrates the inclusion of a trick play stream into the Clusters element of a Matroska container file subject to various constraints that facilitate adaptive bitrate streaming in accordance with an embodiment of the invention.

[0018] FIG. 5 conceptually illustrates a modified Cues element of a specialized Matroska container file, where the Cues element includes information enabling the retrieval of Cluster elements using HTTP byte range requests in accordance with an embodiment of the invention.

[0019] FIG. 5a conceptually illustrates a modified Cues element of a specialized Matroska container file in accordance with an embodiment of the invention, where the Cues element is similar to the Cues element shown in FIG. 5 with the exception that attributes that are not utilized during adaptive bitrate streaming are removed.

[0020] FIG. 5b conceptually illustrates a modified Cues element of a specialized Matroska container file in accordance with an embodiment of the invention, where each CueTrackPositions element includes non-standard CueBlockPosition attributes that point to the BlockGroup (or SimpleBlock) elements within the Cluster element pointed to by the CueTrackPositions element.

[0021] FIG. 6 conceptually illustrates the indexing of Cluster elements within a specialized Matroska container file utilizing modified CuePoint elements within the container file in accordance with embodiments of the invention.

[0022] FIG. 7 conceptually illustrates the indexing of BlockGroup (or SimpleBlock) elements within a specialized Matroska container file utilizing non-standard CueBlockPosition attributes within the container file in accordance with embodiments of the invention.

[0023] FIG. 8 is a flow chart illustrating a process for encoding source media for adaptive bitrate streaming in accordance with an embodiment of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0685
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0024] FIGS. 9a – 9b conceptually illustrate communication between a playback device and an HTTP server associated with switching between streams in response to the streaming conditions experienced by the playback device and depending upon the index information available to the playback device prior to the decision to switch streams in accordance with embodiments of the invention.

DETAILED DISCLOSURE OF THE INVENTION

[0025] Turning now to the drawings, systems and methods for performing smooth visual search of media encoded for adaptive bitrate streaming using trick play streams are illustrated. In a number of embodiments, source media is encoded as a number of alternative streams for adaptive bitrate streaming and an additional trick play stream that can be utilized to perform visual search of the encoded media. The trick play stream encodes the source media in such a way that playback of the trick play stream appears to the user to be source media played back at a higher speed. In this way, the trick play stream can be utilized to perform a smooth visual search as opposed to the disjointed sequence of frames experienced when visual search is performed by jumping between intra or IDR frames in one of the streams utilized for normal playback. Each stream is stored in a Matroska (MKV) container file. In many embodiments, the Matroska container file is a specialized Matroska container file in that the manner in which the media in each stream is encoded and stored within the container is constrained to improve streaming performance. In several embodiments, the Matroska container file is further specialized in that additional index elements (i.e. elements that are not specified as part of the Matroska container format) can be included within the file to facilitate the retrieval of individual frames of video from the trick play stream during fast visual searches. A top level index file containing an index to the streams contained within each of the container files is also generated to enable adaptive bitrate streaming of the encoded media. In many embodiments, the top level index file is a Synchronized Multimedia Integration Language (SMIL) file containing URIs for each of

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0686
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the Matroska container files. In other embodiments, any of a variety of file formats can be utilized in the generation of the top level index file.

[0026] The performance of an adaptive bitstrate streaming system in accordance with embodiments of the invention can be significantly enhanced by encoding each portion of the source video at each bit rate in such a way that the portion of video is encoded in each stream as a single (or at least one) closed group of pictures (GOP) starting with an Instantaneous Decoder Refresh (IDR) frame, which is an intra frame. In a number of embodiments, the trick play stream is further constrained such that every frame of the trick play stream is an IDR frame. In many embodiments, the trick play stream is also encoded at a lower bitrate, frame rate and/or resolution. The frames of the trick play stream are stored within Cluster elements having time codes corresponding to the Cluster elements of the alternative streams used during normal playback. In this way, the playback device can switch between the alternative streams used during normal playback and the trick play stream at the completion of the playback of a Cluster and, irrespective of the stream from which a Cluster is obtained the first frame in the Cluster will be an IDR frame and can be decoded without reference to any encoded media other than the encoded media contained within the Cluster element.

[0027] In a number of embodiments, the indexes used in the Matroska container files containing the alternative streams used during normal playback is a reduced index in that the index only points to the IDRs at the start of each cluster. Retrieval of media using HTTP during streaming of a trick play stream can be improved by adding additional index information to the Matroska container files used to contain the trick play stream. The manner in which a conventional Matroska container file identifies a BlockGroup element within a Cluster element is using a block number. In many embodiments, the Matroska container file format is enhanced by adding a non-standard CueBlockPosition attribute to identify the location of a specific BlockGroup or SimpleBlock within the Matroska container file. In several embodiments, the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0687
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

CueBlockPosition attribute identifies the location of a specific BlockGroup or SimpleBlock within a Cluster element relative to the start of the Clusters element. By encoding the source media so that the index to the trick play stream identifies the location of individual frames within the trick play stream, a playback device can request individual frames from the trick play stream based upon the desired playback rate (typically expressed as a multiple of the normal playback rate) without the need to stream the entire trick play stream.

[0028] Adaptive streaming of source media encoded in the manner outlined above can be coordinated by a playback device in accordance with embodiments of the invention. The playback device obtains information concerning each of the available streams from the top level index file and selects one or more streams to utilize in the playback of the media. The playback device can then obtain header information from the Matroska container files containing the one or more bitstreams or streams, and the headers provide information concerning the decoding of the streams. The playback device can also request index information that indexes the encoded media stored within the relevant Matroska container files. The index information can be stored within the Matroska container files or separately from the Matroska container files in the top level index or in separate index files. The index information enables the playback device to request byte ranges corresponding to Cluster elements within the Matroska container file containing specific portions of encoded media via HTTP from the server. When the user selects to perform visual search of the encoded media using the trick play stream, the index enables the playback device to use multiple byte range requests to select portions of the file containing the trick play stream corresponding to individual frames of the trick play stream. In this way, the playback device can request only the frames from the trick play stream utilized in the visual search of the encoded media at a desired rate. When the user has located the desired portion of the encoded media, the playback device can resume adaptive bitrate streaming using the alternative streams encoded for normal playback.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0688
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0029] The encoding of source video for use in adaptive bitrate streaming systems that support visual search using trick play streams and the playback of the encoded source video using HTTP requests to achieve visual search of the encoded media in accordance with embodiments of the invention is discussed further below.

ADAPTIVE STREAMING SYSTEM ARCHITECTURE

[0030] An adaptive streaming system in accordance with an embodiment of the invention is illustrated in FIG. 1. The adaptive streaming system 10 includes a source encoder 12 configured to encode source media as a number of alternative streams. In the illustrated embodiment, the source encoder is a server. In other embodiments, the source encoder can be any processing device including a processor and sufficient resources to perform the transcoding of source media (including but not limited to video, audio, and/or subtitles). As is discussed further below, the source encoding server 12 can generate a top level index to a plurality of container files containing the alternative streams used during normal playback and a separate trick play stream. Alternative streams are streams that encode the same media content in different ways and/or at different bitrates to enable adaptive bitrate streaming by performing switches between the streams during normal playback based upon the streaming conditions. The trick play stream is used to perform smooth visual search of the encoded media in either the forward or reverse direction at a rate that is typically faster than the normal playback rate. In a number of embodiments, the streams can be encoded with different resolutions and/or at different frame rates. In many embodiments, the trick play stream encodes the source media in such a way that playback of the trick play stream appears to the user to be the source media smoothly playing back at a higher speed. The encoding of media for performing smooth visual searching is disclosed in U.S. Patent Application Serial No. 12/260,404 entitled "Application Enhancement Tacks" to Priyadarshi et al., filed October 29, 2008. The disclosure of U.S. Patent Application Serial No. 12/260,404 is incorporated by reference herein in its entirety. In several

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0689
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

embodiments, the trick play stream is encoded at a lower frame rate and/or at a lower resolution than the other streams.

[0031] The top level index file and the container files are uploaded to an HTTP server 14. A variety of playback devices can then use HTTP or another appropriate stateless protocol to request portions of the top level index file and the container files via a network 16 such as the Internet. The playback device can select between the alternative streams during normal playback based upon the streaming conditions and can request frames from the trick play stream when the user desires to perform a smooth visual search of the encoded media.

[0032] In many embodiments, the top level index file is a SMIL file and the media is stored in Matroska container files. As is discussed further below, the alternative streams are stored within the Matroska container file in a way that facilitates the adaptive bitrate streaming of the media. In many embodiments, the Matroska container files are specialized Matroska container files that include enhancements (i.e. elements that do not form part of the Matroska file format specification) that facilitate the retrieval of specific portions of media via HTTP during the adaptive bitrate streaming of the media. In several embodiments, the Matroska container file containing the trick play stream includes specific enhancements to facilitate fast visual search using the trick play stream.

[0033] In the illustrated embodiment, playback devices include personal computers 18 and mobile phones 20. In other embodiments, playback devices can include consumer electronics devices such as DVD players, Blu-ray players, televisions, set top boxes, video game consoles, tablets, and other devices that are capable of connecting to a server via HTTP and playing back encoded media. Although a specific architecture is shown in FIG. 1 any of a variety of architectures can be utilized that enable playback devices to request portions of the top level index file and the container files in accordance with embodiments of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0690
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

FILE STRUCTURE

[0034] Files generated by a source encoder and/or stored on an HTTP server for streaming to playback devices in accordance with embodiments of the invention are illustrated in FIG. 2. The files utilized in the adaptive bitrate streaming of the source media include a top level index 30 and a plurality of container files 32 that each contain at least one stream. The top level index file describes the content of each of the container files. As is discussed further below, the top level index file can take a variety of forms including a SMIL file and the container files can take a variety of forms including a specialized Matroska container file.

[0035] In many embodiments, each Matroska container file contains a single stream. For example, the stream could be one of a number of alternate video streams, an audio stream, one of a number of alternate audio streams, a subtitle stream, one of a number of alternate subtitle streams, a trick play stream, or one of a number of alternate trick play streams. In several embodiments, the Matroska container file includes multiple multiplexed streams. For example, the Matroska container could include a video stream, and one or more audio streams, one or more subtitle streams, and/or one or more trick play streams. As is discussed further below, in many embodiments the Matroska container files are specialized files. The encoding of the media and the manner in which the media is stored within Cluster elements within the Matroska container file can be subject to constraints designed to enhance the performance of an adaptive bitrate streaming system. In addition, the Matroska container file can include index elements that facilitate the location and downloading of Cluster elements from the various Matroska container files during the adaptive streaming of the media and the downloading of individual BlockGroup or SimpleBlock elements from within a Cluster element. Top level index files and Matroska container files that can be used in adaptive bitrate streaming systems in accordance with embodiments of the invention are discussed below.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0691
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

TOP LEVEL INDEX FILES

[0036] Playback devices in accordance with many embodiments of the invention utilize a top level index file to identify the container files that contain the streams available to the playback device for use in adaptive bitrate streaming. In many embodiments, the top level index files can include references to container files that each include an alternative stream of encoded media or a trick play stream. The playback device can utilize the information in the top level index file to retrieve encoded media from each of the container files according to the streaming conditions experienced by the playback device and/or instructions from the user related to performing visual search of the encoded media.

[0037] In several embodiments, the top level index file provides information enabling the playback device to retrieve information concerning the encoding of the media in each of the container files and an index to encoded media within each of the container files. In a number of embodiments, each container file includes information concerning the encoded media contained within the container file and an index to the encoded media within the container file and the top level index file indicates the portions of each container file containing this information. Therefore, a playback device can retrieve the top level index file and use the top level index file to request the portions of one or more of the container files that include information concerning the encoded media contained within the container file and an index to the encoded media within the container file. A variety of top level index files that can be utilized in adaptive bitrate streaming systems in accordance with embodiments of the invention are discussed further below.

TOP LEVEL INDEX SMIL FILES

[0038] In a number of embodiments, the top level index file utilized in the adaptive bitrate streaming of media is a SMIL file, which is an XML file that includes a list of URIs describing each of the streams and the container files that contain the streams. The URI can include information such as the "system-bitrate" of the stream contained

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0692
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

within the stream and information concerning the location of specific pieces of data within the container file.

[0039] The basic structure of a SMIL file involves providing an XML declaration and a SMIL element. The SMIL element defines the streams available for use in adaptive bitrate streaming and includes a HEAD element, which is typically left empty and a BODY element that typically only contains a PAR (parallel) element. The PAR element describes streams that can be played simultaneously (i.e. include media that can be presented at the same time).

[0040] The SMIL specification defines a number of child elements to the PAR element that can be utilized to specify the streams available for use in adaptive bitrate streaming. The VIDEO, AUDIO and TEXTSTREAM elements can be utilized to define a specific video, audio or subtitle stream. The VIDEO, AUDIO and TEXTSTREAM elements can collectively be referred to as media objects. The basic attributes of a media object are the SRC attribute, which specifies the full path or a URI to a container file containing the relevant stream, and the XML:LANG attribute, which includes a 3 letter language code. Additional information concerning a media object can be specified using the PARAM element. The PARAM element is a standard way within the SMIL format for providing a general name value pair. In a number of embodiments of the invention, specific PARAM elements are defined that are utilized during adaptive bitrate streaming.

[0041] In many embodiments, a "header-request" PARAM element is defined that specifies the size of the header section of the container file containing the stream. The value of the "header-request" PARAM element typically specifies the number of bytes between the start of the file and the start of the encoded media within the file. In many embodiments, the header contains information concerning the manner in which the media is encoded and a playback device retrieves the header prior to playback of the encoded media in order to be able to configure the decoder for playback of the encoded media. An example of a "header-request" PARAM element is follows:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0693
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
<param
  name="header-request"
  value="1026"
  valuetype="data" />
```

[0042] In a number of embodiments, a "mime" PARAM element is defined that specifies the MIME type of the stream. A "mime" PARAM element that identifies the stream as being an H.264 stream (i.e. a stream encoded in accordance with the MPEG-4 Advanced Video Codec standard) is as follows:

```
<param
  name="mime"
  value="V_MPEG4/ISO/AVC"
  valuetype="data" />
```

[0043] The MIME type of the stream can be specified using a "mime" PARAM element as appropriate to the encoding of a specific stream (e.g. AAC audio or UTF-8 text stream).

[0044] When the media object is a VIDEO element, additional attributes are defined within the SMIL file format specification including the systemBitrate attribute, which specifies the bitrate of the stream in the container file identified by the VIDEO element, and width and height attributes, which specify the dimensions of the encoded video in pixels. Additional attributes can also be defined using the PARAM element. In several embodiments, a "vbv" PARAM element is defined that specified the VBV buffer size of the video stream in bytes. The video buffering verifier (VBV) is a theoretical MPEG video buffer model used to ensure that an encoded video stream can be correctly buffered and played back at the decoder device. An example of a "vbv" PARAM element that specifies a VBV size of 1000 bytes is as follows:

```
<param
  name="vbv"
  value="1000"
  valuetype="data" />
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0694
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0045] An example of VIDEO element including the attributes discussed above is as follows:

```
<video
        src="http://cnd.com/video1_620kbps.mkv"
        systemBitrate="620"
        width="480"
        height="270" >
        <param
          name="vbv"
          value="1000"
          valuetype="data" />
</video>
```

[0046] Adaptive bitrate streaming systems in accordance with embodiments of the invention can support trick play streams, which can be used to provide smooth visual search through source content encoded for adaptive bitrate streaming. A trick play stream can be encoded that appears to be an accelerated visual search through the source media when played back, when in reality the trick play stream is simply a separate track encoding the source media at a lower frame rate. In many embodiments of the system a VIDEO element that references a trick play stream is indicated by the systemProfile attribute of the VIDEO element. In other embodiments, any of a variety of techniques can be utilized to signify within the top level index file that a specific stream is a trick play stream. An example of a trick play stream VIDEO element in accordance with an embodiment of the invention is as follows:

```
<video
        src="http://cnd.com/video_test2_600kbps.mkv"
        systemProfile="DivXPlusTrickTrack"
        width="480"
        height="240">
        <param name="vbv" value="1000" valuetype="data" />
        <param name="header-request" value="1000" valuetype="data" />
    </video>
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0695
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0047] In a number of embodiments of the invention, a "reservedBandwidth" PARAM element can be defined for an AUDIO element. The "reservedBandwidth" PARAM element specifies the bitrate of the audio stream in Kbps. An example of an AUDIO element specified in accordance with an embodiment of the invention is as follows:

```
<audio
  src="http://cnd.com/audio_test1_277kbps.mkv"
  xml:lang="gem"
<param
 name="reservedBandwidth"
 value="128"
 valuetype="data" />
/>
```

[0048] In several embodiments, the "reservedBandwidth" PARAM element is also defined for a TEXTSTREAM element. An example of a TEXTSTREAM element including a "reservedBandwidth" PARAM element in accordance with an embodiment of the invention is as follows:

```
<textstream
  src="http://cnd.com/text_stream_ger.mkv"
  xml:lang="gem"
<param
 name="reservedBandwidth"
 value="32"
 valuetype="data" />
/>
```

[0049] In other embodiments, any of a variety of mechanisms can be utilized to specify information concerning VIDEO, AUDIO, and SUBTITLE elements as appropriate to specific applications.

[0050] A SWITCH element is a mechanism defined within the SMIL file format specification that can be utilized to define adaptive or alternative streams. An example of the manner in which a SWITCH element can be utilized to specify alternative video streams at different bitrates is as follows:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0696
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
<switch>
  <video src="http://cnd.com/video_test1_300kbps.mkv"/>
  <video src="http://cnd.com/video_test2_900kbps.mkv"/>
  <video src="http://cnd.com/video_test3_1200kbps.mkv"/>
</switch>
```

[0051] The SWTICH element specifies the URLs of three alternative video streams. The file names indicate that the different bitrates of each of the streams. As is discussed further below, the SMIL file format specification provides mechanisms that can be utilized in accordance with embodiments of the invention to specify within the top level index SMIL file additional information concerning a stream and the container file in which it is contained.

[0052] In many embodiments of the invention, the EXCL (exclusive) element is used to define alternative tracks that do not adapt during playback with streaming conditions. For example, the EXCL element can be used to define alternative audio tracks or alternative subtitle tracks. An example of the manner in which an EXCL element can be utilized to specify alternative English and French audio streams is as follows:

```
<excl>
  <audio
    src="http://cnd.com/english-audio.mkv"
    xml:lang="eng"/>
  <audio
    src="http://cnd.com/french-audio.mkv"
    xml:lang="fre"/>
</excl>
```

[0053] An example of a top level index SMIL file that defines the attributes and parameters of two alternative video levels, an audio stream and a subtitle stream in accordance with an embodiment of the invention is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
  <head>
  </head>
  <body>
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0697
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

D1:02357AB.PCT

```
<par>
  <switch>
    <video
      src="http://cnd.com/video_test1_300kbps.mkv"
      systemBitrate="300"
      vbv="600"
      width="320"
      height="240" >
      <param
        name="vbv"
        value="600"
        valuetype="data" />
      <param
        name="header-request"
        value="1000"
        valuetype="data" />
    </video>
    <video
      src="http://cnd.com/video_test2_600kbps.mkv"
      systemBitrate="600"
      vbv ="900"
      width="640"
      height="480">
      <param
        name="vbv"
        value="1000"
        valuetype="data" />
      <param
        name="header-request"
        value="1000"
        valuetype="data" />
    </video>
  </switch>
  <audio
    src="http://cnd.com/audio.mkv"
    xml:lang="eng">
    <param
      name="header-request"
      value="1000"
      valuetype="data" />
    <param name="reservedBandwidth" value="128" valuetype="data" />
  </audio>
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0698
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

D1:02357AB.PCT

```
        <textstream
          src="http://cnd.com/subtitles.mkv"
          xml:lang="eng">
          <param
            name="header-request"
            value="1000"
            valuetype="data" />
          <param name="reservedBandwidth" value="32" valuetype="data" />
        </textstream>
      </par>
    </body>
</smil>
```

[0054] The top level index SMIL file can be generated when the source media is encoded for playback via adaptive bitrate streaming. Alternatively, the top level index SMIL file can be generated when a playback device requests the commencement of playback of the encoded media. When the playback device receives the top level index SMIL file, the playback device can parse the SMIL file to identify the available streams. The playback device can then select the streams to utilize to playback the content and can use the SMIL file to identify the portions of the container file to download to obtain information concerning the encoding of a specific stream and/or to obtain an index to the encoded media within the container file.

[0055] Although top level index SMIL files are described above, any of a variety of top level index file formats can be utilized to create top level index files as appropriate to a specific application in accordance with an embodiment of the invention. The use of top level index files to enable playback of encoded media using adaptive bitrate streaming in accordance with embodiments of the invention is discussed further below.

STORING MEDIA IN MATROSKA FILES FOR STREAMING

[0056] A Matroska container file used to store encoded video in accordance with an embodiment of the invention is illustrated in FIG. 3. The container file 32 is an Extensible Binary Markup Language (EBML) file that is an extension of the Matroska

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0699
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

container file format. The specialized Matroska container file 32 includes a standard EBML element 34, and a standard Segment element 36 that includes a standard Seek Head element 40, a standard Segment Information element 42, and a standard Tracks element 44. These standard elements describe the media contained within the Matroska container file. The Segment element 36 also includes a standard Clusters element 46. As is described below, the manner in which encoded media is inserted within individual Cluster elements 48 within the Clusters element 46 is constrained to improve the playback of the media in an adaptive streaming system. In many embodiments, the constraints imposed upon the encoded video are consistent with the specification of the Matroska container file format and involve encoding the video so that each Cluster includes at least one closed GOP commencing with an IDR frame. When the stream is a trick play stream, each frame in the stream is an IDR frame. In addition to the above standard elements, the Segment element 36 also includes a modified version of the standard Cues element 52. As is discussed further below, the Cues element includes specialized CuePoint elements (i.e. non-standard CuePoint elements) that facilitate the retrieval of the media contained within specific Cluster elements via HTTP and, in the case of the trick play stream, facilitate the retrieval of specific frames of video from within specific Cluster elements via HTTP or a similar stateless protocol.

[0057] The constraints imposed upon the encoding of media and the formatting of the encoded media within the Clusters element of a Matroska container file for adaptive bitrate streaming supporting visual search and the additional index information inserted within the container file based upon whether the stream is one of the alternative streams used during normal playback or a trick play stream in accordance with embodiments of the invention is discussed further below.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0700
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ENCODING MEDIA FOR INSERTION IN CLUSTER ELEMENTS

[0058] An adaptive bitrate streaming system provides a playback device with the option of selecting between different streams of encoded media during playback according to the streaming conditions experienced by the playback device or in response to a user instruction to perform a visual search using the trick play stream. In many embodiments, switching between streams is facilitated by separately pre-encoding discrete portions of the source media in accordance with the encoding parameters of each stream and then including each separately encoded portion in its own Cluster element within the stream's container file. Furthermore, the media contained within each cluster is encoded so that the media is capable of playback without reference to media contained in any other cluster within the stream. In this way, each stream includes a Cluster element corresponding to the same discrete portion of the source media and, at any time, the playback device can select the Cluster element from the stream that is most appropriate to the streaming conditions experienced by the playback device and can commence playback of the media contained within the Cluster element. Accordingly, the playback device can select clusters from different streams as the streaming conditions experienced by the playback device change over time. When the user provides a trick play command, the playback device can select frames from an appropriate Cluster element in the trick play stream based upon the direction of visual search and the speed of the visual search. The frame rate of the trick play stream is typically much lower than that of the other streams (e.g. 5 frames per second of the source content as opposed to 30 frames per second of the source content for a stream used during normal playback). In many embodiments, each frame of the trick play stream is an IDR frame. By playing back the frames of the trick play stream at a higher speed than the nominal frame rate of the trick play stream, a smooth visual search can be provided without significantly increasing the processing burden on the decode device beyond that typically experienced during normal playback.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0701
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The specific constraints applied to the media encoded within each Cluster element depending upon the type of media (i.e. video, audio, or subtitles) are discussed below.

[0059] A Clusters element of a Matroska container file containing a video stream in accordance with an embodiment of the invention is illustrated in FIG. 4a. The Clusters element 46 includes a plurality of Cluster elements 48 that each contain a discrete portion of encoded video. In the illustrated embodiment, each Cluster element 48 includes encoded video corresponding to a two second fragment of the source video. In other embodiments, the Cluster elements include encoded video corresponding to a portion of the source video having a greater or lesser duration than two seconds. The smaller the Cluster elements (i.e. the smaller the duration of the encoded media within each Cluster element), the higher the overhead associated with requesting each Cluster element. Therefore, a tradeoff exists between the responsiveness of the playback device to changes in streaming conditions and the effective data rate of the adaptive streaming system for a given set of streaming conditions (i.e. the portion of the available bandwidth actually utilized to transmit encoded media). In a number of embodiments, the Cluster elements of the alternative video streams used during normal playback contain portions of video having the same duration and the Cluster elements of the trick play stream have a longer duration. In a number of embodiments, the Cluster elements of the alternative video streams contain two second portions of video and the Cluster elements containing the trick play stream contain 64 frames of video, which corresponds to approximately 12.8 seconds of the source media (depending on the frame rate of the trick play stream). In several embodiments, the encoded video sequences in the Cluster elements for a stream have different durations. Each Cluster element 48 includes a Timecode element 60 indicating the start time of the encoded video within the Cluster element and a plurality of BlockGroup (or SimpleBlock) elements. As noted above, the encoded video stored within the Cluster is constrained so that the encoded video can be played back without reference to the encoded video contained within any of the other Cluster elements in the container file.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0702
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

In many embodiments, encoding the video contained within the Cluster element as a GOP in which the first frame is an IDR frame enforces the constraint. In the illustrated embodiment, the first BlockGroup (or SimpleBlock) element 62 contains an IDR frame (i.e. an intra frame). Therefore, the first BlockGroup (or SimpleBlock) element 62 does not include a ReferenceBlock element. The first BlockGroup (or SimpleBlock) element 62 includes a Block element 64, which specifies the Timecode attribute of the frame encoded within the Block element 64 relative to the Timecode of the Cluster element 48. Subsequent BlockGroup (or SimpleBlock) elements 66 are not restricted in the types of frames that they can contain (other than that they cannot reference frames that are not contained within the Cluster element). Therefore, subsequent BlockGroup (or SimpleBlock) elements 66 can include ReferenceBlock elements 68 referencing other BlockGroup element(s) utilized in the decoding of the frame contained within the BlockGroup (or SimpleBlock) element or can contain IDR frames and are similar to the first BlockGroup (or SimpleBlock) element 62. As noted above, the manner in which encoded video is inserted within the Cluster elements of the Matroska file conforms with the specification of the Matroska file format.

[0060] The insertion of encoded audio and subtitle information within a Clusters element 46 of a Matroska container file in accordance with embodiments of the invention is illustrated in FIGS. 4b and 4c. In the illustrated embodiments, the encoded media is inserted within the Cluster elements 48 subject to the same constraints applied to the encoded video discussed above with respect to FIG. 4a. In addition, the duration of the encoded audio and subtitle information within each Cluster element corresponds to the duration of the encoded video in the corresponding Cluster element of the Matroska container file containing the encoded video. In other embodiments, the Cluster elements within the container files containing the audio and/or subtitle streams need not correspond with the start time and duration of the Cluster elements in the container files containing the alternative video streams.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0703
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

MULIPLEXING STREAMS IN A SINGLE MKV CONTAINER FILE

[0061] The Clusters elements shown in FIGS. 4a – 4c assume that a single stream is contained within each Matroska container file. In several embodiments, media from multiple streams is multiplexed within a single Matroska container file. In this way, a single container file can contain a video stream multiplexed with one or more corresponding audio streams, and/or one or more corresponding subtitle streams. Storing the streams in this way can result in duplication of the audio and subtitle streams across multiple alternative video streams. However, the seek time to retrieve encoded media from a video stream and an associated audio, and/or subtitle stream can be reduced due to the adjacent storage of the data on the server. The Clusters element 46 of a Matroska container file containing multiplexed video, audio and subtitle data in accordance with an embodiment of the invention is illustrated in FIG. 4d. In the illustrated embodiment, each Cluster element 48 includes additional BlockGroup (or SimpleBlock) elements for each of the multiplexed streams. The first Cluster element includes a first BlockGroup (or SimpleBlock) element 62v for encoded video that includes a Block element 64v containing an encoded video frame and indicating the Timecode attribute of the frame relative to the start time of the Cluster element (i.e. the Timecode attribute 60). A second BlockGroup (or SimpleBlock) element 62a includes a Block element 64a including an encoded audio sequence and indicating the timecode of the encoded audio relative to the start time of the Cluster element, and a third BlockGroup (or SimpleBlock) element 62s including a Block element 64s containing an encoded subtitle and indicating the timecode of the encoded subtitle relative to the start time of the Cluster element. Although not shown in the illustrated embodiment, each Cluster element 48 likely would include additional BlockGroup (or SimpleBlock) elements containing additional encoded video, audio or subtitles. Despite the multiplexing of the encoded video, audio, and/or subtitle streams, the same constraints concerning the encoded media apply.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0704
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

INCORPORATING TRICK PLAY STREAMS IN MKV CONTAINER FILES

[0062] As noted above, a separate trick play stream can be encoded that appears to be a smooth visual search through the source media when played back, when in reality the trick play stream is simply a separate stream encoding the source media at a lower frame rate and played back at a higher rate. In several embodiments, the trick play stream is created by generating a trick play stream in the manner outlined in U.S. Patent Application No. 12/260,404 and inserting the trick play stream into a Matroska container file subject to the constraints mentioned above with respect to insertion of a video stream into a Matroksa container file. In many embodiments, the trick play stream is also subject to the further constraint that every frame in each Cluster element in the trick play stream is encoded as an IDR frame (i.e. an intra frame). Transitions to and from a trick play stream can be treated in the same way as transitions between any of the other encoded streams are treated within an adaptive bitrate streaming system in accordance with embodiments of the invention. As noted above, however, the duration of the Cluster elements of the trick play streams need not correspond to the duration of the Cluster elements of the alternative video streams utilized during normal playback. Playback of the frames contained within the trick play stream to achieve accelerated visual search typically involves the playback device manipulating the timecodes assigned to the frames of encoded video prior to providing the frames to the playback device's decoder to achieve a desired increase in rate of accelerated search (e.g. x2, x4, x6, etc.).

[0063] A Clusters element containing encoded media from a trick play stream is shown in FIG. 4e. In the illustrated embodiment, the encoded trick play stream is inserted within the Cluster elements 48 subject to the same constraints applied to the encoded video discussed above with respect to FIG. 4a. However, each Block element contains an IDR. Therefore, the BlockGroup (or SimpleBlock) elements do not include a ReferenceBlock element. In other embodiments, the Cluster elements within the container file containing the trick play stream need not correspond with the start time

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0705
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

and duration of the Cluster elements in the container files containing the alternative video streams utilized during normal playback. Furthermore, the frames of the trick play stream can be contained within a SimpleBlock element.

[0064] In many embodiments, source content can be encoded to provide a single trick play stream or multiple trick play streams for use by the adaptive bit rate streaming system. When a single trick play stream is provided, the trick play stream is typically encoded at a low bitrate, low frame rate and low resolution. For example, a trick play stream could be encoded at around 384 kbps with a frame rate of 5 fps. In a number of instances, a trick play stream could also be encoded at a lower resolution. Even when the trick play stream is encoded at a very low bitrate requesting all frames of the trick play stream can require considerable bandwidth. For example, a trick play stream encoded at 384 kbps and 5 fps played at 8x visual search speed utilizes at least 2,304 kbps, which is much higher than the lowest streaming levels that are used in many adaptive bitrate streaming systems, and is played back at 40 fps, which is beyond the capabilities of most devices (typically limited to 30 fps). Therefore, the visual search could stall at high visual search speeds during network congestion or simply due to the limitations of the playback device. In a number of embodiments of the invention, the bandwidth utilized during visual search is reduced by only requesting the frames from the trick play stream utilized at the visual search speed. In this way, the frame rate and the bandwidth utilization can remain relatively constant irrespective of the speed of the visual search. As is discussed further below, the ability to request individual frames from within a Cluster element involves utilizing a modified Cues element to index the content within a Matroska container file containing a trick play stream to index each frame within the trick play stream.

[0065] When multiple alternative trick play streams are provided, a separate trick play stream can be utilized for each visual search speed. Multiple alternative trick play streams can also be provided that encode the trick play stream at different bitrates to enable adaptive rate streaming with respect to the trick play stream.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0706
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

INDEXING CLUSTERS WITHIN MKV CONTAINER FILES FOR NORMAL PLAYBACK

[0066] The specification for the Matroska container file format provides for an optional Cues element that is used to index Block elements within the container file. A modified Cues element 52 that can be incorporated into a Matroska container file in accordance with an embodiment of the invention to facilitate the requesting of clusters by a playback device using HTTP during normal playback is illustrated in FIG. 5. The modified Cues element 52 includes a plurality of CuePoint elements 70 that each include a CueTime attribute 72. Each CuePoint element includes a CueTrackPositions element 74 containing the CueTrack 76 and CueClusterPosition 78 attributes. In many embodiments, the CuePoint element is mainly configured to identify a specific Cluster element as opposed to a specific Block element within a Cluster element. Although, in several applications the ability to seek to specific BlockGroup (or SimpleBlock) elements within a Cluster element is required and additional index information is included in the Cues element.

[0067] The use of a modified Cues element to index encoded media within a Clusters element of a Matroska file in accordance with an embodiment of the invention is illustrated in FIG. 6. A CuePoint element is generated to correspond to each Cluster element within the Matroska container file. The CueTime attribute 72 of the CuePoint element 70 corresponds to the Timecode attribute 60 of the corresponding Cluster element 48. In addition, the CuePoint element contains a CueTrackPositions element 74 having a CueClusterPosition attribute 78 that points to the start of the corresponding Cluster element 48. The CueTrackPositions element 74 can also include a CueBlockNumber attribute, which is typically used to indicate the Block element containing the first IDR frame within the Cluster element 48.

[0068] As can readily be appreciated the modified Cues element 52 forms an index to each of the Cluster elements 48 within the Matroska container file. Furthermore, the CueTrackPosition elements provide information that can be used by a playback device to request the byte range of a specific Cluster element 48 via HTTP or another suitable

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0707
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

protocol from a remote server. The Cues element of a conventional Matroska file does not directly provide a playback device with information concerning the number of bytes to request from the start of the Cluster element in order to obtain all of the encoded video contained within the Cluster element. The size of a Cluster element can be inferred in a modified Cuse element by using the CueClusterPosition attribute of the CueTrackPositions element that indexes the first byte of the next Cluster element. Alternatively, additional CueTrackPosition elements can be added to the modified Cues elements in accordance with embodiments of the invention that index the last byte of the Cluster element (in addition to the CueTrackPositions elements that index the first byte of the Cluster element), and/or a non-standard CueClusterSize attribute that specifies the size of the Cluster element pointed to by the CueClusterPosition attribute is included in each CueTrackPosition element to assist with the retrieval of specific Cluster elements within a Matroska container file via HTTP byte range requests or a similar protocol.

[0069] The modification of the Cues element in the manner outlined above significantly simplifies the retrieval of Cluster elements from a Matroska container file via HTTP or a similar protocol during adaptive bitrate streaming. In addition, by only indexing the first frame in each Cluster the size of the index is significantly reduced. Given that the index is typically downloaded prior to playback, the reduced size of the Cues element (i.e. index) means that playback can commence more rapidly. Using the CueClusterPosition elements, a playback device can request a specific Cluster element from the stream most suited to the streaming conditions experienced by the playback device by simply referencing the index of the relevant Matroska container file using the Timecode attribute for the desired Cluster element.

[0070] A number of the attributes within the Cues element are not utilized during adaptive bitrate streaming. Therefore, the Cues element can be further modified by removing the unutilized attributes to reduce the overall size of the index for each Matroska container file. A modified Cues element that can be utilized in a Matroska

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0708
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

container file that includes a single encoded stream in accordance with an embodiment of the invention is illustrated in FIG. 5a. The Cues element 52' shown in FIG. 5a is similar to the Cues element 52 shown in FIG. 5 with the exception that the CuePoint elements 70' do not include a CueTime attribute (see 72 in FIG. 5) and/or the CueTrackPositions elements 74' do not include a CueTrack attribute (76 in FIG. 5). When the portions of encoded media in each Cluster element in the Motroska container file have the same duration, the CueTime attribute is not necessary. When the Matroska contain file includes a single encoded stream, the CueTrack attribute is not necessary. In other embodiments, the Cues element and/or other elements of the Matroska container file can be modified to remove elements and/or attributes that are not necessary for the adaptive bitrate streaming of the encoded stream contained within the Matroska container file, given the manner in which the stream is encoded and inserted in the Matroska container file.

[0071] Although various modifications to the Cues element to include information concerning the size of each of the Cluster elements within a Matroska container file and to eliminate unnecessary attributes are described above, many embodiments of the invention utilize a conventional Matroska container. In several embodiments, the playback device simply determines the size of Cluster elements on the fly using information obtained from a conventional Cues element, and/or relies upon a separate index file containing information concerning the size and/or location of the Cluster elements within the MKV container file. In several embodiments, the additional index information is stored in the top level index file. In a number of embodiments, the additional index information is stored in separate files that are identified in the top level index file. When index information utilized to retrieve Cluster elements from a Matroska container file is stored separately from the container file, the Matroska container file is still typically constrained to encode media for inclusion in the Cluster elements in the manner outlined above. In addition, wherever the index information is located, the index information will typically index each Cluster element and include (but

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0709
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

not be limited to) information concerning at least the starting location and, in many instances, the size of each Cluster element.

INDEXING CLUSTERS WITHIN A MKV CONTAINER CONTAINING A TRICK PLAY STREAM

[0072] The modified Cues element utilized in MKV container files containing streams utilized during normal playback index each Cluster element within the MKV container file. As noted above, not indexing every frame of the stream reduces the overall size of the index and the time taken to download the index prior to the commencement of playback. When performing higher rate visual searches, the ability to download only the frames displayed during the visual search can significantly reduce the bandwidth requirements for performing visual search and the processing load on the playback device. Therefore, the index to a container file containing a trick play stream in accordance with many embodiments of the invention indexes all of the frames in the trick play stream.

[0073] A modified Cues element of a MKV container file containing a trick play stream in accordance with an embodiment of the invention is illustrated in FIG. 5b. The modified Cues element 52" includes a number of CuePoint elements 70" that each include a CueTime attribute 72" and a CueTrackPositions element 74" that references a Cluster element within the MKV container file (now shown). Each CueTrackPositions element 74" includes a CueClusterPosition attribute 78" that indicates the location within the MKV container file of the start of the Cluster element (now shown) referenced by the CuePoint element 70". The CueTrackPositions element 74" also includes a CueBlockPositions element 79", which is a non-standard element that includes CueBlockPosition attributes 80" that reference the starting location of each of the BlockGroup or SimpleBlock elements that contain frames in the Cluster element referenced by the CuePoint element 70". The MKV container file format specification does not provide for the indexing of the location of BlockGroups (or SimpleBlocks) within the MKV container file. Therefore, the CueBlockPostion element 79" is a non-

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0710
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

standard element specified for the purpose of supporting the selective downloading of frames from trick play streams by playback devices.

[0074] The manner in which CueBlockPosition attributes of a modified Cues element in an MKV container file containing a trick play stream reference the BlockGroup or SimpleBlock elements in the Cluster elements of the MKV container file in accordance with an embodiment of the invention is illustrated in FIG. 7. The CueClusterPosition attribute 78" in each CueTrackPositions element 74" points to the start of a Cluster element 48 within the MKV container file. The CueBlockPosition attributes 80" within the CueBlockPositions element 79" point to the start of each BlockGroup or SimpleBlock element 62 within the Cluster element pointed to by the CueCluster Position attribute 78". In many embodiments, the CueBlockPositions element 79" is identified with the ID 0X78 and the CueBlockPosition attribute 80" is identified with the ID 0X79. In this way, a playback device can use multiple byte range HTTP requests to only request the portions of each Cluster (i.e. the BlockGroup or Simple Block elements) containing the frames that will be utilized during visual search. When a playback device selectively requests portions of the Cluster element to exclude specific BlockGroup or SimpleBlock elements, the Cluster element that is received by the playback device is a valid Cluster element with the exception that the size attribute of the Cluster element will be incorrect. In a number of embodiments, the size of the Cluster element is modified by the playback device as it is received and/or parsed so that it can be processed like any other Cluster element. Playback of trick play streams in accordance with embodiments of the invention is discussed further below.

ENCODING SOURCE MEDIA FOR ADAPTIVE BITRATE STREAMING

[0075] A process for encoding source media as a top level index file and a plurality of Matroska container files for use in an adaptive bitrate streaming system in accordance with an embodiment of the invention is illustrated in FIG. 8. The encoding process 100 commences by selecting (102) a first portion of the source media and encoding (104) the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0711
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

source media using the encoding parameters for each stream. When the portion of media is video, then the portion of source video is encoded as a single GOP commencing with an IDR frame. In many embodiments, encoding parameters used to create the alternative GOPs vary based upon bitrate, frame rate, encoding parameters and resolution. In this way, the portion of media is encoded as a set of interchangeable alternatives and a portion of a trick play stream. A playback device can select the alternative most appropriate to the streaming conditions experienced by the playback device and can utilize the trick play stream to perform visual search of the encoded media. When different resolutions are supported, the encoding of the streams is constrained so that each stream has the same display aspect ratio. A constant display aspect ratio can be achieved across different resolution streams by varying the sample aspect ratio with the resolution of the stream. In many instances, reducing resolution can result in higher quality video compared with higher resolution video encoded at the same bit rate. In many embodiments, the source media is itself encoded and the encoding process (104) involves transcoding or transrating of the encoded source media according to the encoding parameters of each of the alternative streams supported by the adaptive bitrate streaming system.

[0076] Once the source media has been encoded as a set of alternative portions of encoded media, each of the alternative portions of encoded media is inserted (106) into a Cluster element within the Matroska container file corresponding to the stream to which the portion of encoded media belongs. In many embodiments, the encoding process also constructs indexes for each Matroska container file as media is inserted into Cluster elements within the container. Therefore, the process 100 can also include creating a CuePoint element that points to the Cluster element inserted within the Matroska container file. When the CuePoint element points to the Cluster element of a trick play stream, the CuePoint element also includes CueBlockPosition attributes that point to the BlockGroup (or SimpleBlock) elements within the Cluster element. The CuePoint element can be held in a buffer until the source media is completely encoded.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0712
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Although the above process describes encoding each of the alternative portions of encoded media and the portion of media included in the trick play stream sequentially in a single pass through the source media, many embodiments of the invention involve performing a separate pass through the source media to encode each of the alternative streams and/or the trick play stream.

[0077] Referring back to FIG. 8, the process continues to select (102) and encode (104) portions of the source media and then insert (106) the encoded portions of media into the Matroska container file corresponding to the appropriate stream until the entire source media is encoded for adaptive bitrate streaming (108). At which point, the process can insert an index (110) into the Matroska container for each stream and create (112) a top level index file that indexes each of the encoded streams contained within the Matroska container files. As noted above, the indexes can be created as encoded media is inserted into the Matroska container files so that a CuePoint element indexes each Cluster element within the Mastroska container file (and BlockGroup or SimpleBlock elements when the Matroska container file contains a trick play stream). Upon completion of the encoding each of the CuePoint elements can be included in a Cues element and the Cues element inserted into the Matroska container file following the Clusters element.

[0078] Following the encoding of the source media to create Matroska container files containing each of the streams generated during the encoding process, which can include the generation of trick play streams, and a top level index file that indexes each of the streams within the Matroska container files, the top level index file and the Matroska container files can be uploaded to an HTTP server for adaptive bitrate streaming to playback devices. The adaptive bitrate streaming of media encoded in accordance with embodiments of the invention using HTTP requests is discussed further below.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0713
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ADAPTIVE BITRATE STREAMING FROM MKV CONTAINER FILES USING HTTP

[0079] When source media is encoded so that there are alternative streams contained in separate Matroska container files for at least one of video, audio, and subtitle content, adaptive streaming of the media contained within the Matroska container files can be achieved using HTTP requests or a similar stateless data transfer protocol. HTTP requests can also be used to transition from normal playback to visual search using a separate trick play stream. In many embodiments, a playback device requests the top level index file resident on the server and uses the index information to identify the streams that are available to the playback device. The playback device can then retrieve the indexes for one or more of the Matroska files and can use the indexes to request media from one or more of the streams contained within the Matroska container files using HTTP requests or using a similar stateless protocol. As noted above, many embodiment of the invention implement the indexes for each of the Matroska container files using a modified Cues element. In a number of embodiments, however, the encoded media for each stream is contained within a standard Matroska container file and separate index file(s) can also be provided for each of the container files. Based upon the streaming conditions experienced by the playback device, the playback device can select media from alternative streams encoded at different bitrates. When the media from each of the streams is inserted into the Matroska container file in the manner outlined above, transitions between streams can occur upon the completion of playback of media within a Cluster element. This is true whether the transition is between alternative streams during normal playback or a trick play stream utilized during visual search. Therefore, the size of the Cluster elements (i.e the duration of the encoded media within the Cluster elements) is typically chosen so that the playback device is able to respond quickly enough to changing streaming conditions and to instructions from the user that involve utilization of a trick play stream. The smaller the Cluster elements (i.e. the smaller the duration of the encoded media within each Cluster element) the higher the overhead associated with requesting

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0714
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

each Cluster element. Therefore, a tradeoff exists between the responsiveness of the playback device to changes in streaming conditions and the effective data rate of the adaptive streaming system for a given set of streaming conditions (i.e. the portion of the available bandwidth actually utilized to transmit encoded media). In many embodiments, the size of the Cluster elements is chosen so that each Cluster element contains two seconds of encoded media. In other embodiments, the duration of the encoded media can be greater or less than two seconds and/or the duration of the encoded media can vary from Cluster element to Cluster element.

[0080] Communication between a playback device or client and an HTTP server during the normal playback of media encoded in separate streams contained within Matroska container files indexed by a top level index file in accordance with an embodiment of the invention is illustrated in FIG. 9a. In the illustrated embodiment, the playback device 200 commences playback by requesting the top level index file from the server 202 using an HTTP request or a similar protocol for retrieving data. The server 202 provides the bytes corresponding to the request. The playback device 200 then parses the top level index file to identify the URIs of each of the Matroska container files containing the streams of encoded media derived from a specific piece of source media. The playback device can then request the byte ranges corresponding to headers of one or more of the Matroska container files via HTTP or a similar protocol, where the byte ranges are determined using the information contained in the URI for the relevant Matroska container files (see discussion above). The server returns the following information in response to a request for the byte range containing the headers of a Matroska container file:

ELEM("EBML")
ELEM("SEEKHEAD")
ELEM("SEGMENTINFO")
ELEM("TRACKS")

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0715
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0081] The EBML element is typically processed by the playback device to ensure that the file version is supported. The SeekHead element is parsed to find the location of the Matroska index elements and the SegmentInfo element contains two key elements utilized in playback: TimecodeScale and Duration. The TimecodeScale specifies the timecode scale for all timecodes within the Segment of the Matroska container file and the Duration specifies the duration of the Segment based upon the TimecodeScale. The Tracks element contains the information used by the playback device to decode the encoded media contained within the Clusters element of the Matroska file. As noted above, adaptive bitrate streaming systems in accordance with embodiments of the invention can support different streams encoded using different encoding parameters including but not limited to frame rate, and resolution. Therefore, the playback device can use the information contained within the Matroska container file's headers to configure the decoder every time a transition is made between encoded streams.

[0082] In many embodiments, the playback device does not retrieve the headers for all of the Matroska container files indexed in the top level index file. Instead, the playback device determines the stream(s) that will be utilized to initially commence playback and requests the headers from the corresponding Matroska container files. Depending upon the structure of the URIs contained within the top level index file, the playback device can either use information from the URIs or information from the headers of the Matroska container files to request byte ranges from the server that contain at least a portion of the index from relevant Matroska container files. The byte ranges can correspond to the entire index. The server provides the relevant byte ranges containing the index information to the playback device, and the playback device can use the index information to request the byte ranges of Cluster elements containing encoded media using this information. When the Cluster elements are received, the playback device can extract encoded media from the Block elements within the Cluster

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0716
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

element, and can decode and playback the media within the Block elements in accordance with their associated Timecode attributes.

[0083] In the illustrated embodiment, the playback device 200 requests sufficient index information from the HTTP server prior to the commencement of playback that the playback device can stream the entirety of each of the selected streams using the index information. In other embodiments, the playback device continuously retrieves index information as media is played back. In several embodiments, all of the index information for the lowest bitrate steam is requested prior to playback so that the index information for the lowest bitrate stream is available to the playback device in the event that streaming conditions deteriorate rapidly during playback.

SWITCHING BETWEEN STREAMS

[0084] The communications illustrated in FIG. 9a assume that the playback device continues to request media from the same streams (i.e. Matroska container files) throughout playback of the media. In reality, the streaming conditions experienced by the playback device are likely to change during the playback of the streaming media and the playback device can request media from alternative streams (i.e. different Matroska container files) to provide the best picture quality for the streaming conditions experienced by the playback device. In addition, the playback device may switch streams in order to perform a visual search (i.e. fast forward, rewind) trick play function that utilizes a trick play stream.

[0085] Communication between a playback device and a server when a playback device switches to a new stream in accordance with embodiments of the invention are illustrated in FIG. 9a. The communications illustrated in FIG. 9a assume that the index information for the new stream has not been previously requested by the playback device and that downloading of Cluster elements from the old stream proceeds while information is obtained concerning the Matroska container file containing the new stream. When the playback device 200 detects a change in streaming conditions,

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0717
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

determines that a higher bitrate stream can be utilized at the present streaming conditions, or receives a trick play instruction from a user, the playback device can use the top level index file to identify the URI for a more appropriate alternative stream to at least one of the video, audio, or subtitle streams from which the playback device is currently requesting encoded media. The playback device can save the information concerning the current stream(s) and can request the byte ranges of the headers for the Matroska container file(s) containing the new stream(s) using the parameters of the corresponding URIs. Caching the information in this way can be beneficial when the playback device attempts to adapt the bitrate of the stream downward or return to the stream following the completion of a visual search by the user. When the playback device experiences a reduction in available bandwidth or receives an instruction to resume normal playback, the playback device ideally will quickly switch between streams. Due to the reduced bandwidth experienced by the playback device, the playback device is unlikely to have additional bandwidth to request header and index information when there is congestion. Ideally, the playback device utilizes all available bandwidth to download already requested higher rate Cluster elements and uses locally cached index information to start requesting Cluster elements from Matroska container file(s) containing lower bitrate stream(s).

[0086] Byte ranges for index information for the Matroska container file(s) containing the new stream(s) can be requested from the HTTP server 202 in a manner similar to that outlined above with respect to FIG. 9a. In the case of trick play streams, multiple byte range HTTP requests can be used that request only the BlockGroup or Simple Block elements from a Cluster element that will be utilized by the playback device during the visual search of the encoded media. Typically, the number of BlockGroup or SimpleBlock elements that are not requested depends upon the rate of the visual search. As discussed above, the incorrect size attribute of the Cluster element downloaded via HTTP in this way can be corrected based upon the size of the data received by the playback device prior to passing the Cluster element to a decoder.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0718
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

The time stamps on the buffered frames can also be manipulated prior to providing the frames to the playback device's decoder to achieve accelerated playback of the buffered frames.

[0087] Once the index of the new stream is downloaded, the playback device can stop downloading of Cluster elements from the previous stream and can commence requesting the byte ranges of the appropriate Cluster elements from the Matroska container file(s) containing the new stream(s) from the HTTP server. As noted above, the encoding of the alternative streams so that corresponding Cluster elements within alternative streams and the trick play stream (i.e. Cluster elements containing the same portion of the source media encoded in accordance with different encoding parameters) start with the same Timecode element and an IDR frame facilitates the smooth transition from one stream to another.

[0088] When the playback device caches the header and the entire index for each stream that has been utilized in the playback of the media, the process of switching back to a previously used stream can be simplified. The playback device already has the header and index information for the Matroska file containing the previously utilized stream and the playback device can simply use this information to start requesting Cluster elements from the Matroska container file of the previously utilized stream via HTTP. Communication between a playback device and an HTTP server when switching back to a stream(s) for which the playback device has cached header and index information in accordance with an embodiment of the invention is illustrated in FIG. 9b. The process illustrated in FIG. 9b is ideally performed when adapting bitrate downwards, because a reduction in available resources can be exacerbated by a need to download index information in addition to media. The likelihood of interruption to playback is reduced by increasing the speed with which the playback device can switch between streams and reducing the amount of overhead data downloaded to achieve the switch.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0719
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0089] Although the present invention has been described in certain specific aspects, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that the present invention may be practiced otherwise than specifically described, including various changes in the implementation such as utilizing encoders and decoders that support features beyond those specified within a particular standard with which they comply, without departing from the scope and spirit of the present invention. Thus, embodiments of the present invention should be considered in all respects as illustrative and not restrictive.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0720
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS CLAIMED:

1. A method of performing visual search of encoded media using a trick play stream, where the media is encoded as a plurality of alternative streams used during normal playback and a trick play stream used during visual search, the method comprising:

requesting and buffering portions of video from at least one of the alternative streams using a playback device;

decoding the buffered portions of video using a decoder on the playback device;

receiving at least one user instruction directing the playback device to perform a visual search of the media;

requesting and buffering potions of video from the trick play stream using the playback device; and

decoding the buffered portions of the trick play stream using a decoder on the playback device.

2. The method of claim 1, further comprising measuring the current streaming conditions by measuring the time taken to receive requested portions of a stream from the time at which the portions were requested.

3. The method of claim 2, wherein requesting and buffering portions of video from at least one of the alternative streams using a playback device further comprises requesting and buffering portions of video from at least one of the alternative streams based upon the bitrates of the alternative streams and the measured streaming conditions using a playback device.

4. The method of claim 1, wherein each frame of the trick play stream is an intra frame starting a closed group of pictures (GOP).

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0721
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

5. The method of claim 1, wherein:

the at least one user instruction specifies the rate and direction of the visual search; and

requesting and buffering potions of video from the trick play stream using the playback device further comprises requesting frames from the trick play stream based upon the rate and direction of the visual search specified by the at least one user instruction.

6. The method of claim 1, wherein:

the alternative streams and the trick play stream are stored in separate container files; and

requesting portions of video from a stream further comprising requesting portions of files from remote servers via Hypertext Transfer Protocol (HTTP) byte range requests using the playback device.

7. The method of claim 1, wherein the alternative streams and the trick play stream are stored in separate Extensible Binary Markup Language (EBML) container files.

8. The method of claim 7, wherein:

each of the EBML container files comprises a plurality of Cluster elements, where each Cluster element contains a portion of encoded video; and

the portion of encoded video in each of the Cluster elements commences with an intra frame.

9. The method of claim 8, wherein each Cluster element includes at least one closed group of pictures

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0722
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

10. The method of claim 8, wherein the portions of encoded video in each of the Cluster elements of the alternative streams have the same duration.

11. The method of claim 10, wherein the portions of encoded video in each of the Cluster elements of the alternative streams have a 2 second duration and each of the Cluster element of the trick play stream include 64 frames.

12. The method of claim 8, wherein each Cluster element contains a time code and each encoded frame of the portion of encoded video contained within the Cluster element is contained within a separate element.

13. The method of claim 12, wherein the first element containing a frame in the Cluster element contains the intra frame.

14. The method of claim 13, wherein the first element containing a frame is a BlockGroup element that contains a Block element, which specifies the time code attribute of the intra frame relative to the time code of the Cluster element.

15. The method of claim 12, wherein all of the elements that contain frames in the Cluster elements of the EBML container file containing the trick play stream contain intra frames.

16. The method of claim 8, wherein the EBML container file containing the trick play stream includes at least one modified Cues element containing an index that references Clusters elements containing portions of encoded video within the EBML container file and the elements containing frames within the referenced Clusters elements.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0723
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

17. The method of claim 16, further comprising:

retrieving the modified Cues element using the playback device; and

identifying frames to retrieve from the EBML container file based upon a visual search rate using the Cues element;

wherein requesting and buffering potions of video from the trick play stream using the playback device further comprises requesting and buffering elements that contain frames from the EBML container file containing the trick play stream using the playback device.

18. The method of claim 17, further comprising updating the size attribute of Cluster elements retrieved by the playback device.

19. The method of claim 1, further comprising modifying the time stamps of the buffered portions of the trick play stream prior to providing the portions of the trick play stream to the decoder for decoding.

20. The method of claim 1, further comprising retrieving a top level index file using the playback device that identifies the alternative streams used during normal playback of the encoded video and identifies the trick play stream used during visual search of the encoded video.

21. A playback device configured to perform visual search of encoded media using a trick play stream, where the media is encoded as a plurality of alternative streams used during normal playback and a trick play stream used during visual search, the playback device comprising:

a processor configured to communicate with memory, where the processor is configured by a playback client application stored in the memory;

wherein the playback client application configures the processor to:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0724
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

request portions of video from at least one of the alternative streams from a remote server and buffer the requested portions of video in memory;

decode the buffered portions of video;

receive at least one user instruction directing the playback device to perform a visual search of the media;

request potions of video from the trick play stream from a remote server and buffer requested portions of video from the trick play stream; and

decode the buffered portions of the trick play stream.

22. A method of encoding source video as a plurality of alternative streams and a trick play stream using a source encoder, the method comprising:

repeatedly selecting a portion of the source video using the source encoder;

transcoding the selected portion of the source video into a plurality of alternative portions of encoded video using the source encoder, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed group of pictures (GOP);

transcoding the selected portion of the source video into a portion of trick play stream video for inclusion in a trick play stream using the source encoder, where the portion of trick play stream video is encoded at a lower frame rate and a lower resolution than the source video;

writing each of the alternative portions of encoded video to a separate container file using the source encoder;

writing the trick play stream to a separate container file using the source encoder; and

adding an entry to an index that identifies the location of each frame in the portion of trick play stream video within the container file containing the trick play stream.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0725
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

23. The method of claim 22, wherein transcoding a selected portion of the source video into a portion of trick play stream video further comprises transcoding the selected portion so that each frame is an IDR frame.

24. The method of claim 22, wherein the selected portions of source video are selected based upon their duration.

25. The method of claim 24, wherein the selected portions of source video each have a duration of two seconds.

26. The method of claim 22, wherein the container files containing the alternative streams and the trick play stream are Extensible Binary Markup Language (EBML) container files.

27. The method of claim 26, wherein:

the element of the EBML container file to which each alternative portion of encoded video is written is a Cluster element; and

each Cluster element to which each of the alternative portions of encoded video is written is assigned the same time code.

28. The method of claim 26, wherein the element of the EBML container file to which the portion of trick play stream video is written is a Cluster element.

29. The method of claim 28, wherein the frames of each portion of encoded video are contained within elements within the Cluster element.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0726
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

30. The method of claim 29, wherein the first element that contains a frame in the Cluster element contains the intra frame.

31. The method of claim 30, wherein the first element that contains a frame is a BlockGroup element that contains a Block element, which specifies the time code attribute of the intra frame relative to the time code of the Cluster element.

32. The method of claim 29, wherein each frame in the portion of trick play stream video is an intra frame.

33. The method of claim 32, wherein each element containing a frame in the Cluster element is a BlockGroup element containing the portion of trick play stream video contains a Block element, which specifies the time code attribute of the intra frame relative to the time code of the Cluster element.

34. The method of claim 33, wherein adding an entry to an index that identifies the location of each frame in the portion of trick play stream video within the container file containing the trick play stream, further comprises adding the location of each BlockGroup element in the Cluster element containing the portion of trick play stream video to the index for the EBML container file containing the trick play stream.

35. The method of claim 34, further comprising packing the index for the EBML container file containing the trick play stream into the EBML container file containing the trick play stream.

36. The method of claim 35, wherein each index comprises a Cues element.

37. The method of claim 36, wherein:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0727
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

each Cues element includes a CuePoint element that points to the location of the Cluster element containing the portion of trick play stream video within the EBML file containing the trick play stream; and

each CuePoint element comprises a plurality of elements that point to the BlockGroup elements within the Cluster element pointed to by the CuePoint element.

38. The method of claim 26, further comprising writing an element to each of the EBML container files specifying a set of encoding parameters.

39. The method of claim 38, wherein the set of encoding parameters includes at least one parameter selected from the group consisting of frame rate, frame height, frame width, sample, aspect ratio, maximum bitrate, and minimum buffer size.

40. The method of claim 22, further comprising creating a top level index file that identifies each of the container files containing the alternative streams used during normal playback of the encoded video and identifies the container file that includes the trick play stream used during visual search of the encoded video.

41. A source encoder configured to encode source video as a plurality of alternative streams and a trick play stream, the source encoder comprising:

a processor configured to communicate with memory, the processor configured by a source encoder application stored in memory;

wherein the source encoder application configures the processor to:

repeatedly select a portion of the source video using the source encoder;

transcode the selected portion of the source video into a plurality of alternative portions of encoded video, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0728
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

closed group of pictures (GOP);

transcode the selected portion of the source video into a portion of trick play stream video for inclusion in a trick play stream using the source encoder, where the portion of trick play stream video is encoded at a lower frame rate and a lower resolution than the source video;

write each of the alternative portions of encoded video to a separate container file using the source encoder;

write the trick play stream to a separate container file using the source encoder; and

add an entry to an index that identifies the location of each frame in the portion of trick play stream video within the container file containing the trick play stream.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0729
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

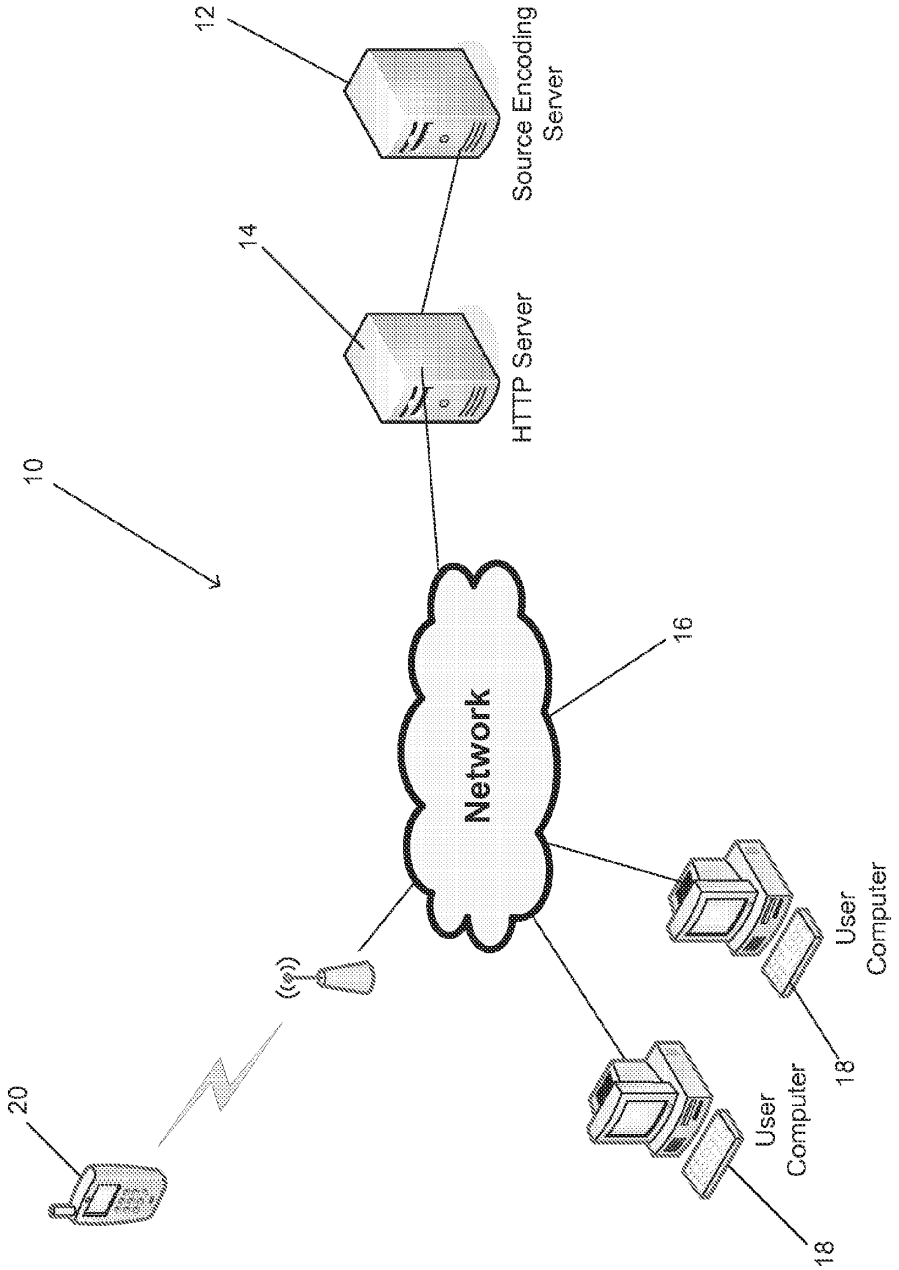


FIG. 1

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0730
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

SMIL File
30
MKVFile (Stream 1)
32
MKVFile (Stream 2)
32
MKVFile (Stream 3)
32
MKVFile (Trick Play)
32

FIG. 2

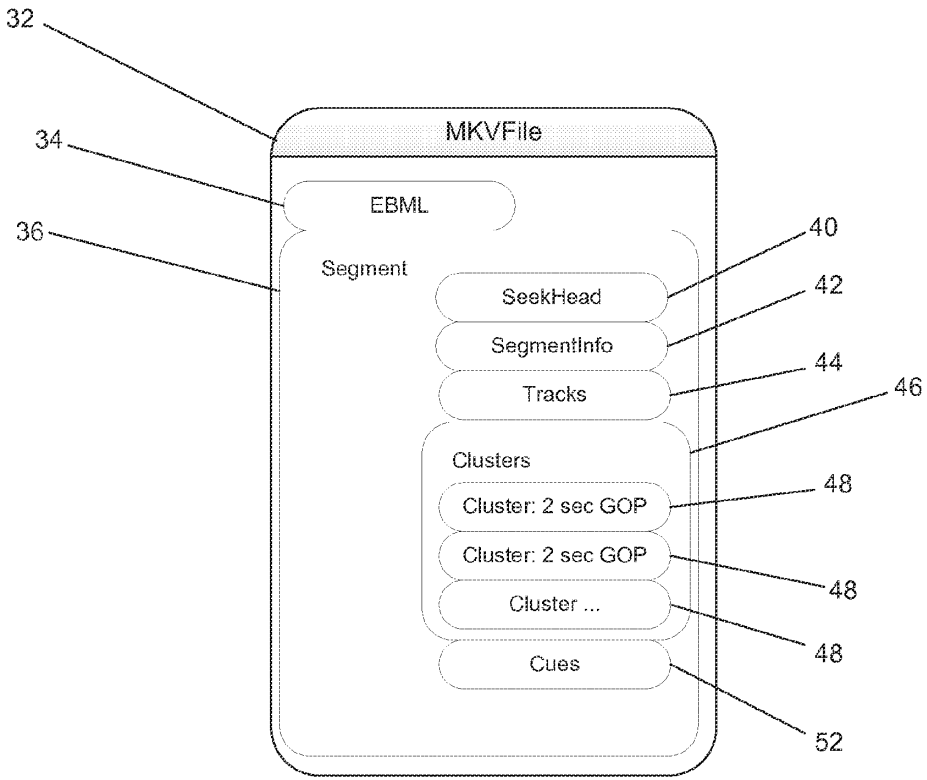

32
34
36
MKVFile
EBML
Segment
SeekHead
SegmentInfo
Tracks
Clusters
Cluster: 2 sec GOP
Cluster: 2 sec GOP
Cluster ...
Cues
40
42
44
46
48
48
48
52

FIG. 3

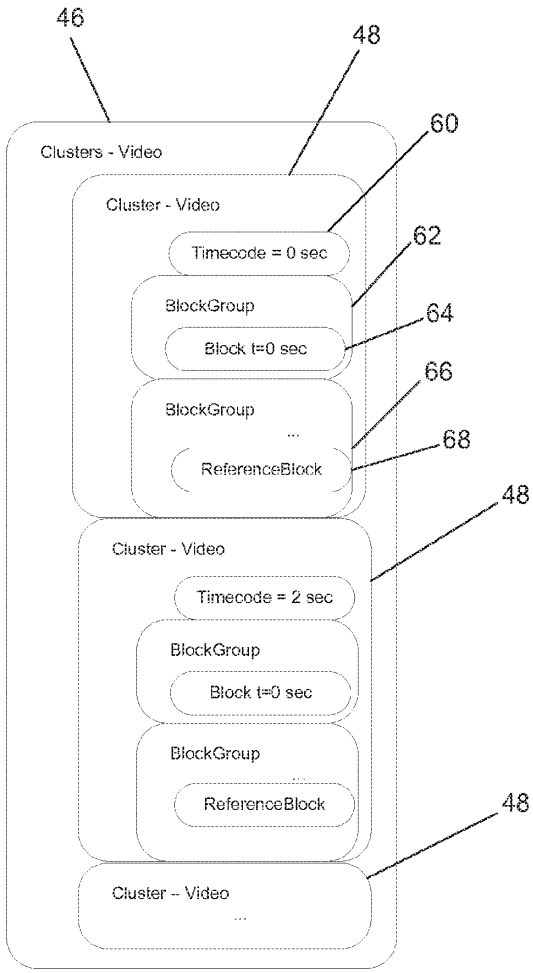

46
48
Clusters - Video
60
Cluster - Video
Timecode = 0 sec
62
BlockGroup
64
Block t=0 sec
66
BlockGroup
...
68
ReferenceBlock
48
Cluster - Video
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
...
ReferenceBlock
48
Cluster – Video
...

FIG. 4a

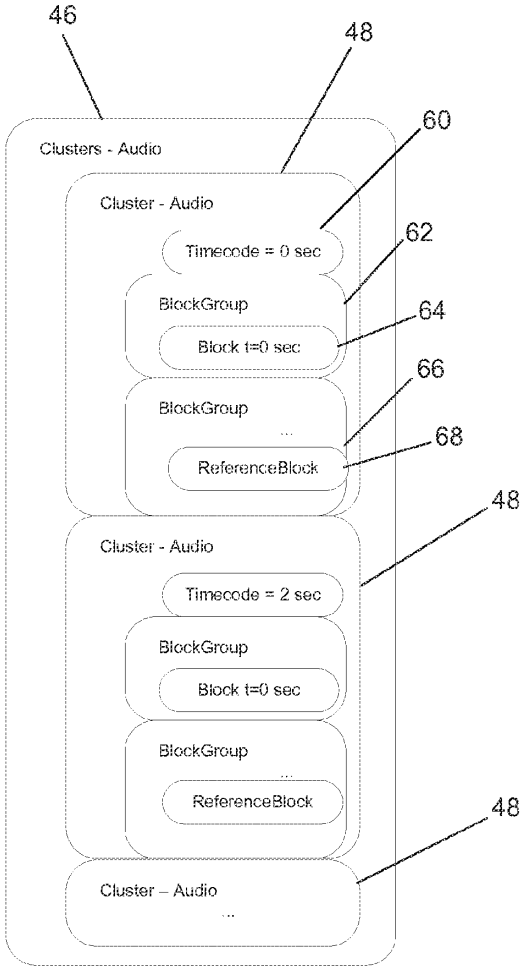

46
48
Clusters - Audio
60
Cluster - Audio
Timecode = 0 sec
62
BlockGroup
64
Block t=0 sec
66
BlockGroup
...
68
ReferenceBlock
48
Cluster - Audio
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
...
ReferenceBlock
48
Cluster – Audio
...

FIG. 4b

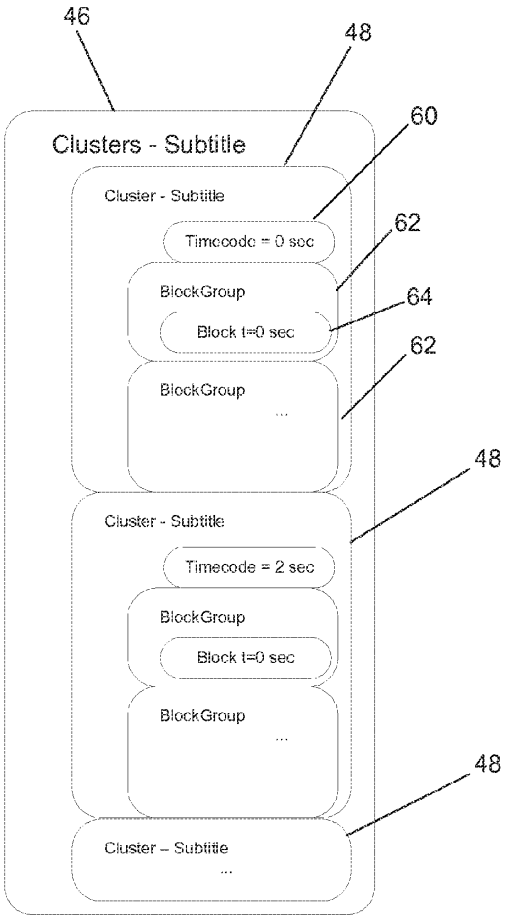

46
48
Clusters - Subtitle
60
Cluster - Subtitle
Timecode = 0 sec
62
BlockGroup
Block t=0 sec
64
62
BlockGroup
...
48
Cluster - Subtitle
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
...
48
Cluster – Subtitle
...

FIG. 4c

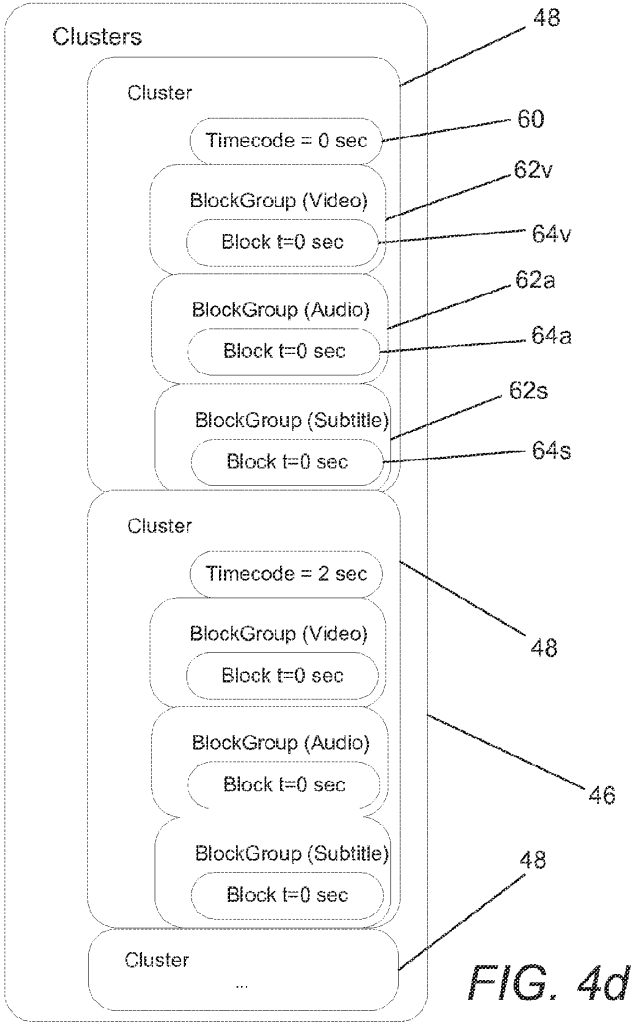

Clusters
48
Cluster
Timecode = 0 sec
60
BlockGroup (Video)
62v
Block t=0 sec
64v
BlockGroup (Audio)
62a
Block t=0 sec
64a
BlockGroup (Subtitle)
62s
Block t=0 sec
64s
Cluster
Timecode = 2 sec
48
BlockGroup (Video)
Block t=0 sec
BlockGroup (Audio)
Block t=0 sec
46
BlockGroup (Subtitle)
Block t=0 sec
48
Cluster
...

FIG. 4d

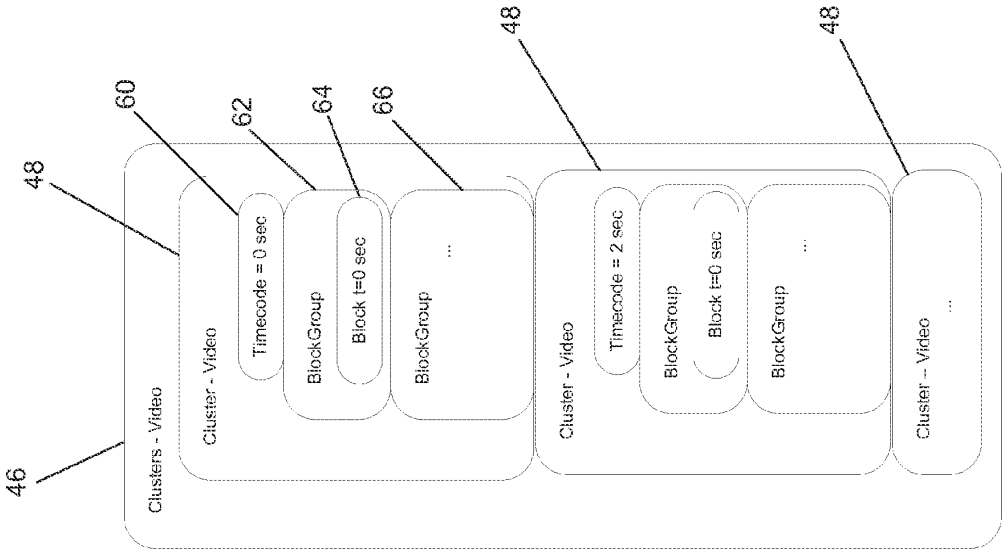

46
48
60
62
64
66
48
48
Clusters - Video
Cluster - Video
Timecode = 0 sec
BlockGroup
Block t=0 sec
BlockGroup
...
Cluster - Video
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
...
Cluster - Video
...

FIG. 4e

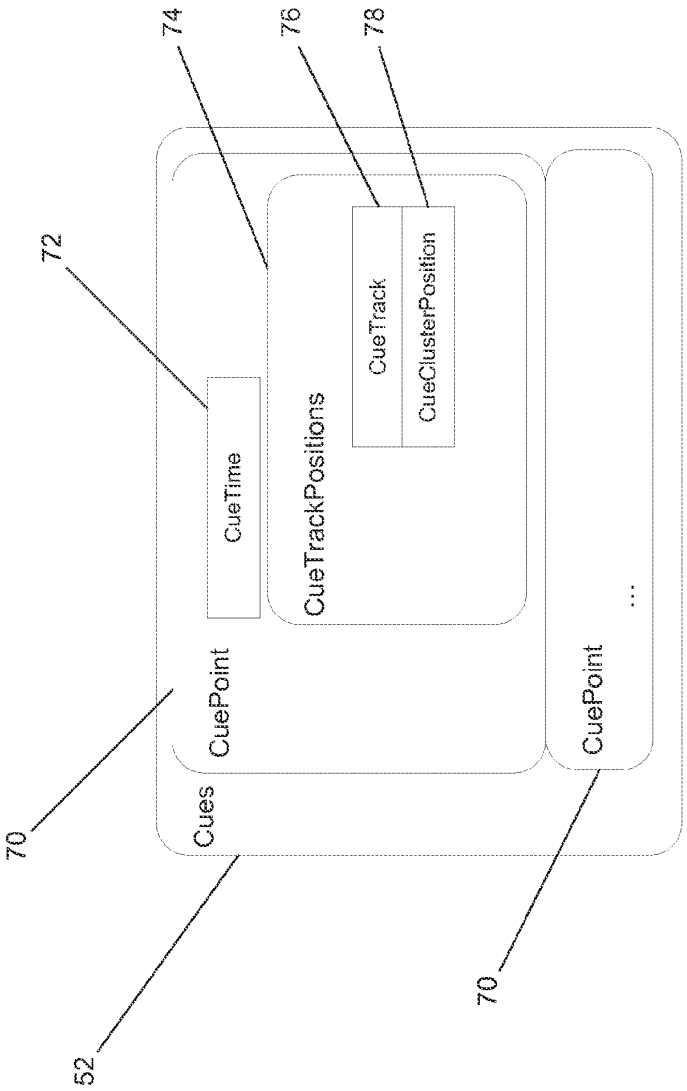

74
76
78
72
CueTime
CueTrackPositions
CueTrack
CueClusterPosition
CuePoint
...
CuePoint
Cues
70
52
70

FIG. 5

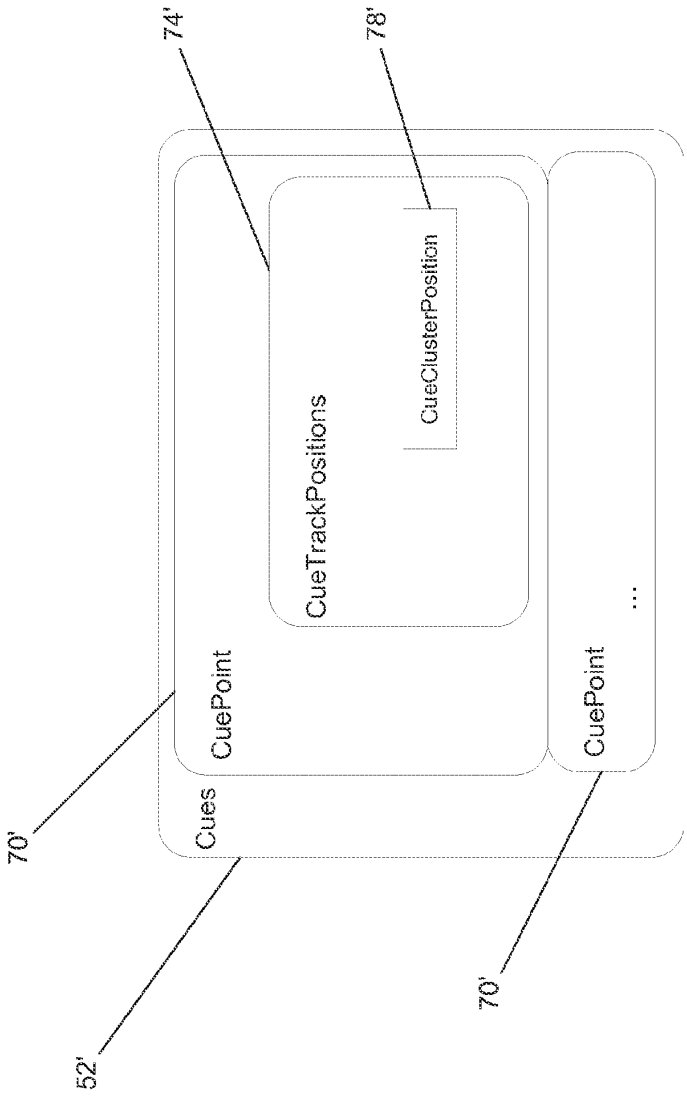

74'
78'
CueClusterPosition
CueTrackPositions
CuePoint
...
CuePoint
Cues
70'
52'
70'

FIG. 5a

FIG. 5b

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0738
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

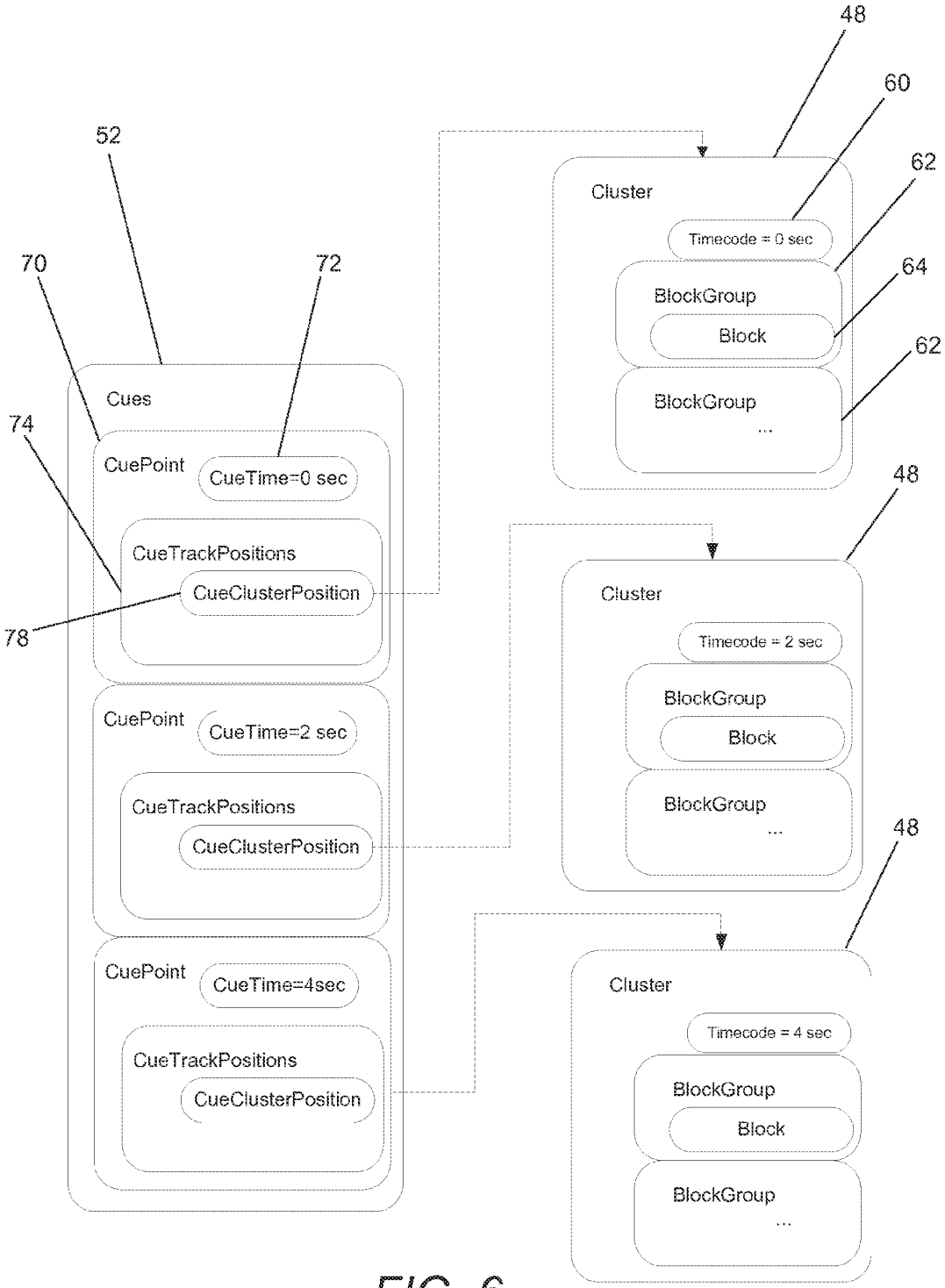

48
60
52
62
Cluster
Timecode = 0 sec
70
72
64
BlockGroup
Block
62
Cues
BlockGroup
...
74
CuePoint
CueTime=0 sec
48
CueTrackPositions
CueClusterPosition
78
Cluster
Timecode = 2 sec
CuePoint
CueTime=2 sec
BlockGroup
Block
CueTrackPositions
BlockGroup
...
CueClusterPosition
48
CuePoint
CueTime=4sec
Cluster
Timecode = 4 sec
CueTrackPositions
CueClusterPosition
BlockGroup
Block
BlockGroup
...

FIG. 6

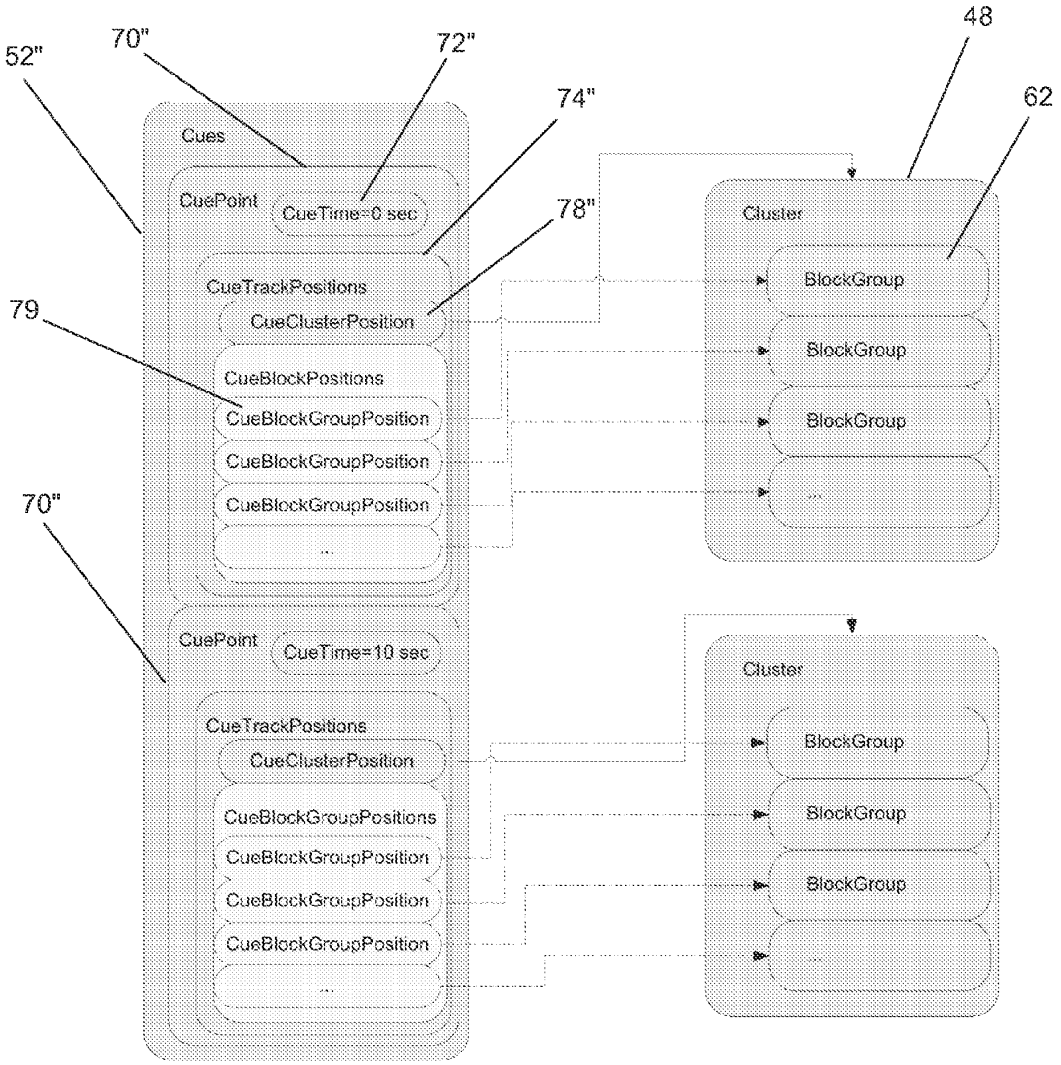

52"
70"
72"
74"
48
62
78"
79
70"
Cues
CuePoint
CueTime=0 sec
CueTrackPositions
CueClusterPosition
CueBlockPositions
CueBlockGroupPosition
CueBlockGroupPosition
CueBlockGroupPosition
...
CuePoint
CueTime=10 sec
CueTrackPositions
CueClusterPosition
CueBlockGroupPositions
CueBlockGroupPosition
CueBlockGroupPosition
CueBlockGroupPosition
...
Cluster
BlockGroup
BlockGroup
BlockGroup
...
Cluster
BlockGroup
BlockGroup
BlockGroup
...

FIG. 7

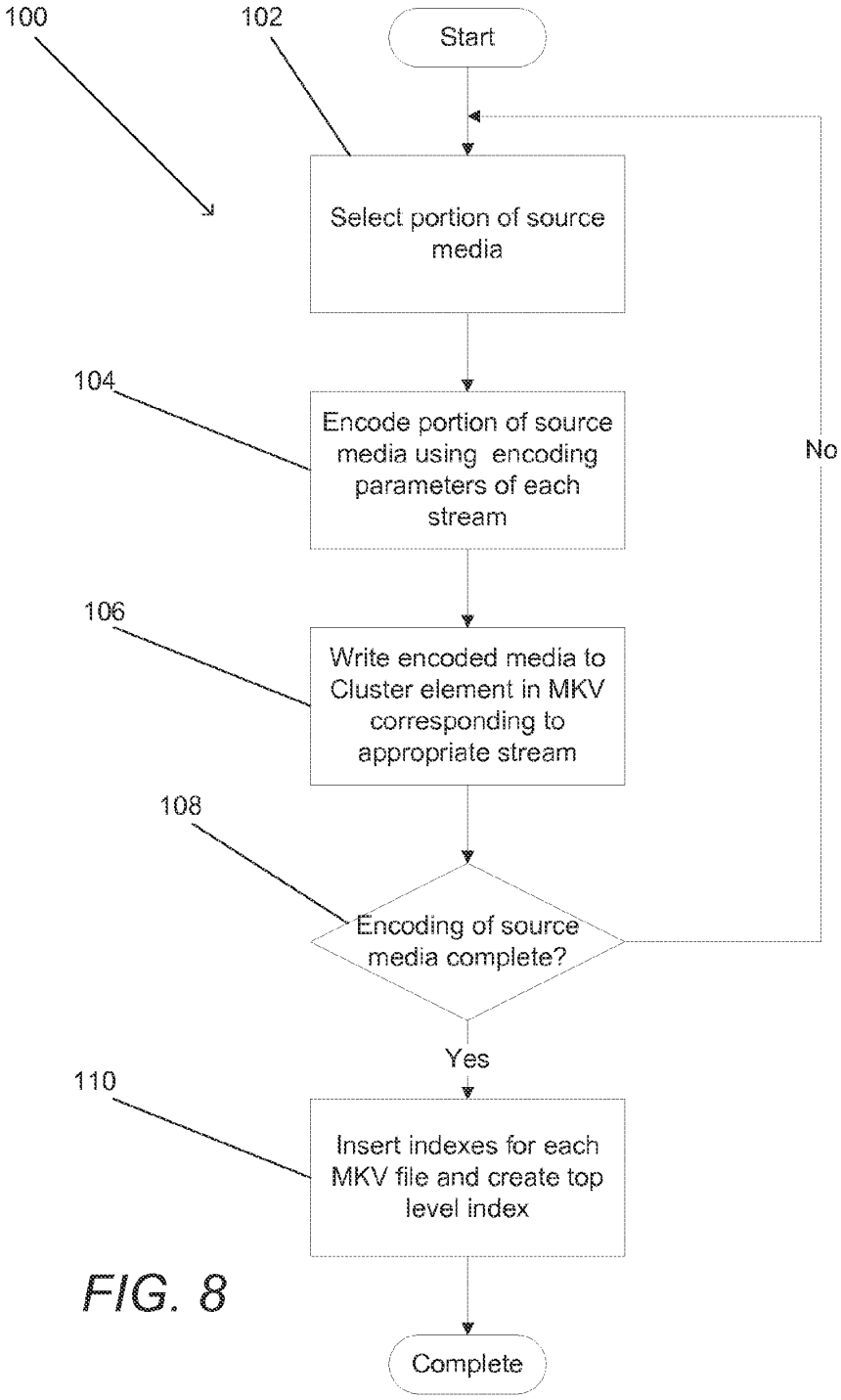

100
102
Start
Select portion of source media
104
Encode portion of source media using encoding parameters of each stream
106
Write encoded media to Cluster element in MKV corresponding to appropriate stream
108
Encoding of source media complete?
No
Yes
110
Insert indexes for each MKV file and create top level index
Complete

FIG. 8

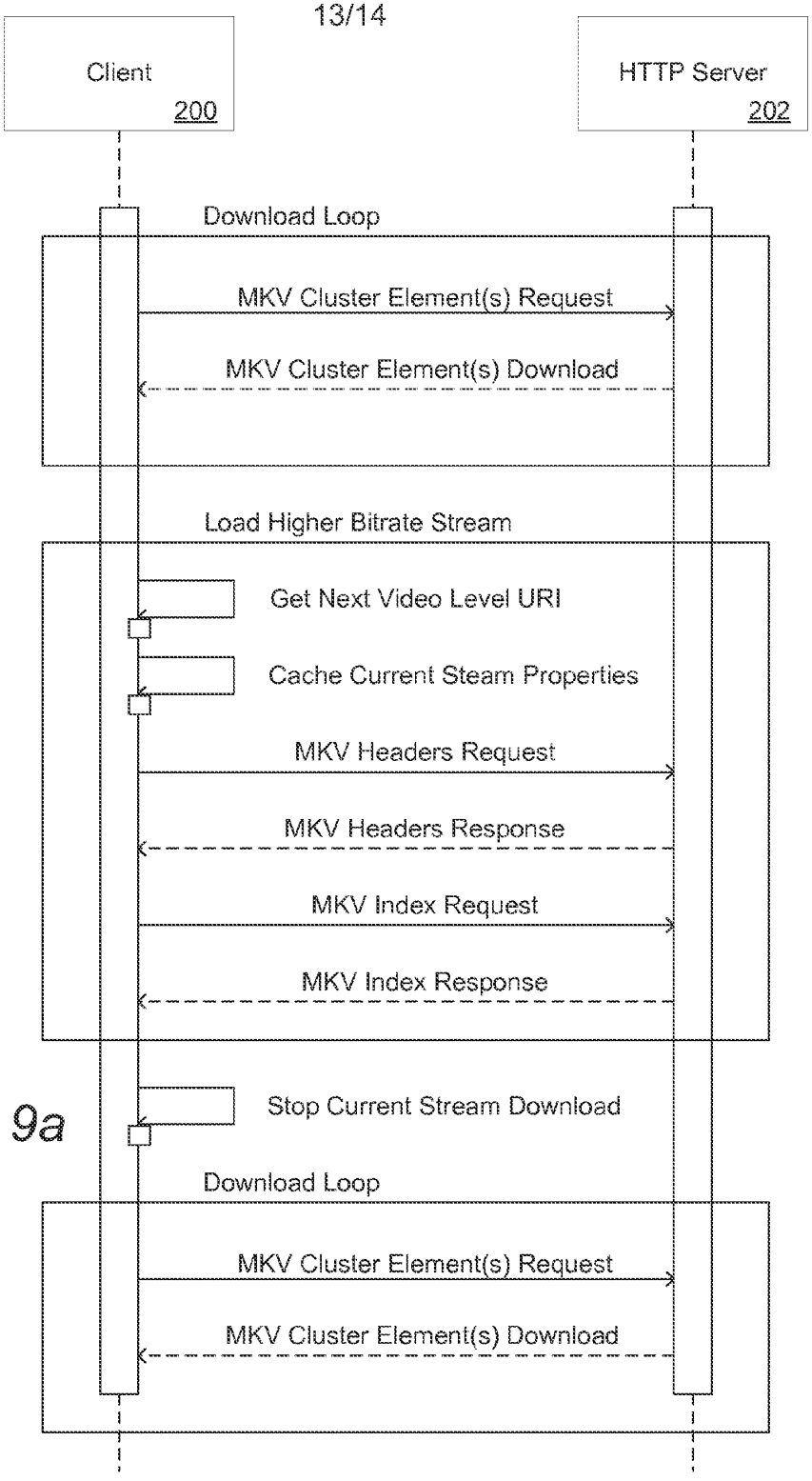

Client
200
HTTP Server
202
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download
Load Higher Bitrate Stream
Get Next Video Level URI
Cache Current Steam Properties
MKV Headers Request
MKV Headers Response
MKV Index Request
MKV Index Response
Stop Current Stream Download
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 9a

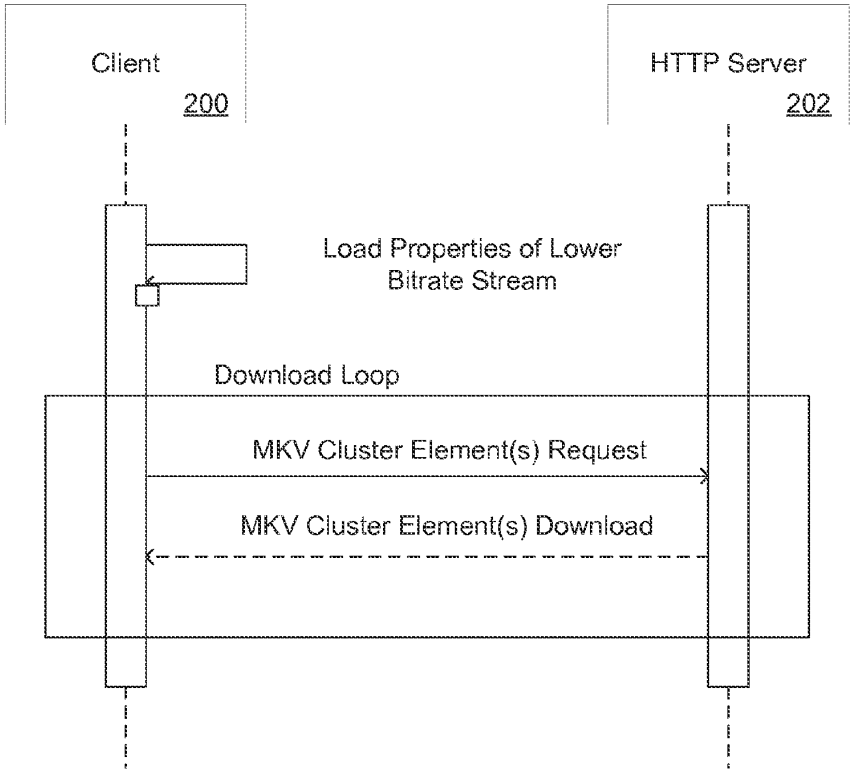

Client
200
HTTP Server
202
Load Properties of Lower Bitrate Stream
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 9b

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 11/67243

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H04N 7/12 (2012.01)
USPC - 348/404.1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
USPC: 348/404.1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC:348/408.1; 375/E7.198; 709/231; 725/62; 725/86

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PUBWEST (PGPB, USPT, EPAB, JPAB), Google Scholar, Google Patents; Search Terms: Encoded media, multimedia, video, trick-play, trickplay, visual-search, cluster element, Intra frame, group of pictures, GOP, Extensible Binary Markup Language, EBML, Hypertext Transfer Protocol, HTTP, alternate stream$, multiple bitrate$, multiple stream$

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009/0169181 A1 (Priyadarshi et al.) 02 July 2009 (02.07.2009) entire document especially Abstract, para [0005], para [0039], para [0057]-[0062], para [0068]-[0075]	1-41
Y	US 2003/0231863 A1 (Eerenberg et al.) 18 December 2003 (18.12.2003) entire document especially Abstract, para [0016], para [0044], para [0155]-[0157].	1-41
Y	US 2009/0132599 A1 (Soroushian et al.) 21 May 2009 (21.05.2009) entire document especially Abstract, para [0028]-[0030], para [0045]-[0049]).	7-18, 26-39
A	US 2003/0231867 A1 (Gates et al.) 18 December 2003 (18.12.2003) entire document especially Abstract, para [0007]-[0008], para [0072]	1-41

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 April 2012 (24.04.2012)	08 MAY 2012
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 July 2012 (12.07.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/094171 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2011/066927

(22) International Filing Date:
22 December 2011 (22.12.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/430,110	5 January 2011 (05.01.2011)	US
13/221,794	30 August 2011 (30.08.2011)	US
13/221,682	30 August 2011 (30.08.2011)	US

(71) Applicant *(for all designated States except US)*: **DIVX, LLC.** [US/US]; 2830 De La Cruz Boulevard, Santa Clara, CA 95050 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BRANESS, Jason** [US/US]; 15885 Concord Ridge Terrace, San Diego, CA 92127 (US). **VAN DER SCHAAR, Auke, Sjoerd** [NL/US]; 10775 Wilkins Ave., Apt. 102, Los Angeles, CA 90024 (US). **SOROUSHIAN, Kamrosh** [US/US]; 3758 Brookshire Street, San Diego, CA 92111 (US).

(74) Agent: **BAILEY, David, J.**; Knobbe, Pomeroy, Peck & Bailey LLP, 2400 E. Katella Ave., Suite 1050, Anaheim, CA 92806 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: ADAPTIVE BITRATE STREAMING OF MEDIA STORED IN MATROSKA CONTAINER FILES USING HYPERTEXT TRANSFER PROTOCOL

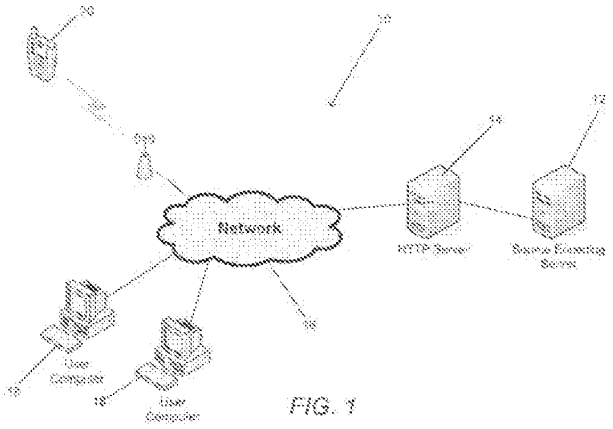


FIG. 1

(57) Abstract: Systems and methods for adaptive bitrate streaming of media stored in Matroska container files utilizing Hypertext Transfer Protocol (HTTP) are disclosed. In one embodiment, a processor requests portions of files from a remote server and retrieves top level index data that identifies EBML container files and describes a maximum bitrate of the alternative streams contained within the EBML container files. The top level index data is parsed to obtain information identifying the EBML container files, requests a portion of at least one of the EBML container files that contains the at least one element that specifies the encoding parameters of the stream contained within the EBML container file, and requests portions of a first EBML container file that includes elements that contain portions of encoded video, where the selection is based upon the measured streaming conditions and the description of the bitrate of the alternative stream contained within the top level data.

WO 2012/094171 A1

ADAPTIVE BITRATE STREAMING OF MEDIA STORED IN MATROSKA CONTAINER FILES USING HYPERTEXT TRANSFER PROTOCOL

FIELD OF THE INVENTION

[0001] The present invention generally relates to adaptive streaming and more specifically to adaptive bitrate streaming of encoded media contained within Matroska container files using Hypertext Transfer Protocol.

BACKGROUND

[0002] The term streaming media describes the playback of media on a playback device, where the media is stored on a server and continuously sent to the playback device over a network during playback. Typically, the playback device stores a sufficient quantity of media in a buffer at any given time during playback to prevent disruption of playback due to the playback device completing playback of all the buffered media prior to receipt of the next portion of media. Adaptive bit rate streaming or adaptive streaming involves detecting the present streaming conditions (e.g. the user's network bandwidth and CPU capacity) in real time and adjusting the quality of the streamed media accordingly. Typically, the source media is encoded at multiple bit rates and the playback device or client switches between streaming the different encodings depending on available resources.

[0003] Adaptive streaming solutions typically utilize either Hypertext Transfer Protocol (HTTP), published by the Internet Engineering Task Force and the World Wide Web Consortium as RFC 2616, or Real Time Streaming Protocol (RTSP), published by the Internet Engineering Task Force as RFC 2326, to stream media between a server and a playback device. HTTP is a stateless protocol that enables a playback device to request a byte range within a file. HTTP is described as stateless, because the server is not required to record information concerning the state of the playback device requesting information or the byte ranges requested by the playback device in order to respond to requests received from the playback device. RTSP is a network control protocol used to control streaming media servers. Playback devices issue control commands, such as "play" and "pause", to the server streaming the media to control the playback of media files. When RTSP is utilized, the media server records the state of each client device and determines the media to stream based upon the instructions received from the client devices and the client's state.

[0004] In adaptive streaming systems, the source media is typically stored on a media server as a top level index file pointing to a number of alternate streams that contain the actual video and audio data. Each stream is typically stored in one or more container files. Different adaptive streaming solutions typically utilize different index and media containers. The Synchronized Multimedia Integration Language (SMIL) developed by the World Wide Web Consortium is utilized to create indexes in several adaptive streaming solutions including IIS Smooth Streaming developed by Microsoft Corporation of Redmond, Washington, and Flash Dynamic Streaming developed by Adobe Systems Incorporated of San Jose, California. HTTP Adaptive Bitrate Streaming developed by Apple Computer Incorporated of Cupertino, California implements index files using an extended M3U playlist file (M3U8), which is a text file containing a list of URIs that typically identify a media container file. The most commonly used media container formats are the MP4 container format specified in MPEG-4 Part 14 (i.e. ISO/IEC 14496-14) and the MPEG transport stream (TS) container specified in MPEG-2 Part 1 (i.e. ISO/IEC Standard 13818-1). The MP4 container format is utilized in IIS Smooth Streaming and Flash Dynamic Streaming. The TS container is used in HTTP Adaptive Bitrate Streaming.

[0005] The Matroska container is a media container developed as an open standard project by the Matroska non-profit organization of Aussonne, France. The Matroska container is based upon Extensible Binary Meta Language (EBML), which is a binary derivative of the Extensible Markup Language (XML). Decoding of the Matroska container is supported by many consumer electronics (CE) devices. The DivX Plus file format developed by DivX, LLC of San Diego, California utilizes an extension of the Matroska container format (i.e. is based upon the Matroska container format, but includes elements that are not specified within the Matroska format).

SUMMARY OF THE INVENTION

[0006] Systems and methods for adaptive bitrate streaming of media stored in Matroska container files utilizing Hypertext Transfer Potocol (HTTP) in accordance with embodiments of the invention are disclosed. In one embodiment, a processor configured, via a client application, to request portions of files from a remote server. In addition, the client application further configures the processor to retrieve top level index data that identifies a plurality of EBML container files and describes at least a maximum bitrate of the alternative streams contained within the EBML container files, parse the top level index data to obtain information identifying

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0747
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the plurality of EBML container files, request a portion of at least one of the EBML container files that contains the at least one element that specifies the encoding parameters of the stream contained within the EBML container file, retrieve an index that references each element containing portions of encoded video within at least one of the EBML container files, utilize the index to request portions of a first EBML container file that includes elements that contain portions of encoded video, receive and buffer the requested elements, decode the encoded video contained within the buffered elements utilizing the encoding parameters, measure current streaming conditions, and select another of the EBML container files from which to retrieve elements containing portions of encoded video for decoding, where the selection is based upon the measured streaming conditions and the description of the bitrate of the alternative stream contained within the top level data.

[0007] A further embodiment includes retrieving top level index data that identifies each of the plurality of EBML container files and describes at least the maximum bitrate of each of the alternative streams contained within the EBML container files, parsing the top level index data to obtain information identifying the plurality of EBML container files, requesting a portion of at least one of the EBML container files that contains the at least one element that specifies the encoding parameters of the stream contained within the EBML container file, retrieving an index that references each element containing portions of encoded video within at least one of the EBML container files, utilizing the index to request portions of a first EBML container file that includes elements that contain portions of encoded video, receiving and buffering the requested elements, decoding the encoded video contained within the buffered elements utilizing the encoding parameters, measuring current streaming conditions, and selecting another of the EBML container files from which to retrieve elements containing portions of encoded video for decoding, where the selection is based upon the measured streaming conditions and the description of the bitrate of the alternative stream contained within the top level index data.

[0008] Another embodiment of the invention includes a processor configured via a source encoding application to ingest at least one multimedia file containing a source video. In addition, the source encoding application further configures the processor to select a portion of the source video, transcode the selected portion of the source video into a plurality of alternative portions of encoded video, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed Group of Pictures

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0748
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

(GOP),write each of the alternative portions of encoded video to an element of a different EBML container file, where each element is located within an EBML container file that also includes another element that indicates the encoding parameters used to encode the alternative portion of encoded video, and add an entry to at least one index that identifies the location of the element containing one of the alternative portions of encoded video within each of the EBML container files.

[0009] Another further embodiment includes repeatedly selecting a portion of the source video using the source encoder, transcoding the selected portion of the source video into a plurality of alternative portions of encoded video using the source encoder, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed Group of Pictures (GOP), writing each of the alternative portions of encoded video to an element of a different EBML container file using the source encoder, where each element is located within an EBML container file that also includes another element containing a set of encoding parameters corresponding to the encoding parameters used to encode the portion of video, and adding an entry to at least one index that identifies the location of the element containing one of the alternative portions of encoded video within each of the EBML container files.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. 1 is a network diagram of an adaptive bitrate streaming system in accordance with an embodiment of the invention.

[0011] FIG. 2 conceptually illustrates a top level index file and Matroska container files generated by the encoding of source media in accordance with embodiments of the invention.

[0012] FIG. 3 conceptually illustrates a specialized Matroska container file incorporating a modified Cues element in accordance with an embodiment of the invention.

[0013] FIGS. 4a – 4c conceptually illustrate the insertion of different types of media into the Clusters element of a Matroska container file subject to various constrains that facilitate adaptive bitrate streaming in accordance with embodiments of the invention.

[0014] FIG. 4d conceptually illustrates the multiplexing of different types of media into the Clusters element of a Matroska container file subject to various constraints that facilitate adaptive bitrate streaming in accordance with an embodiment of the invention.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0749
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0015] FIG. 4e conceptually illustrates the inclusion of a trick play track into the Clusters element of a Matroska container file subject to various constraints that facilitate adaptive bitrate streaming in accordance with an embodiment of the invention.

[0016] FIG. 5 conceptually illustrates a modified Cues element of a specialized Matroska container file, where the Cues element includes information enabling the retrieval of Cluster elements using HTTP byte range requests in accordance with an embodiment of the invention.

[0017] FIG. 5a conceptually illustrates a modified Cues element of a specialized Matroska container file in accordance with an embodiment of the invention, where the Cues element is similar to the Cues element shown in FIG. 5 with the exception that attributes that are not utilized during adaptive bitrate streaming are removed.

[0018] FIG. 6 conceptually illustrates the indexing of Cluster elements within a specialized Matroska container file utilizing modified CuePoint elements within the container file in accordance with embodiments of the invention.

[0019] FIG. 7 is a flow chart illustrating a process for encoding source media for adaptive bitrate streaming in accordance with an embodiment of the invention.

[0020] FIG. 8 conceptually illustrates communication between a playback device and an HTTP server associated with the commencement of streaming of encoded media contained within Matroska container files indexed by a top level index file in accordance with an embodiment of the invention.

[0021] FIGS. 9a and 9b conceptually illustrate communication between a playback device and an HTTP server associated with switching between streams in response to the streaming conditions experienced by the playback device and depending upon the index information available to the playback device prior to the decision to switch streams in accordance with embodiments of the invention.

DETAILED DISCLOSURE OF THE INVENTION

[0022] Turning now to the drawings, systems and methods for adaptive bitrate streaming of media stored in Matroska container files utilizing Hypertext Transfer Potocol (HTTP) in accordance with embodiments of the invention are illustrated. In a number of embodiments, source media is encoded as a number of alternative streams. Each stream is stored in a Matroska (MKV) container file. In many embodiments, the Matroska container file is a specialized

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0750
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Matroska container file in that the manner in which the media in each stream is encoded and stored within the container is constrained to improve streaming performance. In several embodiments, the Matroska container file is further specialized in that additional index elements (i.e. elements that are not specified as part of the Matroska container format) can be included within the file to facilitate the retrieval of desired media during adaptive bitrate streaming. In several embodiments, each stream (i.e. audio, video, or subtitle) is stored within a separate Matroska container file. In other embodiments, an encoded video stream is multiplexed with one or more encoded audio, and/or subtitle streams in each Matroska container file. A top level index file containing an index to the streams contained within each of the container files is also generated to enable adaptive bitrate streaming of the encoded media. In many embodiments, the top level index file is a Synchronized Multimedia Integration Language (SMIL) file containing URIs for each of the Matroska container files. In other embodiments, any of a variety of file formats can be utilized in the generation of the top level index file.

[0023] The performance of an adaptive bitstrate streaming system in accordance with embodiments of the invention can be significantly enhanced by encoding each portion of the source video at each bit rate in such a way that the portion of video is encoded in each stream as a single (or at least one) closed group of pictures (GOP) starting with an Instantaneous Decoder Refresh (IDR) frame. The GOP for each stream can then be stored as a Cluster element within the Matroska container file for the stream. In this way, the playback device can switch between streams at the completion of the playback of a Cluster and, irrespective of the stream from which a Cluster is obtained the first frame in the Cluster will be an IDR frame and can be decoded without reference to any encoded media other than the encoded media contained within the Cluster element. In many embodiments, the sections of the source video that are encoded as GOPs are all the same duration. In a number of embodiments each two second sequence of the source video is encoded as a GOP.

[0024] Retrieval of media using HTTP during adaptive streaming can be improved by adding additional index information to the Matroska container files used to contain each of the encoded streams. In a number of embodiments, the index is a reduced index in that the index only points to the IDRs at the start of each cluster. In many embodiments, the index of the Matroska container file includes additional non-standard attributes (i.e. attributes that do not form part of the Matroska container file format specification) that specify the size of each of the clusters so

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0751
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

that a playback device can retrieve a Cluster element from the Matroska container file via HTTP using a byte range request.

[0025] Adaptive streaming of source media encoded in the manner outlined above can be coordinated by a playback device in accordance with embodiments of the invention. The playback device obtains information concerning each of the available streams from the top level index file and selects one or more streams to utilize in the playback of the media. The playback device can then obtain header information from the Matroska container files containing the one or more bitstreams or streams, and the headers provide information concerning the decoding of the streams. The playback device can also request index information that indexes the encoded media stored within the relevant Matroska container files. The index information can be stored within the Matroska container files or separately from the Matroska container files in the top level index or in separate index files. The index information enables the playback device to request byte ranges corresponding to Cluster elements within the Matroska container file containing specific portions of encoded media via HTTP from the server. As the playback device receives the Cluster elements from the HTTP server, the playback device can evaluate current streaming conditions to determine whether to increase or decrease the bitrate of the streamed media. In the event that the playback device determines that a change in bitrate is necessary, the playback device can obtain header information and index information for the container file(s) containing the desired stream(s) (assuming the playback device has not already obtained this information). The index information can then be used to identify the byte range of the Cluster element containing the next portion of the source media encoded at the desired bit rate and the identified Cluster element can be retrieved from the server via HTTP. The next portion of the source media that is requested is typically identified based upon the Cluster elements already requested by the playback device and the Cluster elements buffered by the playback device. The next portion of source media requested from the alternative stream is requested to minimize the likelihood that the buffer of the playback device will underflow (i.e. run out media to playback) prior to receipt of the Cluster element containing the next portion of source media by the playback device. In this way, the playback device can achieve adaptive bitrate streaming by retrieving sequential Cluster elements from the various streams as appropriate to the streaming conditions using the top level index and index information describing the Cluster elements within each of the Matroska container files.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0752
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0026] In a number of embodiments, variation in the bitrate between different streams can be achieved by modifying the encoding parameters for each stream including but not limited to the bitrate, frame rate, and resolution. When different streams include different resolutions, the display aspect ratio of each stream is the same and the sample aspect ratios are modified to ensure smooth transitions from one resolution to another. The encoding of source video for use in adaptive bitrate streaming and the playback of the encoded source video using HTTP requests to achieve adaptive bitrate streaming in accordance with embodiments of the invention is discussed further below.

ADAPTIVE STREAMING SYSTEM ARCHITECTURE

[0027] An adaptive streaming system in accordance with an embodiment of the invention is illustrated in FIG. 1. The adaptive streaming system 10 includes a source encoder 12 configured to encode source media as a number of alternative streams. In the illustrated embodiment, the source encoder is a server. In other embodiments, the source encoder can be any processing device including a processor and sufficient resources to perform the transcoding of source media (including but not limited to video, audio, and/or subtitles). As is discussed further below, the source encoding server 12 generates a top level index to a plurality of container files containing the streams, at least a plurality of which are alternative streams. Alternative streams are streams that encode the same media content in different ways. In many instances, alternative streams encode media content (such as but not limited to video) at different bitrates. In a number of embodiments, the alternative streams are encoded with different resolutions and/or at different frame rates. The top level index file and the container files are uploaded to an HTTP server 14. A variety of playback devices can then use HTTP or another appropriate stateless protocol to request portions of the top level index file and the container files via a network 16 such as the Internet.

[0028] In many embodiments, the top level index file is a SMIL file and the media is stored in Matroska container files. As is discussed further below, the media can be stored within the Matroska container file in a way that facilitates the adaptive bitrate streaming of the media. In many embodiments, the Matroska container files are specialized Matroska container files that include enhancements (i.e. elements that do not form part of the Matroska file format

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0753
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

specification) that facilitate the retrieval of specific portions of media via HTTP during the adaptive bitrate streaming of the media.

[0029] In the illustrated embodiment, playback devices include personal computers 18 and mobile phones 20. In other embodiments, playback devices can include consumer electronics devices such as DVD players, Blu-ray players, televisions, set top boxes, video game consoles, tablets, and other devices that are capable of connecting to a server via HTTP and playing back encoded media. Although a specific architecture is shown in FIG. 1 any of a variety of architectures can be utilized that enable playback devices to request portions of the top level index file and the container files in accordance with embodiments of the invention.

FILE STRUCTURE

[0030] Files generated by a source encoder and/or stored on an HTTP server for streaming to playback devices in accordance with embodiments of the invention are illustrated in FIG. 2. The files utilized in the adaptive bitrate streaming of the source media include a top level index 30 and a plurality of container files 32 that each contain at least one stream. The top level index file describes the content of each of the container files. As is discussed further below, the top level index file can take a variety of forms including an SMIL file and the container files can take a variety of forms including a specialized Matroska container file.

[0031] In many embodiments, each Matroska container file contains a single stream. For example, the stream could be one of a number of alternate video streams, an audio stream, one of a number of alternate audio streams, a subtitle stream, one of a number of alternate subtitle streams, a trick play stream, or one of a number of alternate trick play streams. In several embodiments, the Matroska container file includes multiple multiplexed streams. For example, the Matroska container could include a video stream, and one or more audio streams, one or more subtitle streams, and/or one or more trick play streams. As is discussed further below, in many embodiments the Matroska container files are specialized files. The encoding of the media and the manner in which the media is stored within Cluster elements within the Matroska container file can be subject to constraints designed to enhance the performance of an adaptive bitrate streaming system. In addition, the Matroska container file can include index elements that facilitate the location and downloading of Cluster elements from the various Matroska container files during the adaptive streaming of the media. Top level index files and Matroska

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0754
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

container files that can be used in adaptive bitrate streaming systems in accordance with embodiments of the invention are discussed below.

TOP LEVEL INDEX FILES

[0032] Playback devices in accordance with many embodiments of the invention utilize a top level index file to identify the container files that contain the streams available to the playback device for use in adaptive bitrate streaming. In many embodiments, the top level index files can include references to container files that each include an alternative stream of encoded media. The playback device can utilize the information in the top level index file to retrieve encoded media from each of the container files according to the streaming conditions experienced by the playback device.

[0033] In several embodiments, the top level index file provides information enabling the playback device to retrieve information concerning the encoding of the media in each of the container files and an index to encoded media within each of the container files. In a number of embodiments, each container file includes information concerning the encoded media contained within the container file and an index to the encoded media within the container file and the top level index file indicates the portions of each container file containing this information. Therefore, a playback device can retrieve the top level index file and use the top level index file to request the portions of one or more of the container files that include information concerning the encoded media contained within the container file and an index to the encoded media within the container file. A variety of top level index files that can be utilized in adaptive bitrate streaming systems in accordance with embodiments of the invention are discussed further below.

TOP LEVEL INDEX SMIL FILES

[0034] In a number of embodiments, the top level index file utilized in the adaptive bitrate streaming of media is a SMIL file, which is an XML file that includes a list of URIs describing each of the streams and the container files that contain the streams. The URI can include information such as the "system-bitrate" of the stream contained within the stream and information concerning the location of specific pieces of data within the container file.

[0035] The basic structure of a SMIL file involves providing an XML declaration and a SMIL element. The SMIL element defines the streams available for use in adaptive bitrate streaming and includes a HEAD element, which is typically left empty and a BODY element that

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0755
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

typically only contains a PAR (parallel) element. The PAR element describes streams that can be played simultaneously (i.e. include media that can be presented at the same time).

[0036] The SMIL specification defines a number of child elements to the PAR element that can be utilized to specify the streams available for use in adaptive bitrate streaming. The VIDEO, AUDIO and TEXTSTREAM elements can be utilized to define a specific video, audio or subtitle stream. The VIDEO, AUDIO and TEXTSTREAM elements can collectively be referred to as media objects. The basic attributes of a media object are the SRC attribute, which specifies the full path or a URI to a container file containing the relevant stream, and the XML:LANG attribute, which includes a 3 letter language code. Additional information concerning a media object can be specified using the PARAM element. The PARAM element is a standard way within the SMIL format for providing a general name value pair. In a number of embodiments of the invention, specific PARAM elements are defined that are utilized during adaptive bitrate streaming.

[0037] In many embodiments, a "header-request" PARAM element is defined that specifies the size of the header section of the container file containing the stream. The value of the "header-request" PARAM element typically specifies the number of bytes between the start of the file and the start of the encoded media within the file. In many embodiments, the header contains information concerning the manner in which the media is encoded and a playback device retrieves the header prior to playback of the encoded media in order to be able to configure the decoder for playback of the encoded media. An example of a "header-request" PARAM element is follows:

```
<param
  name="header-request"
  value="1026"
  valuetype="data" />
```

[0038] In a number of embodiments, a "mime" PARAM element is defined that specifies the MIME type of the stream. A "mime" PARAM element that identifies the stream as being an H.264 stream (i.e. a stream encoded in accordance with the MPEG-4 Advanced Video Codec standard) is as follows:

```
<param
  name="mime"
  value="V_MPEG4/ISO/AVC"
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0756
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

valuetype="data" />

[0039] The MIME type of the stream can be specified using a "mime" PARAM element as appropriate to the encoding of a specific stream (e.g. AAC audio or UTF-8 text stream).

[0040] When the media object is a VIDEO element, additional attributes are defined within the SMIL file format specification including the systemBitrate attribute, which specifies the bitrate of the stream in the container file identified by the VIDEO element, and width and height attributes, which specify the dimensions of the encoded video in pixels. Additional attributes can also be defined using the PARAM element. In several embodiments, a "vbv" PARAM element is defined that specified the VBV buffer size of the video stream in bytes. The video buffering verifier (VBV) is a theoretical MPEG video buffer model used to ensure that an encoded video stream can be correctly buffered and played back at the decoder device. An example of a "vbv" PARAM element that specifies a VBV size of 1000 bytes is as follows:

```
<param
  name="vbv"
  value="1000"
  valuetype="data" />
```

[0041] An example of VIDEO element including the attributes discussed above is as follows:

```
<video
        src="http://cnd.com/video1_620kbps.mkv"
        systemBitrate="620"
        width="480"
        height="270" >
        <param
          name="vbv"
          value="1000"
          valuetype="data" />
</video>
```

[0042] Adaptive bitrate streaming systems in accordance with embodiments of the invention can support trick play streams, which can be used to provide smooth visual search through source content encoded for adaptive bitrate streaming. A trick play stream can be encoded that appears to be an accelerated visual search through the source media when played back, when in reality the trick play stream is simply a separate track encoding the source media at a lower frame rate. In many embodiments of the system a VIDEO element that references a trick play

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0757
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

track is indicated by the systemProfile attribute of the VIDEO element. In other embodiments, any of a variety of techniques can be utilized to signify within the top level index file that a specific stream is a trick play stream. An example of a trick play stream VIDEO element in accordance with an embodiment of the invention is as follows:

```
<video
        src="http://cnd.com/video_test2_600kbps.mkv"
        systemProfile="DivXPlusTrickTrack"
        width="480"
        height="240">
        <param name="vbv" value="1000" valuetype="data" />
        <param name="header-request" value="1000" valuetype="data" />
    </video>
```

[0043] In a number of embodiments of the invention, a "reservedBandwidth" PARAM element can be defined for an AUDIO element. The "reservedBandwidth" PARAM element specifies the bitrate of the audio stream in Kbps. An example of an AUDIO element specified in accordance with an embodiment of the invention is as follows:

```
<audio
   src="http://cnd.com/audio_test1_277kbps.mkv"
   xml:lang="gem"
<param
  name="reservedBandwidth"
  value="128"
  valuetype="data" />
/>
```

[0044] In several embodiments, the "reservedBandwidth" PARAM element is also defined for a TEXTSTREAM element. An example of a TEXTSTREAM element including a "reservedBandwidth" PARAM element in accordance with an embodiment of the invention is as follows:

```
<textstream
   src="http://cnd.com/text_stream_ger.mkv"
   xml:lang="gem"
<param
  name="reservedBandwidth"
  value="32"
  valuetype="data" />
/>
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0758
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

[0045] In other embodiments, any of a variety of mechanisms can be utilized to specify information concerning VIDEO, AUDIO, and SUBTITLE elements as appropriate to specific applications.

[0046] A SWITCH element is a mechanism defined within the SMIL file format specification that can be utilized to define adaptive or alternative streams. An example of the manner in which a SWITCH element can be utilized to specify alternative video streams at different bitrates is as follows:

```
<switch>
   <video src="http://end.com/video_test1_300kbps.mkv"/>
   <video src="http://end.com/video_test2_900kbps.mkv"/>
   <video src="http://end.com/video_test3_1200kbps.mkv"/>
</switch>
```

[0047] The SWTICH element specifies the URLs of three alternative video streams. The file names indicate that the different bitrates of each of the streams. As is discussed further below, the SMIL file format specification provides mechanisms that can be utilized in accordance with embodiments of the invention to specify within the top level index SMIL file additional information concerning a stream and the container file in which it is contained.

[0048] In many embodiments of the invention, the EXCL (exclusive) element is used to define alternative tracks that do not adapt during playback with streaming conditions. For example, the EXCL element can be used to define alternative audio tracks or alternative subtitle tracks. An example of the manner in which an EXCL element can be utilized to specify alternative English and French audio streams is as follows:

```
<excl>
   <audio
      src="http://end.com/english-audio.mkv"
      xml:lang="eng"/>
   <audio
      src="http://end.com/french-audio.mkv"
      xml:lang="fre"/>
</excl>
```

[0049] An example of a top level index SMIL file that defines the attributes and parameters of two alternative video levels, an audio stream and a subtitle stream in accordance with an embodiment of the invention is as follows:

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0759
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
<?xml version="1.0" encoding="utf-8"?>
<smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
  <head>
  </head>
  <body>
    <par>
      <switch>
        <video
          src="http://cnd.com/video_test1_300kbps.mkv"
          systemBitrate="300"
          vbv="600"
          width="320"
          height="240" >
          <param
            name="vbv"
            value="600"
            valuetype="data" />
          <param
            name="header-request"
            value="1000"
            valuetype="data" />
        </video>
        <video
          src="http://cnd.com/video_test2_600kbps.mkv"
          systemBitrate="600"
          vbv ="900"
          width="640"
          height="480">
          <param
            name="vbv"
            value="1000"
            valuetype="data" />
          <param
            name="header-request"
            value="1000"
            valuetype="data" />
        </video>
      </switch>
      <audio
        src="http://cnd.com/audio.mkv"
        xml:lang="eng">
        <param
          name="header-request"
          value="1000"
          valuetype="data" />
        <param name="reservedBandwidth" value="128" valuetype="data" />
```


Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0760
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

```
        </audio>
        <textstream
          src="http://end.com/subtitles.mkv"
          xml:lang="eng">
          <param
            name="header-request"
            value="1000"
            valuetype="data" />
          <param name="reservedBandwidth" value="32" valuetype="data" />
        </textstream>
      </par>
    </body>
</smil>
```

[0050] The top level index SMIL file can be generated when the source media is encoded for playback via adaptive bitrate streaming. Alternatively, the top level index SMIL file can be generated when a playback device requests the commencement of playback of the encoded media. When the playback device receives the top level index SMIL file, the playback device can parse the SMIL file to identify the available streams. The playback device can then select the streams to utilize to playback the content and can use the SMIL file to identify the portions of the container file to download to obtain information concerning the encoding of a specific stream and/or to obtain an index to the encoded media within the container file.

[0051] Although top level index SMIL files are described above, any of a variety of top level index file formats can be utilized to create top level index files as appropriate to a specific application in accordance with an embodiment of the invention. The use of top level index files to enable playback of encoded media using adaptive bitrate streaming in accordance with embodiments of the invention is discussed further below.

STORING MEDIA IN MATROSKA FILES FOR ADAPTIVE BITRATE STREAMING

[0052] A Matroska container file used to store encoded video in accordance with an embodiment of the invention is illustrated in FIG. 3. The container file 32 is an Extensible Binary Markup Language (EBML) file that is an extension of the Matroska container file format. The specialized Matroska container file 32 includes a standard EBML element 34, and a standard Segment element 36 that includes a standard Seek Head element 40, a standard Segment Information element 42, and a standard Tracks element 44. These standard elements describe the media contained within the Matroska container file. The Segment element 36 also includes a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0761
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

standard Clusters element 46. As is described below, the manner in which encoded media is inserted within individual Cluster elements 48 within the Clusters element 46 is constrained to improve the playback of the media in an adaptive streaming system. In many embodiments, the constraints imposed upon the encoded video are consistent with the specification of the Matroska container file format and involve encoding the video so that each cluster includes at least one closed GOP commencing with an IDR frame. In addition to the above standard elements, the Segment element 36 also includes a modified version of the standard Cues element 52. As is discussed further below, the Cues element includes specialized CuePoint elements (i.e. non-standard CuePoint elements) that facilitate the retrieval of the media contained within specific Cluster elements via HTTP.

[0053] The constraints imposed upon the encoding of media and the formatting of the encoded media within the Clusters element of a Matroska container file for adaptive bitrate streaming and the additional index information inserted within the container file in accordance with embodiments of the invention is discussed further below.

ENCODING MEDIA FOR INSERTION IN CLUSTER ELEMENTS

[0054] An adaptive bitrate streaming system provides a playback device with the option of selecting between different streams of encoded media during playback according to the streaming conditions experienced by the playback device. In many embodiments, switching between streams is facilitated by separately pre-encoding discrete portions of the source media in accordance with the encoding parameters of each stream and then including each separately encoded portion in its own Cluster element within the stream's container file. Furthermore, the media contained within each cluster is encoded so that the media is capable of playback without reference to media contained in any other cluster within the stream. In this way, each stream includes a Cluster element corresponding to the same discrete portion of the source media and, at any time, the playback device can select the Cluster element from the stream that is most appropriate to the streaming conditions experienced by the playback device and can commence playback of the media contained within the Cluster element. Accordingly, the playback device can select clusters from different streams as the streaming conditions experienced by the playback device change over time. In several embodiments, the Cluster elements are further constrained so that each Cluster element contains a portion of encoded media from the source

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0762
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

media having the same duration. In a number of embodiments, each Cluster element includes two seconds of encoded media. The specific constraints applied to the media encoded within each Cluster element depending upon the type of media (i.e. video, audio, or subtitles) are discussed below.

[0055] A Clusters element of a Matroska container file containing a video stream in accordance with an embodiment of the invention is illustrated in FIG. 4a. The Clusters element 46 includes a plurality of Cluster elements 48 that each contains a discrete portion of encoded video. In the illustrated embodiment, each Cluster element 48 includes two seconds of encoded video. In other embodiments, the Cluster elements include encoded video having a greater or lesser duration than two seconds. The smaller the Cluster elements (i.e. the smaller the duration of the encoded media within each Cluster element), the higher the overhead associated with requesting each Cluster element. Therefore, a tradeoff exists between the responsiveness of the playback device to changes in streaming conditions and the effective data rate of the adaptive streaming system for a given set of streaming conditions (i.e. the portion of the available bandwidth actually utilized to transmit encoded media). In several embodiments, the encoded video sequences in the Cluster elements have different durations. Each Cluster element 48 includes a Timecode element 60 indicating the start time of the encoded video within the Cluster element and a plurality of BlockGroup elements. As noted above, the encoded video stored within the Cluster is constrained so that the encoded video can be played back without reference to the encoded video contained within any of the other Cluster elements in the container file. In many embodiments, encoding the video contained within the Cluster element as a GOP in which the first frame is an IDR frame enforces the constraint. In the illustrated embodiment, the first BlockGroup element 62 contains an IDR frame. Therefore, the first BlockGroup element 62 does not include a ReferenceBlock element. The first BlockGroup element 62 includes a Block element 64, which specifies the Timecode attribute of the frame encoded within the Block element 64 relative to the Timecode of the Cluster element 48. Subsequent BlockGroup elements 66 are not restricted in the types of frames that they can contain (other than that they cannot reference frames that are not contained within the Cluster element). Therefore, subsequent BlockGroup elements 66 can include ReferenceBlock elements 68 referencing other BlockGroup element(s) utilized in the decoding of the frame contained within the BlockGroup or can contain IDR frames and are similar to the first BlockGroup element 62. As noted above, the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0763
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

manner in which encoded video is inserted within the Cluster elements of the Matroska file conforms with the specification of the Matroska file format.

[0056] The insertion of encoded audio and subtitle information within a Clusters element 46 of a Matroska container file in accordance with embodiments of the invention is illustrated in FIGS. 4b and 4c. In the illustrated embodiments, the encoded media is inserted within the Cluster elements 48 subject to the same constraints applied to the encoded video discussed above with respect to FIG. 4a. In addition, the duration of the encoded audio and subtitle information within each Cluster element corresponds to the duration of the encoded video in the corresponding Cluster element of the Matroska container file containing the encoded video. In other embodiments, the Cluster elements within the container files containing the audio and/or subtitle streams need not correspond with the start time and duration of the Cluster elements in the container files containing the alternative video streams.

MULIPLEXING STREAMS IN A SINGLE MKV CONTAINER FILE

[0057] The Clusters elements shown in FIGS. 4a – 4c assume that a single stream is contained within each Matroska container file. In several embodiments, media from multiple streams is multiplexed within a single Matroska container file. In this way, a single container file can contain a video stream multiplexed with one or more corresponding audio streams, and/or one or more corresponding subtitle streams. Storing the streams in this way can result in duplication of the audio and subtitle streams across multiple alternative video streams. However, the seek time to retrieve encoded media from a video stream and an associated audio, and/or subtitle stream can be reduced due to the adjacent storage of the data on the server. The Clusters element 46 of a Matroska container file containing multiplexed video, audio and subtitle data in accordance with an embodiment of the invention is illustrated in FIG. 4d. In the illustrated embodiment, each Cluster element 48 includes additional BlockGroup elements for each of the multiplexed streams. The first Cluster element includes a first BlockGroup element 62v for encoded video that includes a Block element 64v containing an encoded video frame and indicating the Timecode attribute of the frame relative to the start time of the Cluster element (i.e. the Timecode attribute 60). A second BlockGroup element 62a includes a Block element 64a including an encoded audio sequence and indicating the timecode of the encoded audio relative to the start time of the Cluster element, and a third BlockGroup element 62s including a

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0764
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Block element 64s containing an encoded subtitle and indicating the timecode of the encoded subtitle relative to the start time of the Cluster element. Although not shown in the illustrated embodiment, each Cluster element 48 likely would include additional BlockGroup elements containing additional encoded video, audio or subtitles. Despite the multiplexing of the encoded video, audio, and/or subtitle streams, the same constraints concerning the encoded media apply.

INCORPORATING TRICK PLAY TRACKS IN MKV CONTAINER FILES FOR USE IN ADAPTIVE BITRATE STREAMING SYSTEMS

[0058] The incorporation of trick play tracks within Matroska container files is proposed by DivX, LLC in U.S. Patent Application No. 12/260,404 entitled "Application Enhancement Tracks", filed October 29, 2008, the disclosure of which is hereby incorporated by reference in its entirety. Trick play tracks similar to the trick play tracks described in U.S. Patent Application No. 12/260,404 can be used to provide a trick play stream in an adaptive bitrate streaming system in accordance with an embodiment of the invention to provide smooth visual search through source content encoded for adaptive bitrate streaming. A separate trick play track can be encoded that appears to be an accelerated visual search through the source media when played back, when in reality the trick play track is simply a separate track encoding the source media at a lower frame rate. In several embodiments, the tick play stream is created by generating a trick play track in the manner outlined in U.S. Patent Application No. 12/260,404 and inserting the trick play track into a Matroska container file subject to the constraints mentioned above with respect to insertion of a video stream into a Matroksa container file. In many embodiments, the trick play track is also subject to the further constraint that every frame in the GOP of each Cluster element in the trick play track is encoded as an IDR frame. As with the other video streams, each Cluster element contains a GOP corresponding to the same two seconds of source media as the corresponding Cluster elements in the other streams. There are simply fewer frames in the GOPs of the trick play track and each frame has a longer duration. In this way, transitions to and from a trick play stream can be treated in the same way as transitions between any of the other encoded streams are treated within an adaptive bitrate streaming system in accordance with embodiments of the invention. Playback of the frames contained within the trick play track to achieve accelerated visual search typically involves the playback device manipulating the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0765
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

timecodes assigned to the frames of encoded video prior to providing the frames to the playback device's decoder to achieve a desired increase in rate of accelerated search (e.g. x2, x4, x6, etc.).

[0059] A Clusters element containing encoded media from a trick play track is shown in FIG. 4c. In the illustrated embodiment, the encoded trick play track is inserted within the Cluster elements 48 subject to the same constraints applied to the encoded video discussed above with respect to FIG. 4a. However, each Block element contains an IDR. In other embodiments, the Cluster elements within the container files containing the trick play tracks need not correspond with the start time and duration of the Cluster elements in the container files containing the alternative video streams.

[0060] In many embodiments, source content can be encoded to provide a single trick play track or multiple trick play tracks for use by the adaptive bit rate streaming system. When a single trick play track is provided, the trick play track is typically encoded at a low bitrate. When multiple alternative trick play tracks are provided, adaptive rate streaming can also be performed with respect to the trick play tracks. In several embodiments, multiple trick play tracks are provided to support different rates of accelerated visual search through the encoded media.

INCORPORATING INDEXING INFORMATION WITHIN MKV CONTAINER FILES

[0061] The specification for the Matroska container file format provides for an optional Cues element that is used to index Block elements within the container file. A modified Cues element 52 that can be incorporated into a Matroska container file in accordance with an embodiment of the invention to facilitate the requesting of clusters by a playback device using HTTP is illustrated in FIG. 5. The modified Cues element 52 includes a plurality of CuePoint elements 70 that each include a CueTime attribute 72. Each CuePoint element includes a CueTrackPositions element 74 containing the CueTrack 76 and CueClusterPosition 78 attributes. In many embodiments, the CuePoint element is mainly configured to identify a specific Cluster element as opposed to a specific Block element within a Cluster element. Although, in several applications the ability to seek to specific BlockGroup elements within a Cluster element is required and additional index information is included in the Cues element.

[0062] The use of a modified Cues element to index encoded media within a Clusters element of a Matroska file in accordance with an embodiment of the invention is illustrated in

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0766
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

FIG. 6. A CuePoint element is generated to correspond to each Cluster element within the Matroska container file. The CueTime attribute 72 of the CuePoint element 70 corresponds to the Timecode attribute 60 of the corresponding Cluster element 48. In addition, the CuePoint element contains a CueTrackPositions element 74 having a CueClusterPosition attribute 78 that points to the start of the corresponding Cluster element 48. The CueTrackPositions element 74 can also include a CueBlockNumber attribute, which is typically used to indicate the Block element containing the first IDR frame within the Cluster element 48.

[0063] As can readily be appreciated the modified Cues element 52 forms an index to each of the Cluster elements 48 within the Matroska container file. Furthermore, the CueTrackPosition elements provide information that can be used by a playback device to request the byte range of a specific Cluster element 48 via HTTP or another suitable protocol from a remote server. The Cues element of a conventional Matroska file does not directly provide a playback device with information concerning the number of bytes to request from the start of the Cluster element in order to obtain all of the encoded video contained within the Cluster element. The size of a Cluster element can be inferred in a modified Cues element by using the CueClusterPosition attribute of the CueTrackPositions element that indexes the first byte of the next Cluster element. Alternatively, additional CueTrackPosition elements could be added to modified Cues elements in accordance with embodiments of the invention that index the last byte of the Cluster element (in addition to the CueTrackPositions elements that index the first byte of the Cluster element), and/or a non-standard CueClusterSize attribute that specifies the size of the Cluster element pointed to by the CueClusterPosition attribute is included in each CueTrackPosition element to assist with the retrieval of specific Cluster elements within a Matroska container file via HTTP byte range requests or a similar protocol.

[0064] The modification of the Cues element in the manner outlined above significantly simplifies the retrieval of Cluster elements from a Matroska container file via HTTP or a similar protocol during adaptive bitrate streaming. In addition, by only indexing the first frame in each Cluster the size of the index is significantly reduced. Given that the index is typically downloaded prior to playback, the reduced size of the Cues element (i.e. index) means that playback can commence more rapidly. Using the CueClusterPosition elements, a playback device can request a specific Cluster element from the stream most suited to the streaming

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0767
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

conditions experienced by the playback device by simply referencing the index of the relevant Matroska container file using the Timecode attribute for the desired Cluster element.

[0065] In some embodiments, a number of the attributes within the Cues element are not utilized during adaptive bitrate streaming. Therefore, the Cues element can be further modified by removing the unutilized attributes to reduce the overall size of the index for each Matroska container file. A modified Cues element that can be utilized in a Matroska container file that includes a single encoded stream in accordance with an embodiment of the invention is illustrated in FIG. 5a. The Cues element 52' shown in FIG. 5a is similar to the Cues element 52 shown in FIG. 5 with the exception that the CuePoint elements 70' do not include a CueTime attribute (see 72 in FIG. 5) and/or the CueTrackPositions elements 74' do not include a CueTrack attribute (76 in FIG. 5). When the portions of encoded media in each Cluster element in the Motroska container file have the same duration, the CueTime attribute is not necessary. When the Matroska contain file includes a single encoded stream, the CueTrack attribute is not necessary. In other embodiments, the Cues element and/or other elements of the Matroska container file can be modified to remove elements and/or attributes that are not necessary for the adaptive bitrate streaming of the encoded stream contained within the Matroska container file, given the manner in which the stream is encoded and inserted in the Matroska container file.

[0066] Although various modifications to the Cues element to include information concerning the size of each of the Cluster elements within a Matroska container file and to eliminate unnecessary attributes are described above, many embodiments of the invention utilize a conventional Matroska container. In several embodiments, the playback device simply determines the size of Cluster elements on the fly using information obtained from a conventional Cues element, and/or relies upon a separate index file containing information concerning the size and/or location of the Cluster elements within the MKV container file. In several embodiments, the additional index information is stored in the top level index file. In a number of embodiments, the additional index information is stored in separate files that are identified in the top level index file. When index information utilized to retrieve Cluster elements from a Matroska container file is stored separately from the container file, the Matroska container file is still typically constrained to encode media for inclusion in the Cluster elements in the manner outlined above. In addition, wherever the index information is located, the index information will typically index each Cluster element and include (but not be limited to)

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0768
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

information concerning at least the starting location and, in many instances, the size of each Cluster element.

ENCODING SOURCE MEDIA FOR ADAPTIVE BITRATE STREAMING

[0067] A process for encoding source media as a top level index file and a plurality of Matroska container files for use in an adaptive bitrate streaming system in accordance with an embodiment of the invention is illustrated in FIG. 7. The encoding process 100 commences by selecting (102) a first portion of the source media and encoding (104) the source media using the encoding parameters for each stream. When the portion of media is video, then the portion of source video is encoded as a single GOP commencing with an IDR frame. In many embodiments, encoding parameters used to create the alternative GOPs vary based upon bitrate, frame rate, encoding parameters and resolution. In this way, the portion of media is encoded as a set of interchangeable alternatives and a playback device can select the alternative most appropriate to the streaming conditions experienced by the playback device. When different resolutions are supported, the encoding of the streams is constrained so that each stream has the same display aspect ratio. A constant display aspect ratio can be achieved across different resolution streams by varying the sample aspect ratio with the resolution of the stream. In many instances, reducing resolution can result in higher quality video compared with higher resolution video encoded at the same bit rate. In many embodiments, the source media is itself encoded and the encoding process (104) involves transcoding or transrating of the encoded source media according to the encoding parameters of each of the alternative streams supported by the adaptive bitrate streaming system.

[0068] Once the source media has been encoded as a set of alternative portions of encoded media, each of the alternative portions of encoded media is inserted (106) into a Cluster element within the Matroska container file corresponding to the stream to which the portion of encoded media belongs. In many embodiments, the encoding process also constructs indexes for each Matroska container file as media is inserted into Cluster elements within the container. Therefore, the process 100 can also include creating a CuePoint element that points to the Cluster element inserted within the Matroska container file. The CuePoint element can be held in a buffer until the source media is completely encoded. Although the above process describes encoding each of the alternative portions of encoded media sequentially in a single pass through

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0769
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the source media, many embodiments of the invention involve performing a separate pass through the source media to encode each of the alternative streams.

[0069] Referring back to FIG. 7, the process continues to select (102) and encode (104) portions of the source media and then insert (106) the encoded portions of media into the Matroska container file corresponding to the appropriate stream until the entire source media is encoded for adaptive bitrate streaming (108). At which point, the process can insert an index (110) into the Matroska container for each stream and create (112) a top level index file that indexes each of the encoded streams contained within the Matroska container files. As noted above, the indexes can be created as encoded media and inserted into the Matroska container files so that a CuePoint element indexes each Cluster element within the Mastroska container file. Upon completion of the encoding, each of the CuePoint elements can be included in a Cues element and the Cues element can be inserted into the Matroska container file following the Clusters element.

[0070] Following the encoding of the source media to create Matroska container files containing each of the streams generated during the encoding process, which can include the generation of trick play streams, and a top level index file that indexes each of the streams within the Matroska container files, the top level index file and the Matroska container files can be uploaded to an HTTP server for adaptive bitrate streaming to playback devices. The adaptive bitrate streaming of media encoded in accordance with embodiments of the invention using HTTP requests is discussed further below.

ADAPTIVE BITRATE STREAMING FROM MKV CONTAINER FILES USING HTTP

[0071] When source media is encoded so that there are alternative streams contained in separate Matroska container files for at least one of video, audio, and subtitle content, adaptive streaming of the media contained within the Matroska container files can be achieved using HTTP requests or a similar stateless data transfer protocol. In many embodiments, a playback device requests the top level index file resident on the server and uses the index information to identify the streams that are available to the playback device. The playback device can then retrieve the indexes for one or more of the Matroska files and can use the indexes to request media from one or more of the streams contained within the Matroska container files using HTTP requests or using a similar stateless protocol. As noted above, many embodiments of the

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0770
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

invention implement the indexes for each of the Matroska container files using a modified Cues element. In a number of embodiments, however, the encoded media for each stream is contained within a standard Matroska container file and separate index file(s) can also be provided for each of the container files. Based upon the streaming conditions experienced by the playback device, the playback device can select media from alternative streams encoded at different bitrates. When the media from each of the streams is inserted into the Matroska container file in the manner outlined above, transitions between streams can occur upon the completion of playback of media within a Cluster element. Therefore, the size of the Cluster elements (i.e the duration of the encoded media within the Cluster elements) is typically chosen so that the playback device is able to respond quickly enough to changing streaming conditions and to instructions from the user that involve utilization of a trick play track. The smaller the Cluster elements (i.e. the smaller the duration of the encoded media within each Cluster element), the higher the overhead associated with requesting each Cluster element. Therefore, a tradeoff exists between the responsiveness of the playback device to changes in streaming conditions and the effective data rate of the adaptive streaming system for a given set of streaming conditions (i.e. the portion of the available bandwidth actually utilized to transmit encoded media). In many embodiments, the size of the Cluster elements is chosen so that each Cluster element contains two seconds of encoded media. In other embodiments, the duration of the encoded media can be greater or less than two seconds and/or the duration of the encoded media can vary from Cluster element to Cluster element.

[0072] Communication between a playback device or client and an HTTP server during the playback of media encoded in separate streams contained within Matroska container files indexed by a top level index file in accordance with an embodiment of the invention is illustrated in FIG. 8. In the illustrated embodiment, the playback device 200 commences playback by requesting the top level index file from the server 202 using an HTTP request or a similar protocol for retrieving data. The server 202 provides the bytes corresponding to the request. The playback device 200 then parses the top level index file to identify the URIs of each of the Matroska container files containing the streams of encoded media derived from a specific piece of source media. The playback device can then request the byte ranges corresponding to headers of one or more of the Matroska container files via HTTP or a similar protocol, where the byte ranges are determined using the information contained in the URI for the relevant Matroska

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0771
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

container files (see discussion above). The server returns the following information in response to a request for the byte range containing the headers of a Matroska container file:

ELEM("EBML")
ELEM("SEEKHEAD")
ELEM("SEGMENTINFO")
ELEM("TRACKS")

[0073] The EBML element is typically processed by the playback device to ensure that the file version is supported. The SeekHead element is parsed to find the location of the Matroska index elements and the SegmentInfo element contains two key elements utilized in playback: TimecodeScale and Duration. The TimecodeScale specifies the timecode scale for all timecodes within the Segment of the Matroska container file and the Duration specifies the duration of the Segment based upon the TimecodeScale. The Tracks element contains the information used by the playback device to decode the encoded media contained within the Clusters element of the Matroska file. As noted above, adaptive bitrate streaming systems in accordance with embodiments of the invention can support different streams encoded using different encoding parameters including but not limited to frame rate, and resolution. Therefore, the playback device can use the information contained within the Matroska container file's headers to configure the decoder every time a transition is made between encoded streams.

[0074] In many embodiments, the playback device does not retrieve the headers for all of the Matroska container files indexed in the top level index file. Instead, the playback device determines the stream(s) that will be utilized to initially commence playback and requests the headers from the corresponding Matroska container files. Depending upon the structure of the URIs contained within the top level index file, the playback device can either use information from the URIs or information from the headers of the Matroska container files to request byte ranges from the server that contain at least a portion of the index from relevant Matroska container files. The byte ranges can correspond to the entire index. The server provides the relevant byte ranges containing the index information to the playback device, and the playback device can use the index information to request the byte ranges of Cluster elements containing encoded media using this information. When the Cluster elements are received, the playback device can extract encoded media from the Block elements within the Cluster element, and can

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0772
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

decode and playback the media within the Block elements in accordance with their associated Timecode attributes.

[0075] In the illustrated embodiment, the playback device 200 requests sufficient index information from the HTTP server prior to the commencement of playback that the playback device can stream the entirety of each of the selected streams using the index information. In other embodiments, the playback device continuously retrieves index information as media is played back. In several embodiments, all of the index information for the lowest bitrate steam is requested prior to playback so that the index information for the lowest bitrate stream is available to the playback device in the event that streaming conditions deteriorate rapidly during playback.

SWITCHING BETWEEN STREAMS

[0076] The communications illustrated in FIG. 8 assume that the playback device continues to request media from the same streams (i.e. Matroska container files) throughout playback of the media. In reality, the streaming conditions experienced by the playback device are likely to change during the playback of the streaming media and the playback device can request media from alternative streams (i.e. different Matroska container files) to provide the best picture quality for the streaming conditions experienced by the playback device. In addition, the playback device may switch streams in order to perform a trick play function that utilizes a trick play track stream.

[0077] Communication between a playback device and a server when a playback device switches to a new stream in accordance with embodiments of the invention are illustrated in FIG. 9a. The communications illustrated in FIG. 9a assume that the index information for the new stream has not been previously requested by the playback device and that downloading of Cluster elements from the old stream proceeds while information is obtained concerning the Matroska container file containing the new stream. When the playback device 200 detects a change in streaming conditions, determines that a higher bitrate stream can be utilized at the present streaming conditions, or receives a trick play instruction from a user, the playback device can use the top level index file to identify the URI for a more appropriate alternative stream to at least one of the video, audio, or subtitle streams from which the playback device is currently requesting encoded media. The playback device can save the information concerning the current stream(s) and can request the byte ranges of the headers for the Matroska container file(s)

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0773
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

containing the new stream(s) using the parameters of the corresponding URIs. Caching the information in this way can be beneficial when the playback device attempts to adapt the bitrate of the stream downward. When the playback device experiences a reduction in available bandwidth, the playback device ideally will quickly switch to a lower bitrate stream. Due to the reduced bandwidth experienced by the playback device, the playback device is unlikely to have additional bandwidth to request header and index information. Ideally, the playback device utilizes all available bandwidth to download already requested higher rate Cluster elements and uses locally cached index information to start requesting Cluster elements from Matroska container file(s) containing lower bitrate stream(s).

[0078] Byte ranges for index information for the Matroska container file(s) containing the new stream(s) can be requested from the HTTP server 202 in a manner similar to that outlined above with respect to FIG. 8. At which point, the playback device can stop downloading of cluster elements from the previous streams and can commence requesting the byte ranges of the appropriate Cluster elements from the Matroska container file(s) containing the new stream(s) from the HTTP server, using the index information from the Matroska container file(s) to identify the Cluster element(s) containing the encoded media following the encoded media in the last Cluster element retrieved by the playback device. As noted above, the smooth transition from one stream to another is facilitated by encoding each of the alternative streams so that corresponding Cluster elements start with the same Timecode element and an IDR frame.

[0079] When the playback device caches the header and the entire index for each stream that has be utilized in the playback of the media, the process of switching back to a previously used stream can be simplified. The playback device already has the header and index information for the Matroska file containing the previously utilized stream and the playback device can simply use this information to start requesting Cluster elements from the Matroska container file of the previously utilized stream via HTTP. Communication between a playback device and an HTTP server when switching back to a stream(s) for which the playback device has cached header and index information in accordance with an embodiment of the invention is illustrated in FIG. 9b. The process illustrated in FIG. 9b is ideally performed when adapting bitrate downwards, because a reduction in available resources can be exacerbated by a need to download index information in addition to media. The likelihood of interruption to playback is reduced by

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0774
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

increasing the speed with which the playback device can switch between streams and reducing the amount of overhead data downloaded to achieve the switch.

[0080] Although the present invention has been described in certain specific aspects, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that the present invention may be practiced otherwise than specifically described, including various changes in the implementation such as utilizing encoders and decoders that support features beyond those specified within a particular standard with which they comply, without departing from the scope and spirit of the present invention. Thus, embodiments of the present invention should be considered in all respects as illustrative and not restrictive.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0775
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

WHAT IS CLAIMED:

1. A playback device configured to perform adaptive bitrate streaming using a plurality of alternative streams of encoded video packaged within extensible binary markup language (EBML) container files, where each of the plurality of alternative streams is the same source video encoded using different encoding parameters in such a way that each portion of encoded video packed into an element of an EBML container file commences with an intra frame starting a closed Group of Pictures (GOP) and each of the EBML container files includes at least one element that specifies the encoding parameters of the stream contained within the EBML container file, the playback device comprising:

a processor configured, via a client application, to request portions of files from a remote server;

wherein the client application further configures the processor to:

retrieve top level index data that identifies a plurality of EBML container files and describes at least a maximum bitrate of the alternative streams contained within the EBML container files;

parse the top level index data to obtain information identifying the plurality of EBML container files;

request a portion of at least one of the EBML container files that contains the at least one element that specifies the encoding parameters of the stream contained within the EBML container file;

retrieve an index that references each element containing portions of encoded video within at least one of the EBML container files;

utilize the index to request portions of a first EBML container file that includes elements that contain portions of encoded video;

receive and buffer the requested elements;

decode the encoded video contained within the buffered elements utilizing the encoding parameters;

measure current streaming conditions; and

select another of the EBML container files from which to retrieve elements containing portions of encoded video for decoding, where the selection is based upon

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0776
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the measured streaming conditions and the description of the bitrate of the alternative stream contained within the top level data.

2. The playback device of claim 1, wherein:

each of the EBML container files comprises a plurality of Cluster elements, where each Cluster element contains a portion of encoded video; and

the portion of encoded video in each of the Cluster elements commences with an intra frame and includes at least one closed group of pictures.

3. The playback device of claim 2, wherein the portions of encoded video in each of the Cluster elements have the same duration.

4. The playback device of claim 2, wherein the portions of encoded video in each of the Cluster elements have a 2 second duration.

5. The playback device of claim 2, wherein each Cluster element contains a time code and each encoded frame of the portion of encoded video contained within the Cluster element is contained within a separate BlockGroup element.

6. The playback device of claim 5, wherein the first BlockGroup element in the Cluster element contains the intra frame.

7. The playback device of claim 6, wherein the first BlockGroup element contains a Block element, which specifies the time code attribute of the intra frame relative to the time code of the Cluster element.

8. The playback device of claim 1, wherein each of the alternative streams of encoded video is encoded at a different maximum bitrate.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0777
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

9. The playback device of claim 8, wherein at least two of the alternative streams of encoded video are encoded at the same display aspect ratio, but with different resolutions using different sample aspect ratios.

10. The playback device of claim 9, wherein the at least one element in each of the EBML container files that specifies the encoding parameters of the stream contained within the EBML container file specifies the frame height, frame width and sample aspect ratio of the video stream contained within the EBML container file.

11. The playback device of claim 10, wherein the client application configures the processor to set the frame height, frame width, and sample aspect ratio prior to commencing the decoding of encoded video contained within buffered elements from one of the EBML container files.

12. The playback device of claim 8, wherein at least two of the alternative streams of encoded video are encoded at different frame rates.

13. The playback device of claim 12, wherein the at least one element in each of the EBML container files that specifies the encoding parameters of the stream contained within the EBML container file specifies the frame rate of the video stream contained within the EBML container file.

14. The playback device of claim 13, wherein the client application configures the processor to set the frame rate prior to commencing the decoding of encoded video contained within buffered elements from one of the EBML container files.

15. The playback device of claim 11, wherein the client application configures the playback device to request portions of files from remote servers via Hypertext Transfer Protocol (HTTP) byte range requests.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0778
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

16. The playback device of claim 1, wherein the top level index data is SMIL data and the information that identifies the plurality of EBML container files is a plurality of Universal Resource Indicators that identify the location of each of the EBML container files.

17. The playback device of claim 1, wherein the SMIL data specifies the maximum bitrate of the encoded stream contained within the EBML container file identified by each of the URIs.

18. The playback device of claim 17, wherein:

the index that references each of the elements containing portions of the encoded stream within the EBML container file is located within at least one element of the EBML container file;

the SMIL file specifies the portions of the EBML container file that contain the elements that contain the encoding parameters;

the client application further configures the processor to:

parse the SMIL file to identify the portions of the EBML container file that contain the elements that contain the encoding parameters; and

request the portions of the EBML container file that contain the elements that contain the encoding parameters.

19. The playback device of claim 1, wherein:

each EBML container file includes at least one element containing an index that references each element containing portions of encoded video within the EBML container file; and

the client application configures the processor to retrieve the index that references each element containing portions of encoded video within an EBML container file by requesting the portion of the EBML container file that includes the at least one element containing the index.

20. The playback device of claim 19, wherein the index is located within a Cues element within the EBML container file.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0779
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

21. The playback device of claim 20, wherein the Cues element includes a plurality of CuePoint elements that include a time attribute and point to the location of each element containing a portion of encoded video within the EBML file.

22. The playback device of claim 21, wherein the client application configures the processor to:

search for the location of an element containing a specific portion of encoded video within the EBML file using the time attributes of the CuePoint elements within the index; and

request a portion of the EBML file including at least the element identified using the CuePoint elements.

23. The playback device of claim 1, wherein the encoding parameters utilized to decode the encoded video include at least one encoding parameter selected from the group consisting of frame rate, frame height, frame width, sample aspect ratio, maximum bitrate, and minimum buffer size.

24. The playback device of claim 1, wherein the client application configures the processor to measure the current streaming conditions by measuring the time taken to receive requested elements from the time at which the elements were requested.

25. The playback device of claim 1, wherein:

the client application configures the processor to initially request a portion of a first EBML container file containing at least one element that specifies the encoding parameters of the stream contained within the first EBML container file; and

the client application further configures the processor to request a portion of a second EBML container file containing at least one element that specifies the encoding parameters of the stream contained within the second EBML container file, when the client application selects to retrieve elements containing portions of encoded video for decoding from

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0780
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

the second EBML container file based upon the measured streaming conditions and the description of the bitrate of the alternative stream contained within the top level index.

26. The playback device of claim 1, wherein the playback device is further configured to perform adaptive bitrate streaming using an EBML container file that contains a trick play track.

27. The playback device of claim 26, wherein the client application configures the processor to select the EBML container file containing the trick play track as the EBML container file from which to retrieve elements containing portions of encoded video in response to a user instruction.

28. The playback device of claim 1, wherein the playback device is further configured to perform adaptive streaming using at least one EBML container file containing a track of encoded audio.

29. The playback device of claim 28, wherein the client application configures the processor to retrieve and decode encoded audio from one of the at least one EBML container files containing an audio track and to synchronize the decoded audio with the decoded video.

30. The playback device of claim 1, wherein each of the EBML container files includes elements containing at least one track of encoded audio multiplexed with the encoded video.

31. The playback device of claim 28, wherein the client application is configured to:

retrieve portions of the EBML file containing elements that include portions of one of the tracks of encoded audio;

decode the retrieved encoded audio; and

synchronize the decoded audio and the decoded video.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0781
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

32. The playback device of claim 1, wherein the playback device is further configured to perform adaptive streaming using at least one EBML container file containing a subtitle track.

33. The playback device of claim 32, wherein the client application configures the processor to:

retrieve subtitles from one of the at least one EBML container files containing a subtitle track;

synchronize the subtitle with the decoded video; and

overlay the subtitle over the decoded video.

34. The playback device of claim 1, wherein each of the EBML container files includes elements containing at least one subtitle track multiplexed with the encoded video.

35. The playback device of claim 34, wherein the client application is configured to:

retrieve portions of the EBML file containing elements that include portions of one of the subtitle tracks;

synchronize the subtitles with the decoded video; and

overlay the subtitles on the decoded video.

36. A method of performing adaptive bitrate streaming of media on a playback device using a plurality of alternative streams of encoded video packaged within extensible binary markup language (EBML) container files, where each of the alternative streams of video is the same source video encoded using different encoding parameters in such a way that each portion of encoded video packed into an element of an EBML container file commences with an intra frame starting a closed Group of Pictures (GOP) and each of the EBML container files includes at least one element that specifies the encoding parameters of the stream contained within the EBML container file, the method comprising:

retrieving top level index data that identifies each of the plurality of EBML container files and describes at least the maximum bitrate of each of the alternative streams contained within the EBML container files;

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0782
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

parsing the top level index data to obtain information identifying the plurality of EBML container files;

requesting a portion of at least one of the EBML container files that contains the at least one element that specifies the encoding parameters of the stream contained within the EBML container file;

retrieving an index that references each element containing portions of encoded video within at least one of the EBML container files;

utilizing the index to request portions of a first EBML container file that includes elements that contain portions of encoded video;

receiving and buffering the requested elements;

decoding the encoded video contained within the buffered elements utilizing the encoding parameters;

measuring current streaming conditions; and

selecting another of the EBML container files from which to retrieve elements containing portions of encoded video for decoding, where the selection is based upon the measured streaming conditions and the description of the bitrate of the alternative stream contained within the top level index data.

37. A source encoder configured to encode source video as a plurality of alternative video streams packed in container files, where the container files are extensible binary markup language (EBML) files, the source encoder comprising:

a processor configured via a source encoding application to ingest at least one multimedia file containing a source video;

wherein the source encoding application further configures the processor to:

select a portion of the source video;

transcode the selected portion of the source video into a plurality of alternative portions of encoded video, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed Group of Pictures (GOP);

write each of the alternative portions of encoded video to an element of a different EBML container file, where each element is located within an EBML container file that

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0783
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

also includes another element that indicates the encoding parameters used to encode the alternative portion of encoded video; and

add an entry to at least one index that identifies the location of the element containing one of the alternative portions of encoded video within each of the EBML container files.

38. The source encoder of claim 37, wherein transcoding a selected portion of the source video further comprises transcoding the selected portion into at least one closed group of pictures.

39. The source encoder of claim 37, wherein the portion of source video is selected based upon the duration of the selected portion of source video.

40. The source encoder of claim 39, wherein the source encoding application configures the processor to select a portion of the source video having a duration of two seconds.

41. The source encoder of claim 37, wherein each of the alternative portions of encoded video is encoded with a different maximum bitrate.

42. The source encoder of claim 41, wherein at least two of the alternative portions of encoded video are encoded with different resolutions.

43. The source encoder of claim 41, wherein at least two of the alternative portions of encoded video are encoded with different frame rates.

44. The source encoder of claim 37, wherein the element of the EBML container file to which each alternative portion of encoded video is written is a Cluster element containing a time code and the portion of encoded video is contained within BlockGroup elements within the Cluster element.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0784
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

45. The source encoder of claim 44, wherein each encoded frame of the alternative portion of encoded video contained within the Cluster element is contained within a separate BlockGroup element.

46. The source encoder of claim 45, wherein the first BlockGroup element in the Cluster element contains the IDR frame.

47. The source encoder of claim 46, wherein the first BlockGroup element contains a Block element, which specifies the time code attribute of the IDR frame relative to the time code of the Cluster element.

48. The source encoder of claim 37, wherein each element to which each of the alternative portions of encoded video is written is assigned the same time code.

49. The source encoder of claim 37, wherein the source encoding application further configures the processor to create an index for each of the EBML container files.

50. The source encoder of claim 49, wherein the source encoding application further configures the processor to add the location of the element containing one of the alternative portions of encoded video within each of the EBML container files to the index for the EBML container file.

51. The source encoder of claim 49, wherein the source encoding application further configures the processor to pack the index for each EBML container file into the EBML container file.

52. The source encoder of claim 51, wherein each index comprises a Cues element.

53. The source encoder of claim 52, wherein each Cues element includes a CuePoint element that points to the location of the element containing one of the alternative portions of encoded video within the EBML file.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0785
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

54. The source encoder of claim 37, wherein the source encoding application further configures the processor to create a top level index file that identifies each of the EBML container files.

55. The source encoder of claim 37, wherein the ingested multimedia file also includes source audio.

56. The source encoder of claim 55, wherein the source encoding application configures the processor to multiplex the audio into each of the EBML container files.

57. The source encoder of claim 55, wherein the source encoding application configures the processor to write the audio to a separate EBML container file.

58. The source encoder of claim 55, wherein the source encoding application further configures the processor to transcode at least one of the at least one audio tracks.

59. The source encoder of claim 55, wherein the ingested multimedia file further comprises subtitles.

60. The source encoder of claim 59, wherein the source encoding application configures the processor to multiplex the subtitles into each of the EBML container files.

61. The source encoder of claim 59, wherein the source encoding application configures the processor to write the subtitles to a separate EBML container file.

62. The source encoder of claim 37, wherein the source encoding application further configures the processor to transcode the source video to create a lower frame rate trick play track and to write the trick play track to a separate EBML container file.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0786
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

63. The source encoder of claim 62, wherein the trick play track is also lower resolution than the source video.

64. The source encoder of claim 37, wherein the source encoding application further configures the processor to write the element containing a set of encoding parameters in each of the EBML container files.

65. The source encoder of claim 64, wherein the set of encoding parameters includes at least one parameter selected from the group consisting of frame rate, frame height, frame width, sample aspect ratio, maximum bitrate, and minimum buffer size.

66. A method of encoding source video as a plurality of alternative streams packed in extensible binary markup language (EBML) files using a source encoder, the method comprising:

repeatedly selecting a portion of the source video using the source encoder;

transcoding the selected portion of the source video into a plurality of alternative portions of encoded video using the source encoder, where each alternative portion is encoded using a different set of encoding parameters and commences with an intra frame starting a closed Group of Pictures (GOP);

writing each of the alternative portions of encoded video to an element of a different EBML container file using the source encoder, where each element is located within an EBML container file that also includes another element containing a set of encoding parameters corresponding to the encoding parameters used to encode the portion of video; and

adding an entry to at least one index that identifies the location of the element containing one of the alternative portions of encoded video within each of the EBML container files.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0787
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

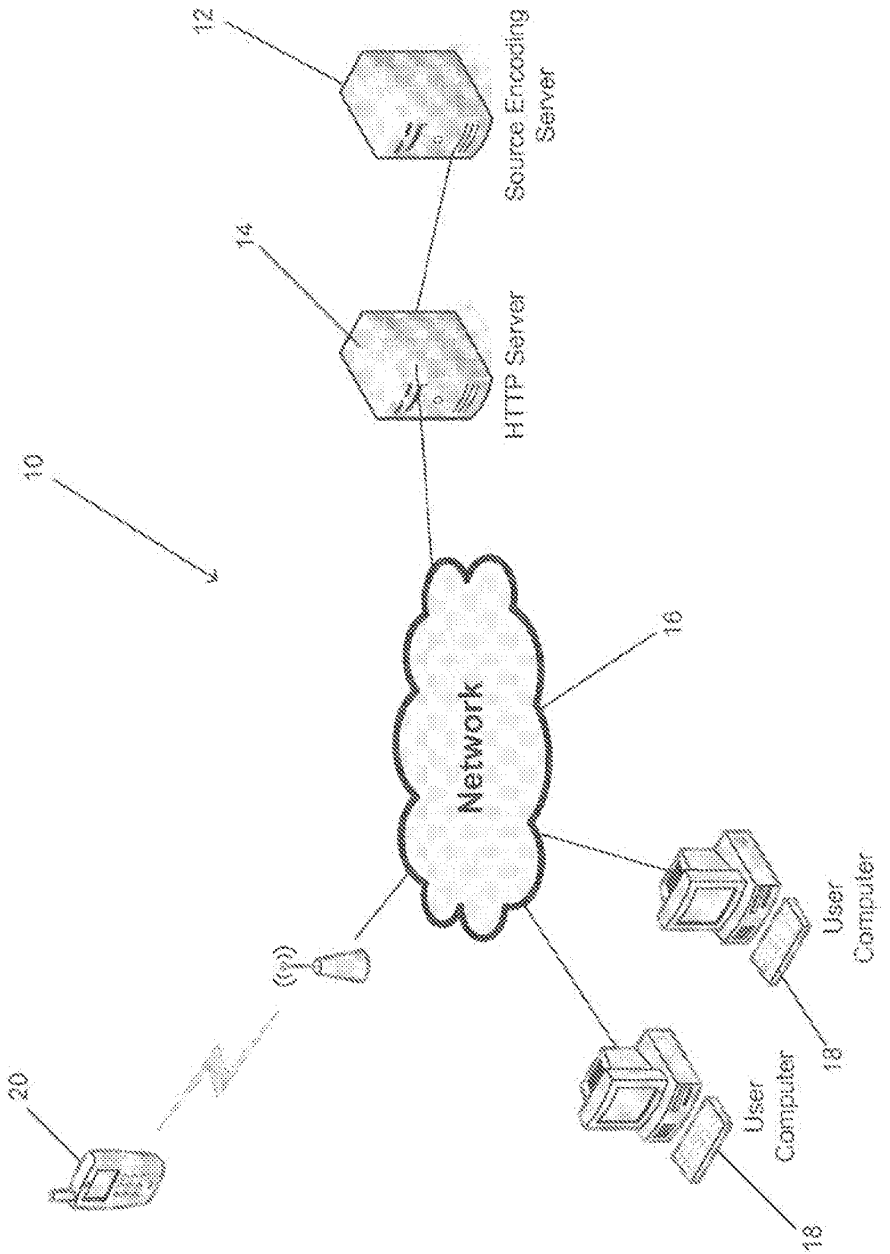

12
Source Encoding Server
14
HTTP Server
10
Network
16
User Computer
18
User Computer
18
20

FIG. 1

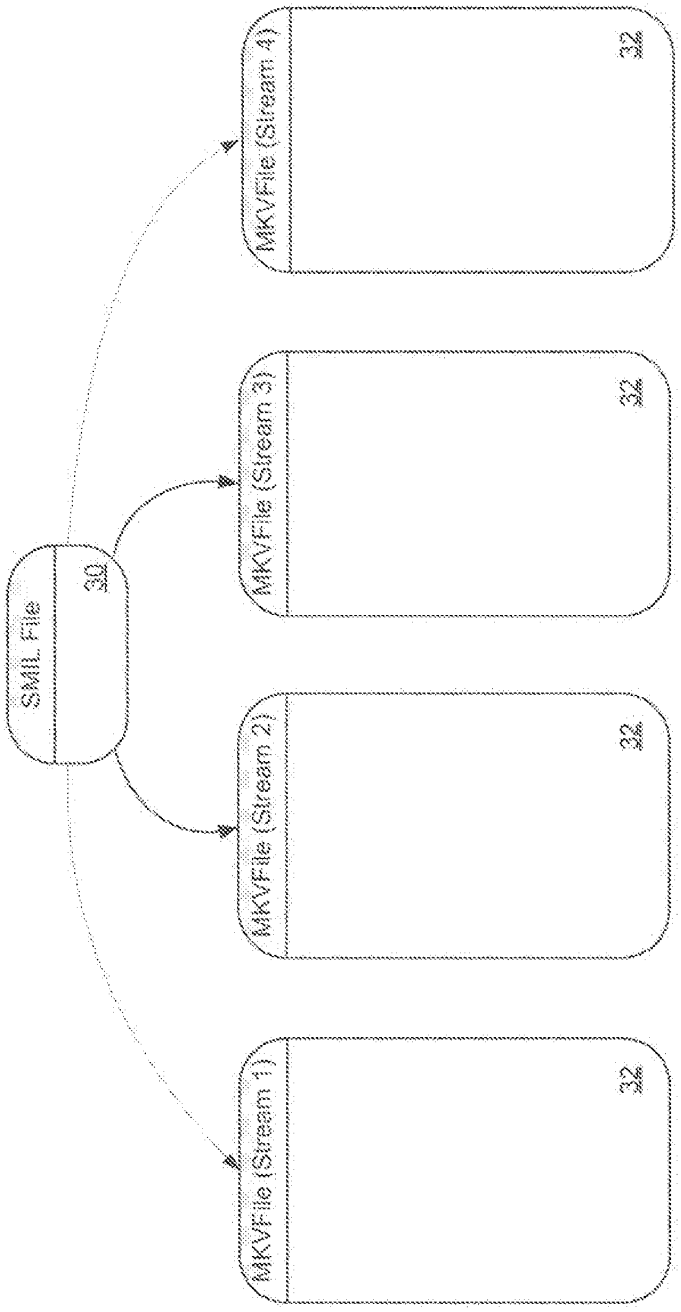

SMIL File
30
MKVFile (Stream 1)
32
MKVFile (Stream 2)
32
MKVFile (Stream 3)
32
MKVFile (Stream 4)
32

FIG. 2

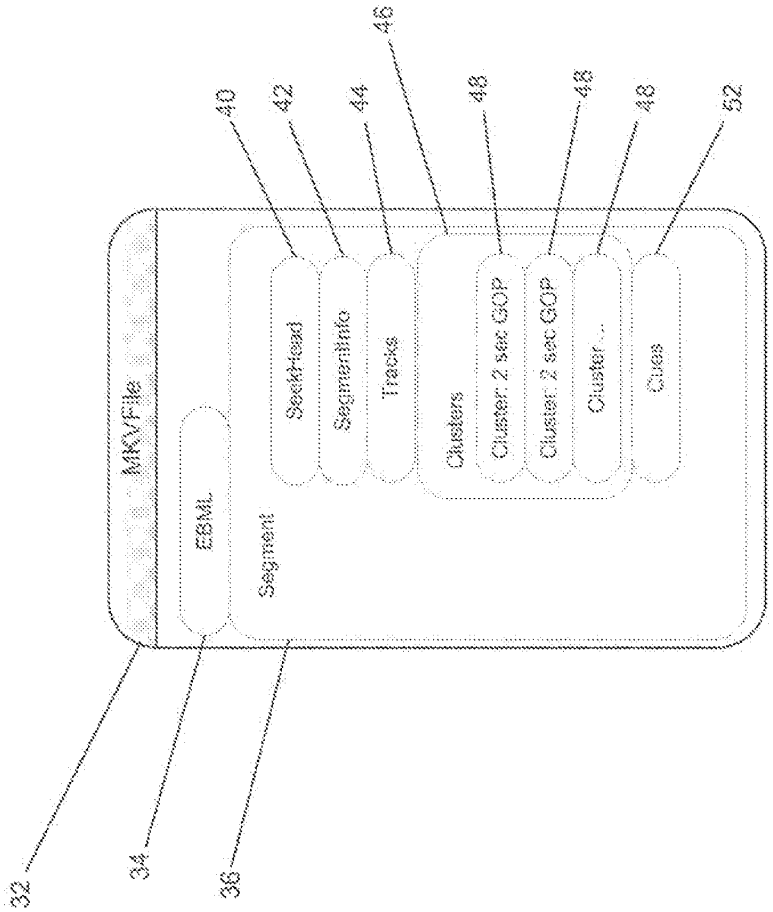

MKVFile
EBML
Segment
SeekHead
SegmentInfo
Tracks
Clusters
Cluster: 2 sec GOP
Cluster: 2 sec GOP
Cluster ...
Cues
32
34
36
40
42
44
46
48
48
48
52

FIG. 3

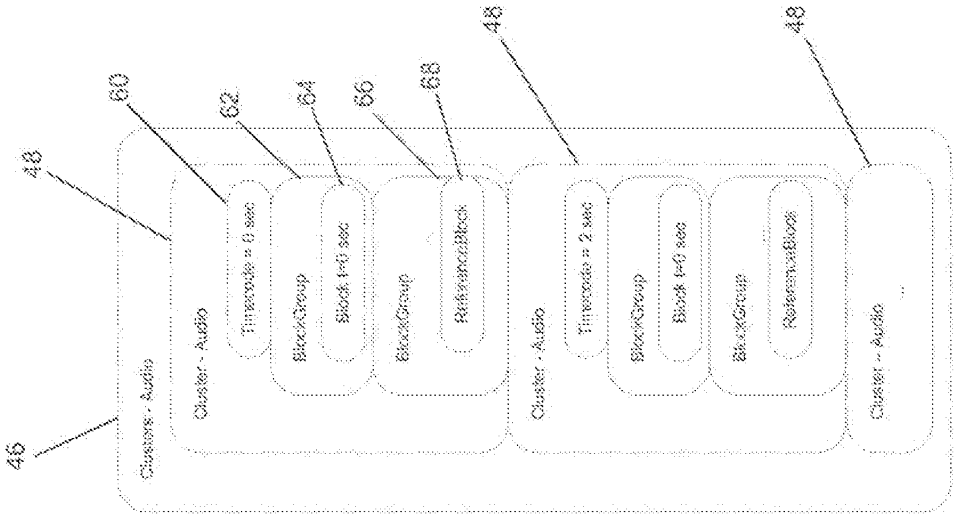

46
48
60
62
64
66
68
48
48
Cluster - Audio
Cluster - Audio
Timecode = 0 sec
BlockGroup
Block I+0 sec
BlockGroup
ReferenceBlock
Cluster - Audio
Timecode = 2 sec
BlockGroup
Block I+0 sec
BlockGroup
ReferenceBlock
Cluster - Audio

FIG. 4b

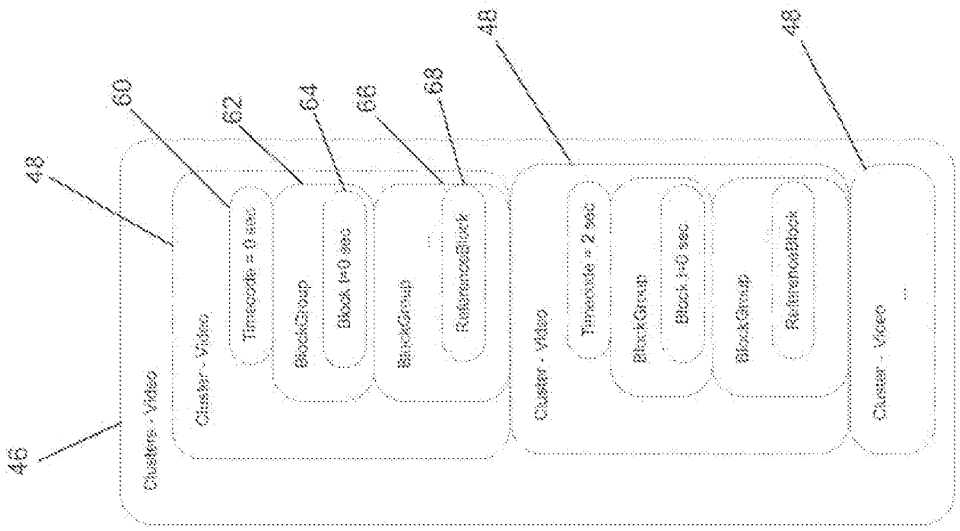

46
48
60
62
64
66
68
48
48
Cluster - Video
Cluster - Video
Timecode = 0 sec
BlockGroup
Block I+0 sec
BlockGroup
ReferenceBlock
Cluster - Video
Timecode = 2 sec
BlockGroup
Block I+0 sec
BlockGroup
ReferenceBlock
Cluster - Video

FIG. 4a

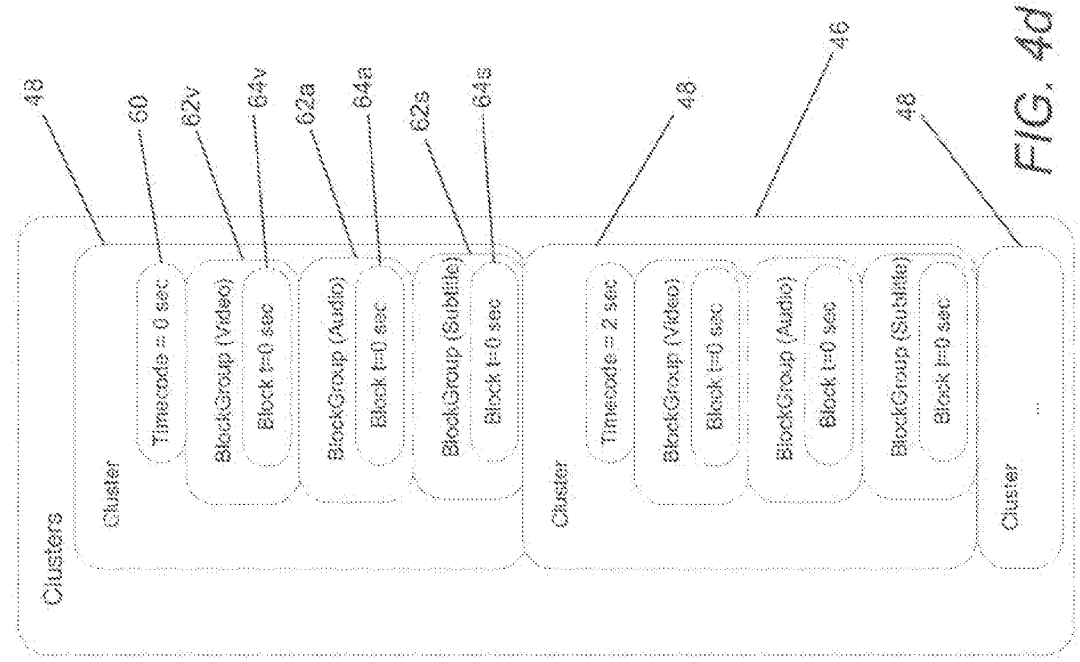

Clusters
Cluster
Timecode = 0 sec
BlockGroup (Video)
Block t=0 sec
BlockGroup (Audio)
Block t=0 sec
BlockGroup (Subtitle)
Block t=0 sec
Cluster
Timecode = 2 sec
BlockGroup (Video)
Block t=0 sec
BlockGroup (Audio)
Block t=0 sec
BlockGroup (Subtitle)
Block t=0 sec
Cluster
48
60
62v
64v
62a
64a
62s
64s
46

FIG. 4d

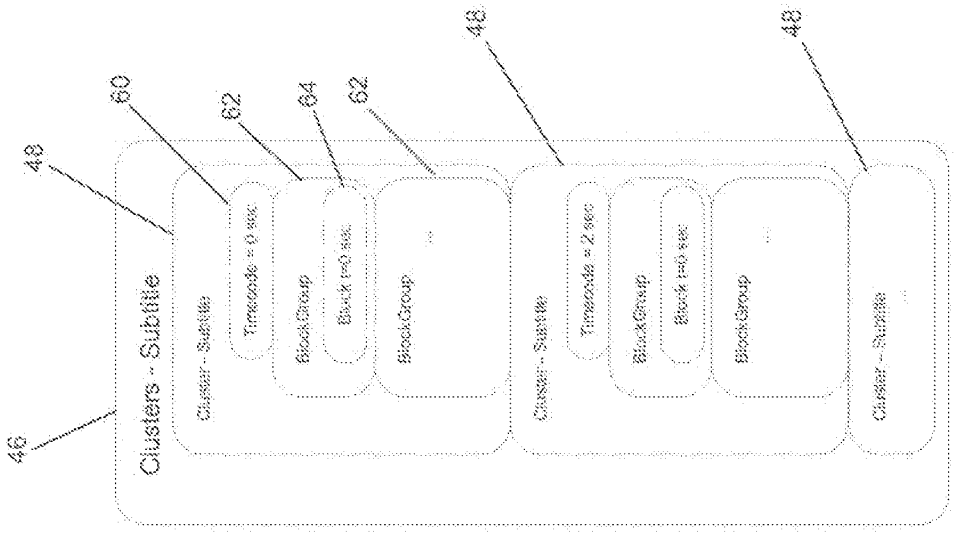

Clusters - Subtitle
Cluster - Subtitle
Timecode = 0 sec
BlockGroup
Block t=0 sec
BlockGroup
Cluster - Subtitle
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
Cluster - Subtitle
46
48
60
62
64
62
48
48

FIG. 4c

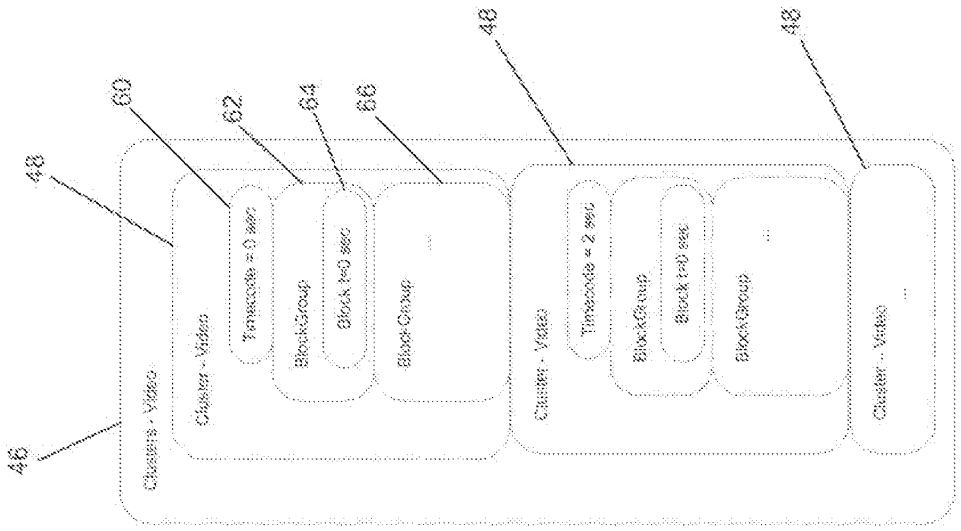

46
48
60
62
64
66
48
48
Cluster - Video
Cluster - Video
Timecode = 0 sec
BlockGroup
Block t=0 sec
Block Group
Cluster - Video
Timecode = 2 sec
BlockGroup
Block t=0 sec
BlockGroup
Cluster - Video

FIG. 4e

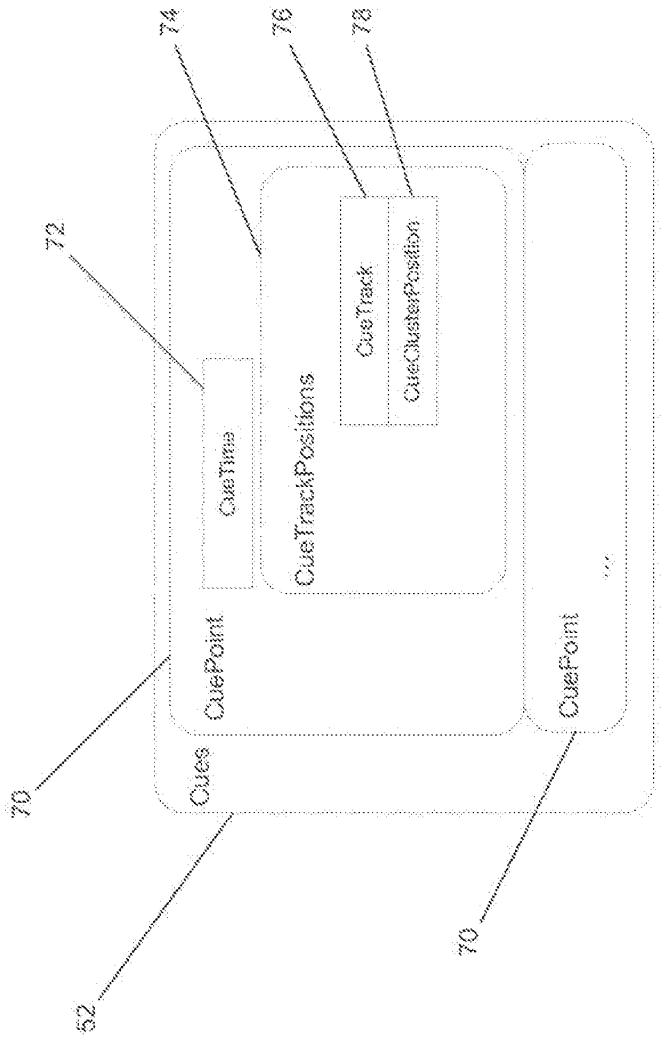

74
76
78
72
CueTime
CueTrackPositions
CueTrack
CueClusterPosition
CuePoint
Cues
CuePoint
...
70
52
70

FIG. 5

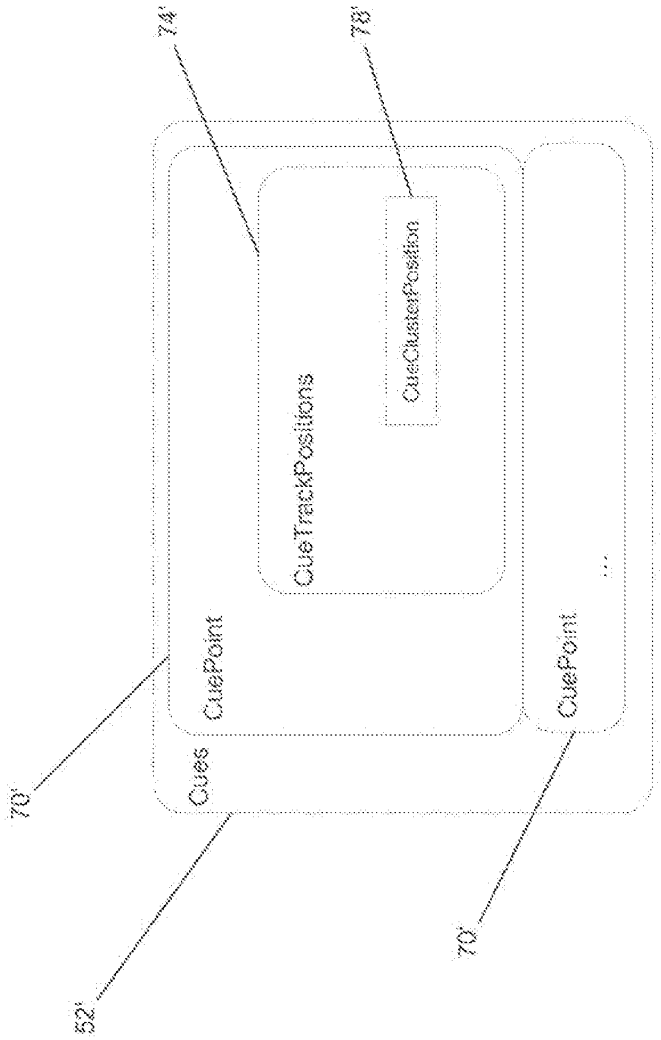

74'
78'
CueClusterPosition
CueTrackPositions
CuePoint
...
CuePoint
Cues
70'
52'
70'

FIG. 5a

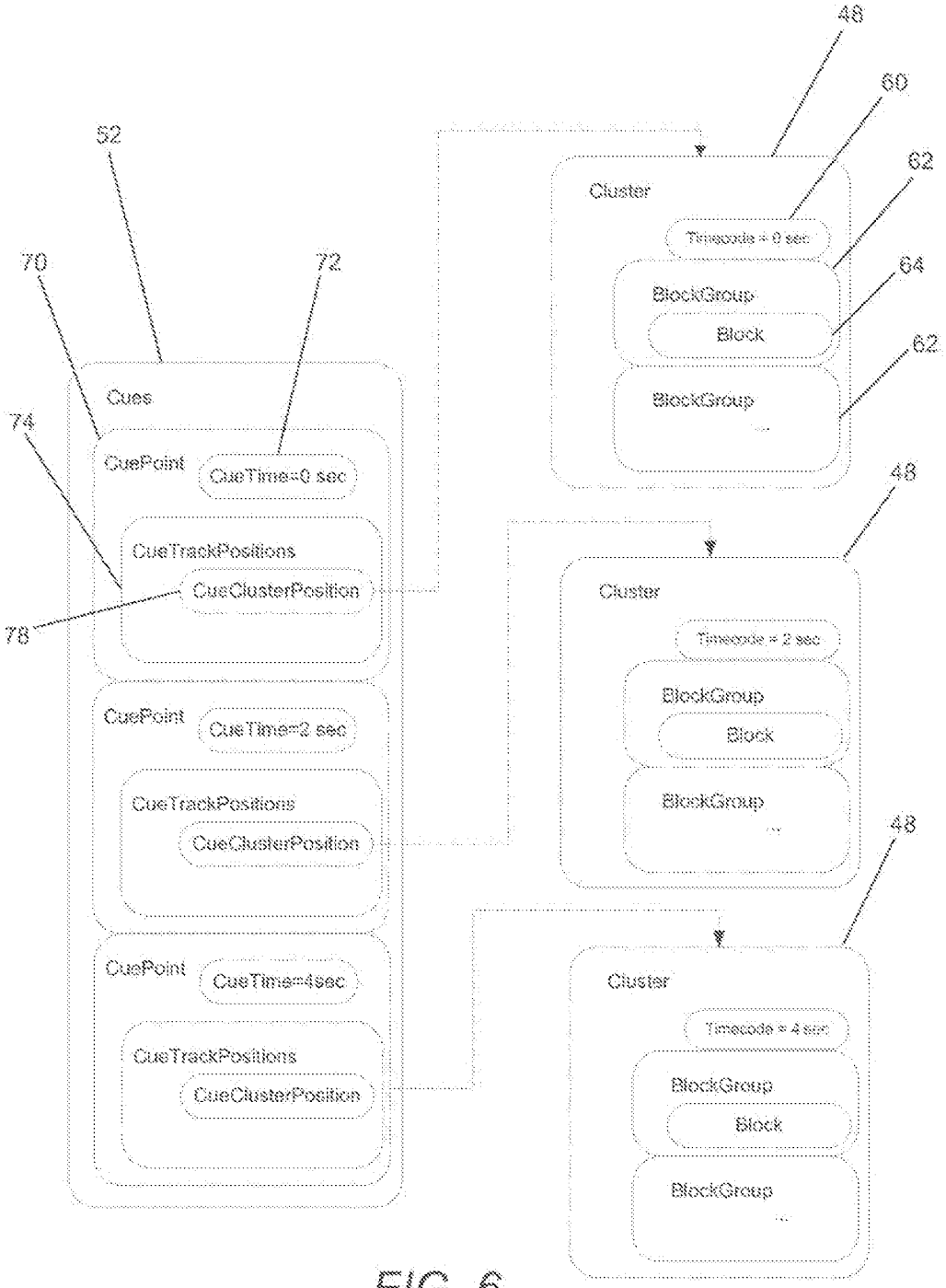

48
60
52
62
Cluster
Timecode = 0 sec
70
72
64
BlockGroup
Block
62
Cues
BlockGroup
74
CuePoint
CueTime=0 sec
48
CueTrackPositions
CueClusterPosition
Cluster
78
Timecode = 2 sec
BlockGroup
Block
CuePoint
CueTime=2 sec
BlockGroup
CueTrackPositions
48
CueClusterPosition
CuePoint
CueTime=4sec
Cluster
CueTrackPositions
Timecode = 4 sec
CueClusterPosition
BlockGroup
Block
BlockGroup

FIG. 6

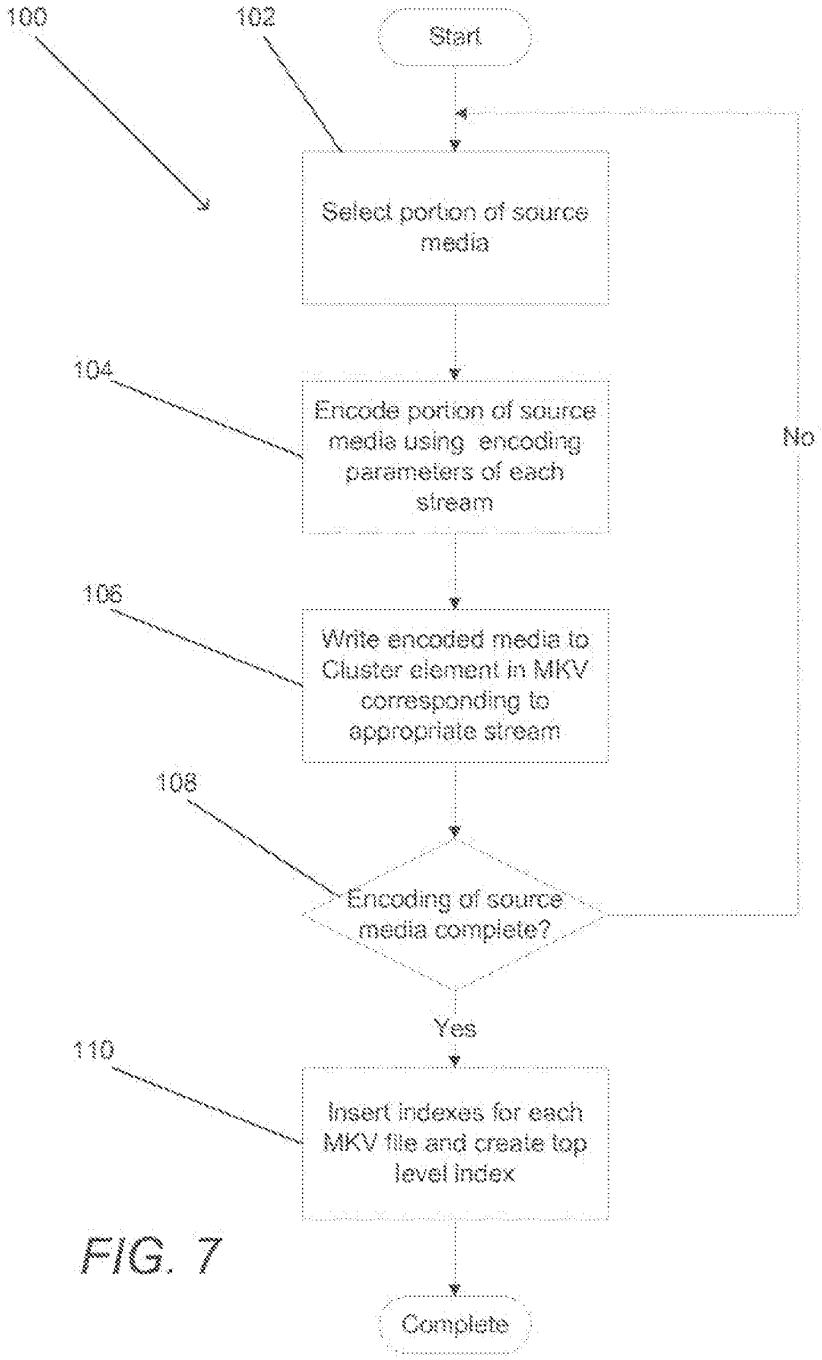

100
102
Start
Select portion of source media
104
Encode portion of source media using encoding parameters of each stream
106
Write encoded media to Cluster element in MKV corresponding to appropriate stream
108
Encoding of source media complete?
No
Yes
110
Insert indexes for each MKV file and create top level index
Complete

FIG. 7

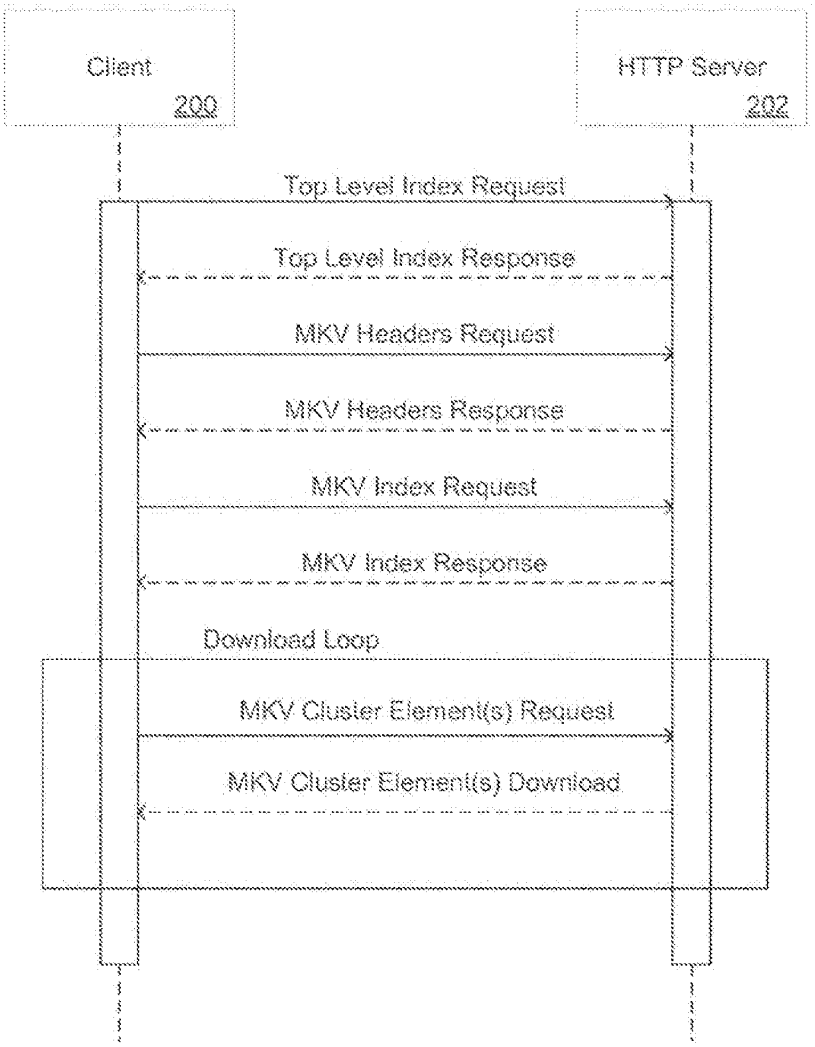

Client
200
HTTP Server
202
Top Level Index Request
Top Level Index Response
MKV Headers Request
MKV Headers Response
MKV Index Request
MKV Index Response
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 8

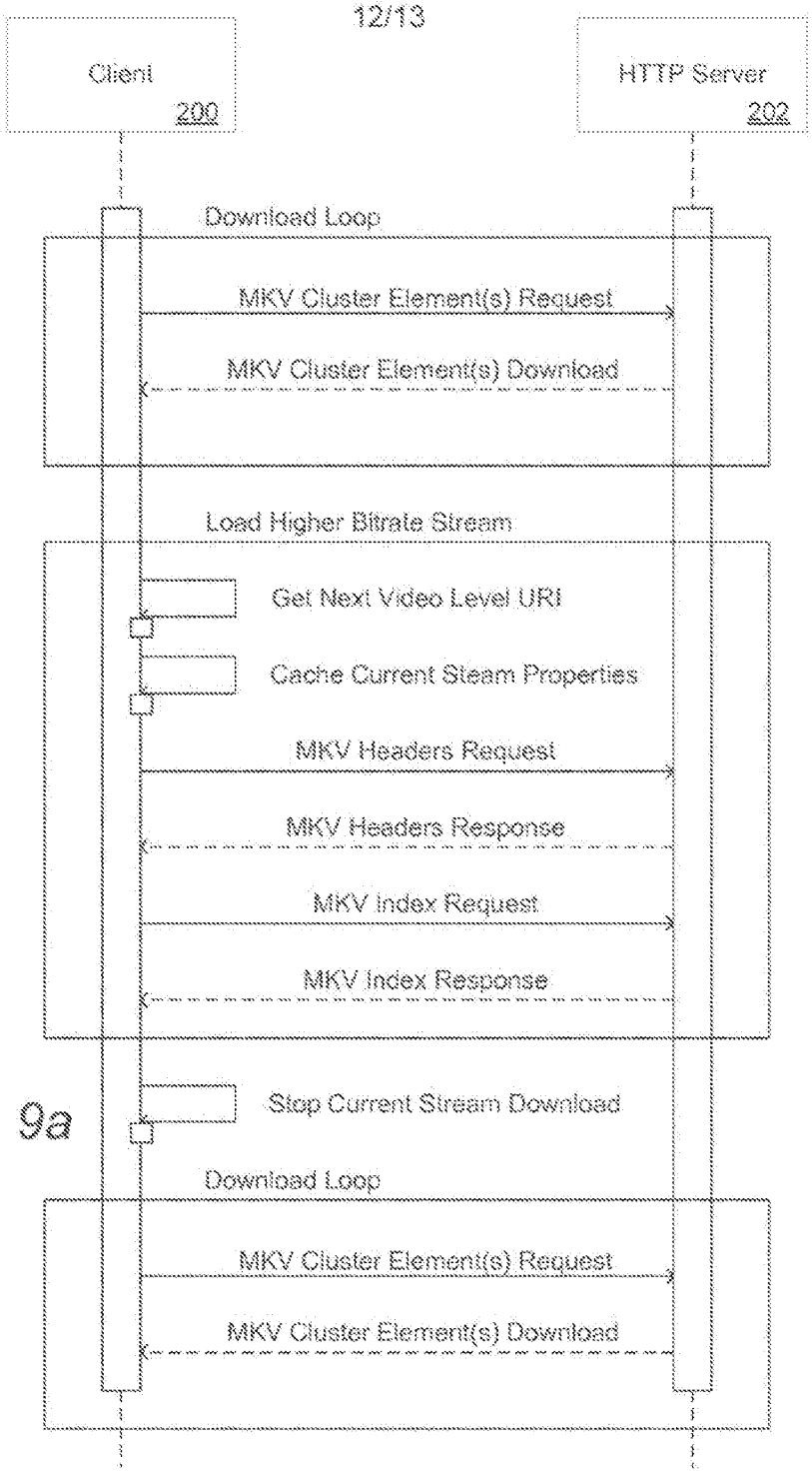

Client
200
HTTP Server
202
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download
Load Higher Bitrate Stream
Get Next Video Level URI
Cache Current Steam Properties
MKV Headers Request
MKV Headers Response
MKV Index Request
MKV Index Response
Stop Current Stream Download
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 9a

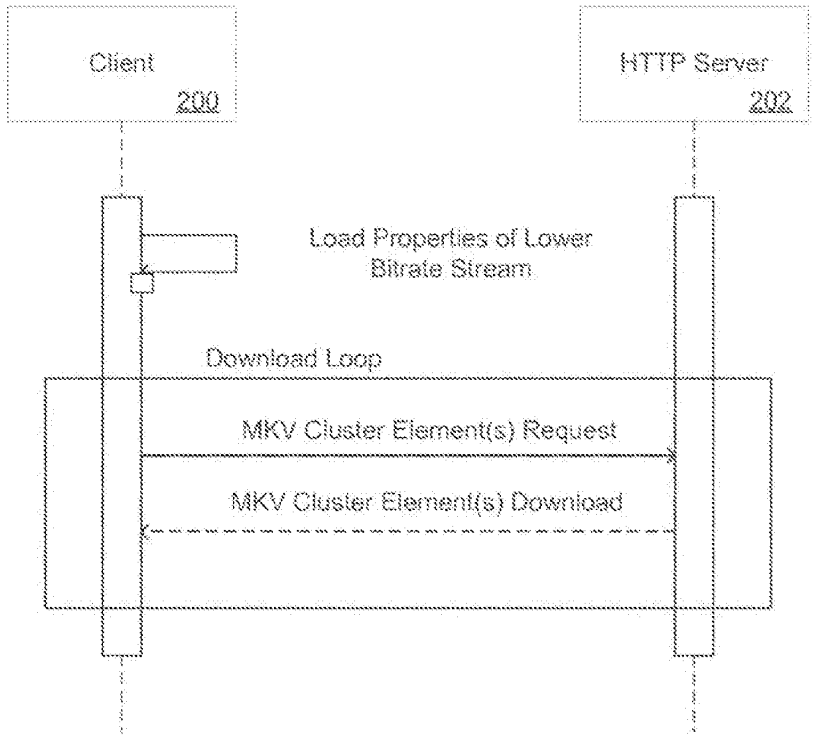

Client
200
HTTP Server
202
Load Properties of Lower Bitrate Stream
Download Loop
MKV Cluster Element(s) Request
MKV Cluster Element(s) Download

FIG. 9b

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 11/66927

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 15/16 (2012.01)
USPC - 709/231
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
USPC: 709/231

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 709/231 | 709/203 | 375/240.28 | 375/E07.011 | 375/E07.154 (keyword limited - see search terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWEST (PGPB, USPT, USOC, EPAB, JPAB); GOOGLE; Google Scholar
Terms: streaming, video, adaptive, server, network, bit, rate, container, frame, block, parse, http, encode, elements, level, layer, cluster, chunk, matroska, xml, stsml.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2010/0189183 A1 (Gu et al.) 29 July 2010 (29.07.2010), entire document, especially abstract, para [0004], [0006], [0008], [0020], [0022], [0024], [0026], [0042], [0046], [0047], [0068]	1-66
Y	US 2007/0168541 A1 (Gupta et al.) 19 July 2007 (19.07.2007), entire document, especially abstract, para [0003], [0004], [0022], [0023], [0028], [0040], [0047], [0055], [0068], [0080], [0083].	1-66
A	US 2007/0083617 A1 (Chakraborti et al.) 12 April 2007 (12.04.2007), entire document, especially abstract, para [0007], [0010], [0012], [0014], [0024], [0030], [0061], [0078], [0102].	1-66

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 April 2012 (03.04.2012)	20 APR 2012
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2005-080204
(43)Date of publication of application : 24.03.2005

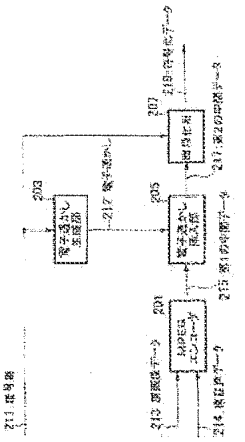
(51)Int.Cl. H04N 1/387
G06T 1/00
G09C 5/00
H04L 9/08
H04N 7/167
H04N 7/24

(21)Application number : 2003-311589
(22)Date of filing : 03.09.2003

(71)Applicant : NEC CORP
(72)Inventor : WATANABE JUNYA

(54) ENCODING APPARATUS AND DECODING APPARATUS USING ENCRYPTION KEY INCLUDED IN DIGITAL WATERMARK, METHODS THEREOF

(57)Abstract:
PROBLEM TO BE SOLVED: To make random access possible while keeping secrecy that decryption cannot be performed without detecting a digital watermark.
SOLUTION: A digital watermark insertion section 205 inserts a digital watermark including an encryption key for decrypting an I picture at a decryption side into a P picture and a B picture in pictures of GOP. An encryption section 207 encrypts the I picture in the pictures of GOP. Thus, at the decryption side, pictures in a certain GOP cannot be decrypted as a whole unless the digital watermark is detected from the P picture and the B picture in the GOP. Furthermore, if the digital watermark can be detected from the P picture and B picture in a certain GOP, the pictures in the GOP can be decrypted as a whole, so that it is not necessary to access the other GOP, thereby coping with random access.



Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0802
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2005-80204
(P2005-80204A)
(43)公開日　平成17年3月24日(2005.3.24)

(51)Int.Cl.[7]		F I			テーマコード（参考）
H04N	1/387	H04N	1/387		5B057
G06T	1/00	G06T	1/00	500B	5C059
G09C	5/00	G09C	5/00		5C064
H04L	9/08	H04L	9/00	601B	5C076
H04N	7/167	H04L	9/00	601E	5J104

審査請求 有　請求項の数 30　OL　（全 21 頁）　最終頁に続く

(21)出願番号　特願2003-311589(P2003-311589)
(22)出願日　平成15年9月3日(2003.9.3)

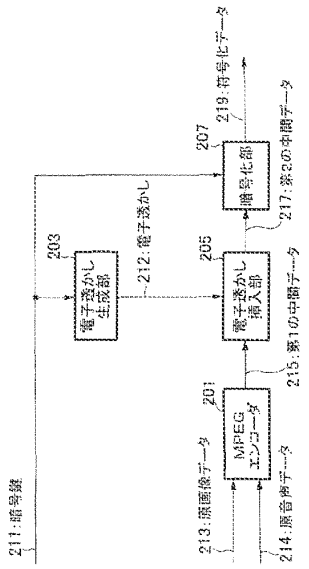
（出願人による申告）平成１５年度、総務省、コンテンツのライフサイクル管理に関するシステムの研究開発委託研究、産業再生法第３０条の適用を受ける特許出願

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100065385
弁理士　山下　穣平
(74)代理人　100122921
弁理士　志村　博
(74)代理人　100130029
弁理士　永井　道雄
(74)代理人　100065385
弁理士　山下　穣平
(72)発明者　渡辺　澤也
東京都港区芝五丁目７番１号　日本電気株式会社内

最終頁に続く

(54)【発明の名称】電子透かしに含めた暗号鍵を用いた符号化装置及び復号化装置並びにそれらの方法

(57)【要約】
【課題】　電子透かしを検出できなければ暗号解読ができないという秘匿性を維持したまま、ランダムアクセスを可能とする。
【解決手段】　電子透かし挿入部２０５は、ＧＯＰのピクチャのうちＰピクチャ及びＢピクチャに復号側でＩピクチャの暗号解読を行うための暗号鍵を含む電子透かしを挿入する。暗号化部２０７は、ＧＯＰのピクチャのうちＩピクチャを暗号化する。こうすることにより、復号側では、あるＧＯＰのＰピクチャ及びＢピクチャから電子透かしを検出できない限り、そのＧＯＰのピクチャ全体を復元することができなくなる。また、あるＧＯＰのＰピクチャ及びＢピクチャから電子透かしを検出できれば、そのＧＯＰのピクチャ全体を復元することができるので、他のＧＯＰをアクセスする必要が無くなり、ランダムアクセスに対応することができる。
【選択図】　図７



Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0803
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

【特許請求の範囲】
【請求項１】
暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成手段と、
データの第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入手段と、
前記データの第２の部分を前記暗号鍵により暗号化する暗号化手段と、
を備える電子透かしに含めた暗号鍵を用いた符号化装置において、
前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されたことを条件として、復元できるデータであることを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置。
【請求項２】
請求項１に記載の電子透かしに含めた暗号鍵を用いた符号化装置において、
前記データの第２の部分は少なくともフレーム内圧縮データを含むことを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置。
【請求項３】
請求項１に記載の電子透かしに含めた暗号鍵を用いた符号化装置において、
前記データを圧縮する圧縮手段を更に備え、前記データの第２の部分は圧縮された後に暗号化されることを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置。
【請求項４】
請求項１に記載の電子透かしに含めた暗号鍵を用いた符号化装置において、
前記電子透かし挿入手段は、前記データの第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入することを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置。
【請求項５】
データの第１の部分から電子透かしを検出する電子透かし検出手段と、
前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出手段と、
前記データの第２の部分を前記暗号鍵で暗号解読する暗号解読手段と、
を備える電子透かしに含めた暗号鍵を用いた復号化装置において、
前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されていることを条件として、復元できるデータであることを特徴とする電子透かしに含めた暗号鍵を用いた復号化装置。
【請求項６】
請求項５に記載の電子透かしに含めた暗号鍵を用いた復号化装置において、
前記データの第２の部分は少なくともフレーム内圧縮データを含むことを特徴とする電子透かしに含めた暗号鍵を用いた暗号復号化装置。
【請求項７】
請求項５に記載の電子透かしに含めた暗号鍵を用いた復号化装置において、
前記データを伸張する伸張手段を更に備え、前記データの第２の部分は暗号復号された後に伸張されることを特徴とする電子透かしに含めた暗号鍵を用いた復号化装置。
【請求項８】
暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成手段と、
前記データのフレーム内又はフィールド内の第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入手段と、
前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵により暗号化する暗号化手段と、
を備えることを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置。
【請求項９】
請求項８に記載の電子透かしに含めた暗号鍵を用いた符号化装置において、
前記電子透かし挿入手段は、前記データのフレーム内又はフィールド内の第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入することを特

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0804
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

徴とする電子透かしに含めた暗号鍵を用いた符号化装置。
【請求項１０】
データのフレーム内又はフィールド内の第１の部分から電子透かしを検出する電子透かし検出手段と、
前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出手段と、
前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵で暗号解読する暗号解読手段と、
を備えることを特徴とする電子透かしに含めた暗号鍵を用いた復号化装置。
【請求項１１】
暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成ステップと、
データの第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入ステップと、
前記データの第２の部分を前記暗号鍵により暗号化する暗号化ステップと、
を備える電子透かしに含めた暗号鍵を用いた符号化方法において、
前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されたことを条件として、復元できるデータであることを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法。
【請求項１２】
請求項１１に記載の電子透かしに含めた暗号鍵を用いた符号化方法において、
前記データの第２の部分は少なくともフレーム内圧縮データを含むことを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法。
【請求項１３】
請求項１１に記載の電子透かしに含めた暗号鍵を用いた符号化方法において、
前記データを圧縮する圧縮ステップを更に備え、前記データの第２の部分は圧縮された後に暗号化されることを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法。
【請求項１４】
請求項１１に記載の電子透かしに含めた暗号鍵を用いた符号化方法において、
前記電子透かし挿入ステップは、前記データの第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入することを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法。
【請求項１５】
データの第１の部分から電子透かしを検出する電子透かし検出ステップと、
前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出ステップと、
前記データの第２の部分を前記暗号鍵で暗号解読する暗号解読ステップと、
を備える電子透かしに含めた暗号鍵を用いた復号化方法において、
前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されていることを条件として、復元できるデータであることを特徴とする電子透かしに含めた暗号鍵を用いた復号化方法。
【請求項１６】
請求項１５に記載の電子透かしに含めた暗号鍵を用いた復号化方法において、
前記データの第２の部分は少なくともフレーム内圧縮データを含むことを特徴とする電子透かしに含めた暗号鍵を用いた暗号復号化方法。
【請求項１７】
請求項１５に記載の電子透かしに含めた暗号鍵を用いた復号化方法において、
前記データを伸張する伸張ステップを更に備え、前記データの第２の部分は暗号復号された後に伸張されることを特徴とする電子透かしに含めた暗号鍵を用いた復号化方法。
【請求項１８】
暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成ステップと、
データのフレーム内又はフィールド内の第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入ステップと、

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0805
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵により暗号化する暗号化ステップと、
を備えることを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法。
【請求項１９】
請求項１８に記載の電子透かしに含めた暗号鍵を用いた符号化方法において、
前記電子透かし挿入ステップは、前記データのフレーム内又はフィールド内の第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入することを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法。
【請求項２０】
データのフレーム内又はフィールド内の第１の部分から電子透かしを検出する電子透かし検出ステップと、
前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出ステップと、
前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵で暗号解読する暗号解読ステップと、
を備えることを特徴とする電子透かしに含めた暗号鍵を用いた復号化方法。
【請求項２１】
暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成ステップと、
データの第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入ステップと、
前記データの第２の部分を前記暗号鍵により暗号化する暗号化ステップと、
を備える電子透かしに含めた暗号鍵を用いた符号化方法をコンピュータに実行させるためのプログラムにおいて、
前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されたことを条件として、復元できるデータであることを特徴とするプログラム。
【請求項２２】
請求項２１に記載のプログラムにおいて、
前記データの第２の部分は少なくともフレーム内圧縮データを含むことを特徴とするプログラム。
【請求項２３】
請求項２１に記載のプログラムにおいて、
前記方法は、前記データを圧縮する圧縮ステップを更に備え、前記データの第２の部分は圧縮された後に暗号化されることを特徴とするプログラム。
【請求項２４】
請求項２１に記載のプログラムにおいて、
前記電子透かし挿入ステップは、前記データの第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入することを特徴とするプログラム。
【請求項２５】
データの第１の部分から電子透かしを検出する電子透かし検出ステップと、
前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出ステップと、
前記データの第２の部分を前記暗号鍵で暗号解読する暗号解読ステップと、
を備える電子透かしに含めた暗号鍵を用いた復号化方法をコンピュータに実行させるためのプログラムにおいて、
前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されていることを条件として、復元できるデータであることを特徴とするプログラム。
【請求項２６】
請求項２５に記載のプログラムにおいて、
前記データの第２の部分は少なくともフレーム内圧縮データを含むことを特徴とするプログラム。
【請求項２７】
請求項２５に記載のプログラムにおいて、

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0806
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

前記方法は、前記データを伸張する伸張ステップを更に備え、前記データの第２の部分は暗号復号された後に伸張されることを特徴とするプログラム。
【請求項２８】
暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成ステップと、
データのフレーム内又はフィールド内の第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入ステップと、
前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵により暗号化する暗号化ステップと、
を備えることを特徴とする電子透かしに含めた暗号鍵を用いた符号化方法をコンピュータに実行させるためのプログラム。
【請求項２９】
請求項２８に記載のプログラムにおいて、
前記電子透かし挿入ステップは、前記データのフレーム内又はフィールド内の第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入することを特徴とするプログラム。
【請求項３０】
データのフレーム内又はフィールド内の第１の部分から電子透かしを検出する電子透かし検出ステップと、
前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出ステップと、
前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵で暗号解読する暗号解読ステップと、
を備えることを特徴とする電子透かしに含めた暗号鍵を用いた復号化方法をコンピュータに実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、デジタルデータに電子透かしを挿入する方法及び装置並びにデジタルデータから電子透かしを検出する方法及び装置に関し、特に、デジタル画像データに電子透かしを挿入する方法及び装置並びにデジタル画像データから電子透かしを検出する方法及び装置に関する。
【０００２】
また、本発明はデジタルデータを暗号化する方法及び装置並びに暗号化されたデジタルデータを復号する方法及び装置に関し、特に、デジタル画像データを暗号化する方法及び装置並びに暗号化されたデジタル画像データを復号する方法及び装置に関する。
【背景技術】
【０００３】
近年、画像データや音声データは、デジタル化されてから蓄積、伝送、配布等がされるようになってきた。一方、デジタル化に伴うデータの違法な複製が大きな問題となっている。電子透かしデータの挿入・検出技術は、そのような違法な複製を防止する技術として注目を浴びており、実現化に向けて検討が進んでいる。また、電子透かしデータとは別にデータやプログラム等の改竄を防ぐ技術として暗号化方式がある。これは、データやプログラム自体をある暗号キーにより暗号化し、その暗号キーがなければそのデータやプログラムがデータやプログラムとして成り立たないようにする技術である。
【０００４】
しかし、暗号化技術は、一度その暗号キーを解読されてしまうと、暗号化したデータも簡単にアクセスできてしまうと言う弱点があり、ＤＶＤ(Digital Versatile Disk)に施されているＣＳＳ(Contents Scrambling System)のように、同一の暗号鍵（単一の暗号鍵又は１組の暗号鍵）で全てのＤＶＤコンテンツを暗号化した場合、その暗号鍵を解読されると、全てのＤＶＤコンテンツの暗号を解いて、不正にコピー出来てしまうと言うことになる。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0807
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

【０００５】
そこで、各コンテンツをコンテンツ毎の暗号鍵を用いて暗号化することによりＣＳＳの問題点を解決できる。しかし、暗号鍵をコンテンツとは別のルートでＤＶＤの購買者に送付すると、購買者はコンテンツ毎に異なった暗号鍵を再生機又は再生ソフトに設定しなければならず、購買者の操作が煩わしい。また、暗号鍵をＭＰＥＧ(Moving Picture Experts Group)データの特定の領域に挿入すると、不正者が容易にＭＰＥＧデータから暗号鍵を抜き取り、抜き取った暗号鍵により暗号化されていないＭＰＥＧデータを取り出し、暗号化されていないＭＰＥＧデータを複製することが可能となってしまう。

【０００６】
特許文献１に記載の技術は、電子透かしを画像データに挿入すると共に、電子透かしをスクランブルのための座標変換データの一部として使用して、画像ブロック単位で画像をスクランブルすることにより、再生側で、画像データに挿入された電子透かしと同一の電子透かしを用意している場合に限り、スクランブルを解除できるようにしたものである。しかし、この技術では、画像データの暗号化が行われておらず、画像ブロック単位でスクランブルが行われているだけであるので、スクランブルを解かなくても、どのフレームからも電子透かしを検出することができ、電子透かしを隠蔽することが全く出来ない。また、画像データを圧縮しようとした際、画像ブロック単位でスクランブルが行われているので、空間の連続性が無くなり、画像ブロックの動きベクトルを検出できず、動き補償フレーム間予測符号化を用いることが出来なくなってしまい、従って、高能率に圧縮することが出来なくなってしまう。

【０００７】
そこで、本願の出願人は、電子透かし技術と、暗号化技術のそれぞれの特徴を活かし、よりセキュアなデジタルコンテンツの再生制限をすること可能とする、電子透かしに含めた暗号鍵を用いた符号化装置及びその方法並びに電子透かしに含めた暗号鍵を用いた復号化装置及びその方法を発明した。この発明は、例えば、特許文献２に記載されているように、第１の暗号鍵を含んだ第１の電子透かしを生成する生成手段と、データの第１の部分に前記第１の暗号鍵を含んだ前記第１の電子透かしを挿入する電子透かし挿入手段と、前記データの第２の部分を前記第１の暗号鍵で暗号化する暗号化手段と、を備えることを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置及びデータの第１の部分より第１の電子透かしを検出する電子透かし検出手段と、前記第１の電子透かしより第１の暗号鍵を抽出する暗号鍵抽出手段と、前記データの第２の部分を前記第１の暗号鍵で暗号解読する暗号解読手段と、を備えることを特徴とする電子透かしに含めた暗号鍵を用いた復号化装置である。この符号化装置において、前記電子透かし生成手段は、第ｎ（ｎは１より大きい整数）の暗号鍵を含んだ第ｎの電子透かしを生成し、前記電子透かし挿入手段は、第ｎの部分が前記暗号化手段により第（ｎ－１）の暗号鍵で暗号化される前に、前記第ｎの部分に前記第ｎの電子透かしを挿入し、前記暗号化手段は、前記データの第（ｎ＋１）の部分を前記第ｎの暗号鍵で暗号化してもよい。また、この復号化装置において、前記電子透かし検出手段は、前記暗号解読手段により第（ｎ－１）（ｎは１より大きい整数）の暗号鍵で暗号解読された第ｎの部分より第ｎの電子透かしを検出し、前記暗号鍵抽出手段は、前記第ｎの電子透かしより第ｎの暗号鍵を抽出し、前記暗号解読手段は、前記データの第（ｎ＋１）の部分を前記第ｎの暗号鍵で暗号解読してもよい。

【０００８】
ここで、他の従来技術について説明すると、特許文献３に記載の発明は、ＳＭＴＰヘッダ、伝送情報及びイメージデータより構成されるデータセットに対し、イメージデータを電子透かしで認証処理し、電子透かしを復号するための復号鍵及び伝送情報を強い暗号の暗号鍵で暗号化するものである。

【０００９】
特許文献４に記載の発明は、動画再生したときには視認できないが、静止画再生したときには視認できる電子透かしを動画の一部のフレームに挿入するものである。

【特許文献１】特開平１１－３１７８５９号公報

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0808
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

【特許文献２】特開２００２－２３２４１２号公報
【特許文献３】特開平１１－８８３１９号公報
【特許文献４】特開平１１－１３６６１８号公報
【発明の開示】
【発明が解決しようとする課題】
【００１０】
しかし、特許文献２に記載されている発明には以下の問題点がある。すなわち、復号化装置においては、データの第１の部分から電子透かしを検出しなければ、データの第２の部分の暗号解読をすることができず、データの第２の部分の暗号解読及びデータの第２の部分からの電子透かしの検出ができなければデータの第３の部分の暗号解読をすることができず、データの第３の部分の暗号解読及びデータの第３の部分からの電子透かしの検出ができなければデータの第４の部分の暗号解読ができなく、この様なことが続く。従って、コンテンツを先頭から末尾まで順を追って再生する場合には、問題が生ずることがないが、コンテンツをランダムアクセスする場合に、暗号解読ができない。特に、ランダムアクセスすることができるＤＶＤへの適用が多いことを考慮すると、それは大きな問題となる。
【００１１】
そこで本発明は、電子透かしを検出できなければ暗号解読ができないという秘匿性を維持したまま、ランダムアクセスを可能とする電子透かしに含めた暗号鍵を用いた符号化装置及びその方法並びに電子透かしに含めた暗号鍵を用いた復号化装置及びその方法を提供することを目的とする。
【課題を解決するための手段】
【００１２】
本発明の第１の観点によれば、暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成手段と、データの第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入手段と、前記データの第２の部分を前記暗号鍵により暗号化する暗号化手段と、を備える電子透かしに含めた暗号鍵を用いた符号化装置において、前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されたことを条件として、復元できるデータであることを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置が提供される。
【００１３】
本発明の第１の観点による電子透かしに含めた暗号鍵を用いた符号化装置において、前記データの第２の部分は少なくともフレーム内圧縮データを含んでいてもよい。
【００１４】
本発明の第１の観点による電子透かしに含めた暗号鍵を用いた符号化装置は、前記データを圧縮する圧縮手段を更に備え、前記データの第２の部分は圧縮された後に暗号化されてもよい。
【００１５】
本発明の第１の観点による電子透かしに含めた暗号鍵を用いた符号化装置において、前記電子透かし挿入手段は、前記データの第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入してもよい。
【００１６】
本発明の第２の観点によれば、データの第１の部分から電子透かしを検出する電子透かし検出手段と、前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出手段と、前記データの第２の部分を前記暗号鍵で暗号解読する暗号解読手段と、を備える電子透かしに含めた暗号鍵を用いた復号化装置において、前記データの第１の部分は、データ伸張において、前記データの第２の部分が復元されていることを条件として、復元できるデータであることを特徴とする電子透かしに含めた暗号鍵を用いた復号化装置が提供される。
【００１７】
本発明の第２の観点による電子透かしに含めた暗号鍵を用いた復号化装置において、前

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0809
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

記データの第２の部分は少なくともフレーム内圧縮データを含んでいてもよい。

【００１８】

本発明の第２の観点による電子透かしに含めた暗号鍵を用いた復号化装置は、前記データを伸張する伸張手段を更に備え、前記データの第２の部分は暗号復号された後に伸張されてもよい。

【００１９】

本発明の第３の観点によれば、暗号鍵の全部又は一部を含んだ電子透かしを生成する電子透かし生成手段と、データのフレーム内又はフィールド内の第１の部分に前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入する電子透かし挿入手段と、前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵により暗号化する暗号化手段と、を備えることを特徴とする電子透かしに含めた暗号鍵を用いた符号化装置が提供される。

【００２０】

本発明の第３の観点による電子透かしに含めた暗号鍵を用いた符号化装置において、前記電子透かし挿入手段は、前記データのフレーム内又はフィールド内の第２の部分の全部又は一部にも前記暗号鍵の全部又は一部を含んだ前記電子透かしを挿入してもよい。

【００２１】

本発明の第４の観点によれば、データのフレーム内又はフィールド内の第１の部分から電子透かしを検出する電子透かし検出手段と、前記電子透かしより暗号鍵の全部又は一部を抽出する暗号鍵抽出手段と、前記データのフレーム内又はフィールド内の第２の部分を前記暗号鍵で暗号解読する暗号解読手段と、を備えることを特徴とする電子透かしに含めた暗号鍵を用いた復号化装置が提供される。

【発明の効果】

【００２２】

データの第２の部分が復元されたことを条件として復元できるデータの第１の部分にデータの第２の部分の暗号解読をするための暗号鍵を含んだ電子透かしを挿入するので、データの第１の部分の電子透かしを検出することができなければ、データの第１の部分及びデータの第２の部分の両方を復元することができない。従って、データの一部であるデータの第２の部分のみを暗号化するのみで、データの秘匿性を高めることができる。従って、データの一部であるデータの第２の部分以外の部分であるデータの第１の部分に、暗号化されてしまうと挿入することが不可能な電子透かしを挿入することとしている。こうすることにより、データの第１の部分又はデータの第２の部分でない他の領域のデータに暗号鍵を含んだ電子透かしを挿入する必要が無くなるので、データの第１の部分及びデータの第２の部分を復元するために他の領域にアクセスする必要が無くなり、ランダムアクセスに対応することができる。

【００２３】

データの第２の部分を暗号化しているので、その第２の部分の暗号解読ができなければ、他の部分が暗号化されていなくても、復号化データは、意味を持たないものとなる。従って、データの一部であるデータの第２の部分以外の部分であるデータの第１の部分に、暗号解読をするための暗号鍵を含んだ電子透かしを挿入することとしている。こうすることにより、データの第１の部分又はデータの第２の部分でない他の領域のデータに暗号鍵を含んだ電子透かしを挿入する必要が無くなるので、データの第１の部分及びデータの第２の部分を意味のあるものに復元するために他の領域にアクセスする必要が無くなり、ランダムアクセスに対応することができる。

【発明を実施するための最良の形態】

【００２４】

以下、図面を参照して本発明の実施形態について詳細に説明する。

【００２５】

〔実施形態１〕

まず、実施形態１による符号化装置について説明する。図１は、実施形態１による符号化装置の構成を示すブロック図である。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0810
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

【００２６】
図１を参照すると、実施形態１による符号化装置は、電子透かし生成部１０１、電子透かし挿入部１０３及び暗号化部１０５を備える。
【００２７】
電子透かし生成部１０１は、暗号鍵１１１の全部又は一部を含んだ電子透かし１１２を生成する。ここで、電子透かし１１２が暗号鍵１１１を含むとは、電子透かし１１２の各成分に暗号鍵の全部又は一部の各数値又は文字が対応付けられることをいう。暗号鍵１１１は、原データのコンテンツ毎に異なったものであることが好ましい。こうすることにより、あるコンテンツに対応した暗号鍵がハッキングされても、他のコンテンツを暗号解読することができない。また、暗号鍵１１１は、復号側でも用いられることを考慮し、共通鍵である。また、電子透かしを各フレームから検出することができるならば、暗号鍵１１１をフレーム毎に変化させてもよい。電子透かしを検出するために、数フレーム必要なのであれば、暗号鍵を数フレーム毎に変化させてもよい。こうすることにより、暗号鍵のハッキングに多大な時間を要することとなり、コンテンツの秘匿性を高めることができる。
【００２８】
電子透かし挿入部１０３は、原データ１１３に電子透かし１１２を挿入し、電子透かしが挿入された中間データ１１５を生成する。原データ１１３は、実施形態では、例えば、動画像データである。電子透かし挿入部１０３の詳細については後述する。
【００２９】
原データ１１３及び中間データ１１５のフィールド又はフレーム（以下、フィールド及び１以上のフレームを含めて単にフレームという。）の構成の模式図を図２に示す。図２に示す原データ１１３等の番号が付された四角は、画素又はブロックを示すが、以下、画素及びブロックを含めて単にブロックという。図２に示す中間データ１１５において、数字が円で囲まれたブロックに電子透かしが挿入されており、その他のブロックには電子透かしは挿入されていない。図２には、２種類の中間データ１１５（１１５Ａ及び１１５Ｂ）を示す。一方の種類の中間データ１１５Ａは、一部のブロックに電子透かしが挿入されたものであり、他方の種類の中間データ１１５Ｂは、全部のブロックに電子透かしが挿入されたものである。電子透かし挿入部１０３は、どちらの種類の中間データ１１５を生成しても良い。
【００３０】
暗号化部１０５は、中間データ１１５を暗号鍵１１１で暗号化し、符号化データ１１７を生成する。図２に符号化データ１１７（１１７Ａ及び１１７Ｂ）の２種類の模式図を示す。図２に示す符号化データ１１７において、数字にバツを付しているブロックが暗号化され、その他のブロックは暗号化されない。暗号化されていないブロックは、原データ１１３とほぼ同様の内容を有するが、一部のブロックが暗号化されているので、符号化データ１１７は、フレーム全体で見ると認識できないデータとなっているので、秘匿性が得られている。
【００３１】
電子透かし挿入部１０３が上述した第１の種類の中間データ１１５を生成する場合には、電子透かしが挿入されたブロックは暗号化されない。これは、ブロックに電子透かしが挿入された後に、そのブロックが暗号化されると、そのブロックから電子透かしを検出できなくなるからである。
【００３２】
しかし、全部のブロックに電子透かしが挿入された後に、一部のブロックが暗号化された場合には、暗号化されたブロックから電子透かしを検出することができなくなるが、暗号化されていないブロックから電子透かしを検出することができる。また、電子透かしを挿入したブロックを暗号化し、暗号解読することは問題なくできる。従って、あるブロックに電子透かしを挿入し、そのブロックを暗号化しても何ら支障はない。他方、全部のブロックに電子透かしを挿入する場合には、電子透かしを挿入するかしないかについての判断部を設ける必要が無く、電子透かし挿入部１０３を簡素化できる。従って、電子透かし

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0811
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

挿入部１０３の簡素化を目的として、電子透かし挿入部１０３が全部のブロックに電子透かしを挿入した上述の第２の種類の中間データ１１５を生成するようにしてもよい。

【００３３】

また、電子透かし挿入部１０３は、暗号化されないブロックのうちの全てのブロック及び暗号化されるブロックのうちの一部のブロックに電子透かしを挿入しても良い。更に、電子透かしの検出が不可能とならない範囲で、暗号化されないブロックのうちの一部のブロックのみに電子透かしを挿入しても良い。

【００３４】

次に、電子透かし挿入部１０３の詳細について図３を参照して説明する。

【００３５】

図３を参照すると、電子透かし挿入部１０３は、離散コサイン変換器１０３－１、電子透かし保持部１０３－２、挿入部１０３－３及び逆離散コサイン変換器１０３－４を備える。

【００３６】

離散コサイン変換器１０３－１は、空間領域の原画像データｘ（ｊ）に対して、例えば８画素×８画素の２次元の離散コサイン変換を行い、周波数領域のＤＣＴ係数ｆ（ｉ）を出力する。ここで、ｊは、２次元から１次元に並べられた後の画素の番号を表し、ｉは、２次元から１次元に並べられた後のＤＣＴ係数の番号を表す。

【００３７】

電子透かしデータ保持部１０３－２は、電子透かし１１２

ｗ（１）、ｗ（２）、・・・、ｗ（ｎ）

を保持する。なお、電子透かしデータは平均０、分散１の正規分布に従う。

【００３８】

挿入部１０３－３は、以下のことを行う。すなわち、ＤＣＴ係数ｆ（ｉ）と電子透かしデータｗ（ｉ）を基に、電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）を、

Ｆ（ｉ）＝ｆ（ｉ）＋α×ａｖｇ（ｆ（ｉ））×ｗ（ｉ）

の計算式により各ｉについて計算する。ここで、αはスケーリング要素であり、ａｖｇ（ｆ（ｉ））はｆ（ｉ）の近傍３点の絶対値の平均を取った部分平均である。そして、Ｆ（ｉ）を出力する。

【００３９】

逆離散コサイン変換器１０３－４は、電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）に対して、逆離散コサイン変換を行い、空間領域の電子透かし挿入済画像データＸ（ｊ）を出力する。

【００４０】

次に、実施形態１による復号化装置について説明する。図４は、実施形態１による復号化装置の構成を示すブロック図である。

【００４１】

図４を参照すると、実施形態１による復号化装置は、電子透かし検出部１２１、遅延部１２２、暗号鍵抽出部１２３及び暗号解読部１２５を備える。

【００４２】

電子透かし検出部１２１は、符号化データ１１７から電子透かし１３１を検出する。電子透かし検出部１２１の詳細については後述する。暗号鍵抽出部１２３は、電子透かし１３１から暗号鍵１３３を抽出する。ここで、暗号鍵１３３は電子透かし生成部１０１が電子透かしに挿入した暗号鍵であり、暗号鍵１１１の全部又は一部である。電子透かし１３１が電子透かし１１１の一部である場合には、暗号鍵抽出部１２３が抽出した暗号鍵１３１に電子透かし１１１の残りの部分を合わせる。電子透かし１１１の残りの部分は、例えば、ユーザがインターネットを介してコンテンツ提供者より入手して、復号化装置に入力する。このように、電子透かし１３３は電子透かし１１１の一部である場合もあるが、以降の説明では、説明を簡単にするために、電子透かし１１１と電子透かし１３３が同一であるとする。遅延部１２２は、符号化データ１１７を、符号化データ１１７から電子透か

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0812
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

し検出部１２１及び暗号鍵抽出部１２３により暗号鍵１３３を得るのに必要な時間だけ遅延させることにより、暗号鍵１３３が得られてから、その暗号鍵１３３を含んでいたフレームのデータが暗号解読部１２５に供給されるようにする。暗号解読部１２５は、暗号鍵１３３を用いて符号化データ１１７の暗号解読をして、復元データ１３５を生成する。

【００４３】

２種類の中間データ１１５Ａ及び１１５Ｂに対応した２種類の復元データ１３５Ａ及び１３５Ｂの構成を図２に示す。図２から明らかなように、復元データ１３５Ａ及び１３５Ｂは、それぞれ、中間データ１１５Ａ及び１１５Ｂと同一である。すなわち、復元データ１３５は暗号解読されているが、電子透かしが挿入されたままである。電子透かしは視覚上画質に影響を与えないので、電子透かしが挿入されたままであっても問題はない。

【００４４】

次に、電子透かし検出部１２１の詳細について図５を参照して説明する。

【００４５】

図５を参照すると、電子透かし検出部１２１は、離散コサイン変換器１２１－１、電子透かしデータ候補保持部１２１－２及び検出部１２１－３を備える。

【００４６】

離散コサイン変換器１２１－１は、空間領域の電子透かし挿入済画像データＸ（ｊ）に対して離散コサイン変換を行い、電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）を出力する。

【００４７】

電子透かしデータ候補保持部１２１－２は、複数の電子透かしデータ候補を保持する。

【００４８】

検出部１２１－３は、電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）を基に、電子透かしデータＷ（ｉ）を、

Ｗ（ｉ）＝Ｆ（ｉ）／ａｖｇ（Ｆ（ｉ））

の計算式により計算し、１フレーム分のＷ（ｉ）の総和ＷＦ（ｉ）を各ｉについて計算する。

【００４９】

次に、検出部１２１－３は、電子透かし候補ｗ（ｉ）とＷＦ（ｉ）との統計的類似度Ｃをベクトルの内積を利用して、

Ｃ＝ＷＦ×ｗ／（ＷＦＤ×ｗＤ）

の計算式により計算する。ここで、

ＷＦ＝（ＷＦ（１）、ＷＦ（２）、・・・、ＷＦ（ｎ））、

ｗ＝（ｗ（１）、ｗ（２）、・・・、ｗ（ｎ））、

ＷＦＤ＝ベクトルＷＦの絶対値、

ｗＤ＝ベクトルｗの絶対値

である。統計的類似度Ｃがある特定の値以上である場合には、該当電子透かしデータが埋め込まれていると判定する。

【００５０】

電子透かしを検出するために、１フレームのうちの電子透かしが挿入されていて暗号化されていないブロックのみを必要とする場合には、各フレーム毎に暗号鍵を変えることができる。これに対し、電子透かしを検出するために、複数フレームのうちの電子透かしが挿入されていて暗号化されていないブロックを必要とする場合には、複数フレーム毎に暗号鍵を変えることができる。

【００５１】

［実施形態２］

次に、実施形態２の符号化装置及び復号化装置の説明をする前に、実施形態２の符号化装置及び復号化装置が扱うＭＰＥＧデータのうちの映像データについて説明する。

【００５２】

ＭＰＥＧデータのうちの映像データは、動き補償双方向フレーム間符号化が行われたものである。映像データは、ＧＯＰ(Group Of Picture)といわれるグループに分割される。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0813
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ここで、ピクチャとは、フレームのことである。図６に示すように、ＧＯＰは、複数のピクチャより構成される。ピクチャは、Ｉピクチャ、Ｂピクチャ及びＰピクチャに分類される。Ｉピクチャは、フレーム内符号化されるピクチャであり、１つのＧＯＰに１つある。Ｐピクチャは、前方向予測符号化が行われるピクチャである。Ｂピクチャは双方向予測符号化が行われるピクチャである。図６のピクチャ間を結ぶ矢印の先にあるピクチャはその矢印の元にあるピクチャを基準にして予測符号化される。例えば、Ｐ６ピクチャは、Ｉ３ピクチャを基準にして予測符号化され、Ｂ１ピクチャは、Ｉ３ピクチャ及びＰ６ピクチャを基準にして予測符号化され、Ｂ２ピクチャはＩ３ピクチャ及びＰ６ピクチャを基準にして予測符号化され、Ｐ９ピクチャは、Ｐ６ピクチャを基準にして予測符号化される。従って、復号化装置で、例えば、Ｐ６ピクチャを復元するためには、予めＩ３ピクチャが復元されていなければならず、Ｂ１ピクチャを復元するためには、予めＩ３ピクチャ及びＰ６ピクチャが復元されていなければならず、Ｂ２ピクチャを復元するためには、予めＩ３ピクチャ及びＰ６ピクチャが復元されていなければならず、Ｐ９ピクチャを復元するためには、予めＰ６ピクチャが復元されていなければならない。従って、復号化装置において、あるＧＯＰに１つしか存在しないＩピクチャが復元できなければ、そのＧＯＰの全てのピクチャを復元することができない。また、図６の例では、Ｉ３ピクチャが復元できたとしても、ＧＯＰ内の最初のＰピクチャであるＰ６ピクチャが復元できなければ、Ｉ３ピクチャを除き、これらが属するＧＯＰに属する全てのピクチャを復元することができない。

【００５３】

次に、実施形態２による符号化装置について説明する。図７は、実施形態２による符号化装置の構成を示すブロック図である。

【００５４】

図７を参照すると、実施形態２による符号化装置は、ＭＰＥＧエンコーダ２０１、電子透かし生成部２０３、電子透かし挿入部２０５及び暗号化部２０７を備える。

【００５５】

ＭＰＥＧエンコーダ２０１は、原画像データ２１３及び原音声データ２１４を入力し、ＭＰＥＧ規格に従ってこれらを符号化し、ＭＰＥＧデータである第１の中間データ２１５を生成する。図８に第１の中間データ２１５の各ピクチャを示す。

【００５６】

電子透かし生成部２０３は、暗号鍵２１１の全部又は一部を含んだ電子透かし２１２を生成する。暗号鍵２１１は、原画像データ２１３及び原音声データ２１４のコンテンツ毎に異なったものであることが好ましい。こうすることにより、あるコンテンツに対応した暗号鍵がハッキングされても、他のコンテンツを暗号解読することができない。また、暗号鍵２１１は、復号側でも用いられることを考慮し、共通鍵である。また、電子透かしを各ＧＯＰから検出することができるならば、暗号鍵１１１をＧＯＰ毎に変化させてもよい。こうすることにより、暗号鍵のハッキングに多大な時間を要することとなり、コンテンツの秘匿性を高めることができる。

【００５７】

電子透かし挿入部２０５は、第１の中間データ２１５のうちのＤＣＴ係数の部分に電子透かしを挿入し、電子透かしが挿入された第２の中間データ２１７を生成する。電子透かし挿入部２０５の詳細については後述する。図８に第２の中間データ２１７の各ピクチャを示す。図８に示す第２の中間データ２１７において、数字が円で囲まれたピックチャに電子透かしが挿入されており、その他のピクチャには電子透かしは挿入されていない。すなわち、Ｉピクチャには電子透かしが挿入されないが、他の全てのピクチャには電子透かしが挿入される。

【００５８】

次に、電子透かし挿入部２０５の詳細について図９を参照して説明する。

【００５９】

図９を参照すると、電子透かし挿入部２０５は、可変長復号化部２０５－１、電子透かし保持部２０５－２、挿入部２０５－３、調整部２０５－４、可変長符号化部２０５－５

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0814
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

、遅延部２０５－６及びマルチプレクサ２０５－７を備える。

【００６０】

可変長復号化部２０５－１は、第１の中間データ２１５を可変長復号し、各画像ブロックのＤＣＴ係数ｆ（ｉ）を出力する。

【００６１】

電子透かしデータ保持部２０５－２は、電子透かし２１２

ｗ（１）、ｗ（２）、・・・、ｗ（ｎ）

を保持する。なお、電子透かしデータは平均０、分散１の正規分布に従う。

【００６２】

挿入部２０５－３は、以下のことを行う。すなわち、ＤＣＴ係数ｆ（ｉ）と電子透かしデータｗ（ｉ）を基に、電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）を、

Ｆ（ｉ）＝ｆ（ｉ）＋α×ａｖｇ（ｆ（ｉ））×ｗ（ｉ）

の計算式により各ｉについて計算する。ここで、αはスケーリング要素であり、ａｖｇ（ｆ（ｉ））はｆ（ｉ）の近傍３点の絶対値の平均を取った部分平均である。そして、Ｆ（ｉ）を出力する。

【００６３】

調整部２０５－４は、調整後の各画像ブロックのＤＣＴ係数を可変長符号化したときの各画像ブロック当たりの総符号長が、電子透かしを挿入する前の各画像ブロックのＤＣＴ係数を可変長符号化したときの各画像ブロック当たりの総符号長が等しくなるように、各画像ブロックのＤＣＴ係数のレベルを調整する。

【００６４】

可変長符号化部２０５－５は、調整部２０５－４が出力するＤＣＴ係数を可変長符号化する。

【００６５】

遅延部２０５－６は、可変長復号化部２０５－１、挿入部２０５－３、調整部２０５－４及び可変長符号化部２０５－５でのデータの遅延時間と同一の時間だけ第１の中間データ２１５を遅延させる。

【００６６】

マルチプレクサ２０５－７は、可変長符号化部２０５－５が出力する可変長符号と、遅延部２０５－６が出力する第１の中間データのうちのＤＣＴ係数の可変長符号を除いた部分とをマルチプレクスする。つまり、マルチプレクサ２０５－７は、遅延部２０５－６が出力する第１の中間データのうちＤＣＴ係数に関連する符号だけ可変長符号化部２０５－５が出力する符号にすげ替える。

【００６７】

このようにして、電子透かし挿入部２０５は、ＭＰＥＧデータのうちのＤＣＴ係数の可変長符号の部分だけに変更を加える。電子透かし挿入部２０５の詳細については、特開２００１－００７７０５に記載されている。

【００６８】

なお、電子透かし挿入後の符号量が電子透かし挿入前の符号量と等しくなくても良いのならば、調整部２０５－４は不要であり、可変長符号化部２０５－５とマルチプレクサ２０５－７の間に可変遅延部を設け、遅延部２０５－６を可変とし、これらの遅延部からデータを出力用クロックで読み出す。

【００６９】

図７に戻り、暗号化部２０７は、第２の中間データ２１７を暗号鍵２１１で暗号化し、符号化データ２１９を生成する。図８に符号化データ２１９の各ピクチャを示す。図８に示す符号化データ２１９において、数字にバツを付しているピクチャが暗号化され、その他のピクチャは暗号化されない。すなわち、Ｉピクチャのみが暗号化される。従って、Ｉピクチャには、電子透かしが挿入されないが、Ｉピクチャは暗号化される。他方、Ｉピクチャ以外のピクチャには、電子透かしが挿入されるが、Ｉピクチャ以外のピクチャは暗号化されない。

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0815
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

【００７０】
従って、Ｉピクチャの暗号解読ができない限り、ＧＯＰの全てのピクチャを復元することができない。Ｉピクチャを暗号解読するための暗号鍵は、電子透かしの形態でＩピクチャ以外のピクチャに挿入されている。従って、復号化装置において、Ｉピクチャ以外のピクチャから電子透かしを検出し、検出された電子透かしから暗号鍵を抽出しない限り、Ｉピクチャの暗号解読ができない。従って、Ｉピクチャ以外のピクチャから電子透かしを検出しない限り、ＧＯＰの全てのピクチャを復元することができない。
【００７１】
一方、Ｉピクチャ以外のピクチャから電子透かしを検出すれば、Ｉピクチャの暗号解読をすることができるので、Ｉピクチャ以外のピクチャから電子透かしを検出すれば、ＧＯＰの全てのピクチャを復元することができる。すなわち、あるＧＯＰの全てのピクチャを復元するためには、そのＧＯＰのＩピクチャ以外のピクチャに挿入されている電子透かしが検出できれば良く、他のＧＯＰのピクチャに挿入されている電子透かしを検出する必要はない。元々、ＭＰＥＧデータの性質上、ランダムアクセスはフレーム単位で行うことができず、ＧＯＰ単位で行うこととなるが、本発明によりこれ以上の制限を加えることがなくなる。特許文献２の発明の場合には、あるＧＯＰを再生するために他のＧＯＰのデータが必要とされる。従って、特許文献２の発明は、ランダムアクセスへの適合性を有さないが、本発明は、ランダムアクセスへの適合性を有する。
【００７２】
図１０（ａ）～（ｃ）に符号化データ２１９の他のパターンを示す。図１０（ａ）に示すパターンでは、Ｉピクチャ及びＧＯＰ内の最初のピクチャが暗号化され、ＧＯＰの他のピクチャに暗号鍵を含む電子透かしが挿入される。図１０（ｂ）に示すパターンでは、Ｉピクチャ及びＧＯＰ内の全てのＰピクチャが暗号化され、ＧＯＰ内の全てのＢピクチャに暗号鍵を含む電子透かしが挿入される。図１０（ｃ）に示すパターンでは、Ｉピクチャ、ＧＯＰ内の全てのＰピクチャ及びＧＯＰの一部のＢピクチャが暗号化され、ＧＯＰ内の残りのＢピクチャに暗号鍵を含む電子透かしが挿入される。
【００７３】
図１１（ａ）～（ｄ）に符号化データ２１９の更に他のパターンを示す。図１１（ａ）～（ｄ）に示すパターンにおいては、全てのピクチャに電子透かしが挿入される。そして、図１１（ａ）に示すパターンでは、Ｉピクチャのみが暗号化され、図１１（ｂ）に示すパターンでは、Ｉピクチャ及びＧＯＰ内の最初のピクチャが暗号化され、図１１（ｃ）に示すパターンでは、Ｉピクチャ及びＧＯＰ内の全てのＰピクチャが暗号化され、図１１（ｄ）に示すパターンでは、Ｉピクチャ、ＧＯＰ内の全てのＰピクチャ及びＧＯＰ内の一部のＢピクチャが暗号化される。実施形態１と同様に、暗号化されるピクチャに電子透かしが挿入されていても、暗号化及び暗号解読に支障はない。また、ピクチャの区別無く電子透かしを挿入すれば、電子透かしを挿入するかしないかを決定するための部分が不要であるため、電子透かし挿入部２０５を簡素化することができる。従って、電子透かし挿入部２０５を簡素化する目的のために、全てのピクチャに電子透かしを挿入するようにしても良い。
【００７４】
また、図１１（ｂ）、（ｃ）又は（ｄ）のパターンを変形させて、電子透かしが挿入されないピクチャをピクチャＩ３のみとしても良いし、電子透かしが挿入されないピクチャをピクチャＩ３及びＰ６のみとしてもよい。
【００７５】
図８に示す符号化データ、図１０（ａ）に示す符号化データ、図１０（ｂ）に示す符号化データ、図１０（ｃ）に示す符号化データの順に、暗号化されるピクチャの数が増加し、暗号鍵を含む電子透かしが挿入されるピクチャの数が減少する。
【００７６】
Ｐピクチャ及びＢピクチャには、画像の内容によっては、フレーム内符号化されるイントラマクロブロックが含まれる場合がある。イントラマクロブロックは、Ｉピクチャが復

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0816
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

元されていなくても、復元することができる。従って、復元できるイントラマクロブロックを減らして、暗号解読できない場合の画像の認識性を低くするためには、暗号化されるピクチャの数が多い方が好ましい。一方、暗号化されるピクチャの数を多くすると、暗号鍵を含んだ電子透かしであって検出できるものを挿入できるピクチャの数が少なくなり、電子透かしの検出の精度が下がる。従って、暗号化されるピクチャの数を、暗号解読できない場合の画像の認識性と電子透かしの検出の精度を考慮して定める。

【００７７】

次に、実施形態２による復号化装置について説明する。図１２は、実施形態２による復号化装置の構成を示すブロック図である。

【００７８】

図１２を参照すると、実施形態２による復号化装置は、電子透かし検出部２２１、遅延部２２２、暗号鍵抽出部２２３、暗号解読部２２５及びＭＰＥＧデコーダ２２７を備える。

【００７９】

電子透かし検出部２２１は、符号化データ２１９から電子透かし２３１を検出する。電子透かし検出部２２１の詳細については後述する。暗号鍵抽出部２２３は、電子透かし２３１から暗号鍵２３３を抽出する。実施形態１の場合と同様に暗号鍵２３３は暗号鍵２１１の全部又は一部であり、暗号鍵２３３に暗号鍵２１１の残りの部分を合わせて暗号解読部２２５に供給する必要があるが、説明の簡単のために暗号鍵２３３は暗号鍵２１１と同一であるとする。遅延部２２２は、暗号鍵抽出部２２３が暗号鍵２３３を出力するまでの間、符号化データ２１９が暗号解読部２２５に入力されるタイミングを遅らせるために、符号化データ２１９を遅延させる。暗号解読部２２５は、暗号鍵２３３を用いて遅延された符号化データ２１９の暗号解読をして、第３の中間データ２３５を生成する。ＭＰＥＧデコーダ２２７は、第３の中間データ２３５を復号して、復元画像データ２３７及び復元音声データ２３９を生成する。

【００８０】

第３の中間データ２３５は、図８に示すように、暗号解読されているが、電子透かしは挿入されたままである。同様に、復元画像データ２３７も、図８に示すように、暗号解読されているが、電子透かしは挿入されたままである。

【００８１】

次に、電子透かし検出部２２１の詳細について図１３を参照して説明する。

【００８２】

図１３を参照すると、電子透かし検出部２２１は、可変長復号部２２１－１、電子透かしデータ候補保持部２２１－２及び検出部２２１－３を備える。

【００８３】

可変長復号部２２１－１は、符号化データ２１９を可変長復号し、暗号化されていないピクチャの各ブロックの電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）を出力する。なお、本実施形態では、ＭＰＥＧデータ中の各種のビデオスタートコードを暗号化しないので、ビデオスタートコードの一種であるピクチャースタートコードにより各ピクチャーの先頭を検出することができるので、途中に暗号化されているピクチャが混じっていても各ピクチャを認識することができる。

【００８４】

電子透かしデータ候補保持部２２１－２は、複数の電子透かしデータ候補を保持する。

【００８５】

検出部２２１－３は、電子透かし挿入後のＤＣＴ係数Ｆ（ｉ）を基に、電子透かしデータＷ（ｉ）を、

Ｗ（ｉ）＝Ｆ（ｉ）／ａｖｇ（Ｆ（ｉ））

の計算式により計算し、１フレーム分のＷ（ｉ）の総和ＷＦ（ｉ）を各ｉについて計算する。

【００８６】

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0817
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

次に、検出部２２１－３は、電子透かし候補ｗ（ｉ）とＷＦ（ｉ）との統計的類似度Ｃをベクトルの内積を利用して、

Ｃ＝ＷＦ×ｗ／（ＷＦＤ×ｗＤ）

の計算式により計算する。ここで、

ＷＦ＝（ＷＦ（１）、ＷＦ（２）、・・・、ＷＦ（ｎ））、

ｗ＝（ｗ（１）、ｗ（２）、・・・、ｗ（ｎ））、

ＷＦＤ＝ベクトルＷＦの絶対値、

ｗＤ＝ベクトルｗの絶対値

である。統計的類似度Ｃがある特定の値以上である場合には、該当電子透かしデータが埋め込まれていると判定する。

【００８７】

［実施形態３］

実施形態２においては、符号化装置において、ＭＰＥＧエンコーダ２０１の後段に電子透かし挿入部２０５を配置していた。これに対し、実施形態７においては、図１４に示すように、符号化装置において、ＭＰＥＧエンコーダ２０１の前段に電子透かし挿入部２０９を配置する。電子透かし挿入部２０９の構成は図３に示すようなものである。図１４において、電子透かし挿入部２０９が出力する第１の中間データ２５１は、全部又は一部のピクチャに電子透かしが挿入された画像データである。ＭＰＥＧエンコーダ２０１が出力する第２の中間データ２５３は、全部又は一部のピクチャに電子透かしが挿入されたＭＰＥＧデータである。実施形態３の符号化装置が出力する符号化データ２１９は、実施形態２の符号化装置が出力する符号化データ２１９と同一である。

【００８８】

なお、上記の実施形態の符号化装置の各部及び復号化装置の各部は、ハードウェアによって実現することもできるし、コンピュータをこれらの部分として機能させるためのプログラムをコンピュータが読み込んで実行することによっても実現することができるし、ハードウェアとソフトウェアのハイブリッド構成で実現することもできる。

【００８９】

また、上記の実施形態では、共通鍵により暗号化及び暗号解読をするとした。この代わりに、公開鍵によりデータの暗号化を行い、データに秘密鍵を電子透かしの形態で挿入し、復号側で、電子透かしを検出し、秘密鍵を抽出し、抽出された秘密鍵でデータの暗号解読を行うようにしてもよい。

【産業上の利用可能性】

【００９０】

本発明は、ＤＶＤ等に記録された映画等のコンテンツの秘匿性を高め、且つ、ランダムアクセスに対応するために用いることができる。

【図面の簡単な説明】

【００９１】

【図１】本発明の実施形態１による符号化装置の構成を示すブロック図である。

【図２】本発明の実施形態１による符号化装置の各部におけるデータの構成を示す図である。

【図３】本発明の実施形態１による符号化装置で用いられる電子透かし挿入部の構成を示すブロック図である。

【図４】本発明の実施形態１による復号化装置の構成を示すブロック図である。

【図５】本発明の実施形態１による復号化装置で用いられる電子透かし検出部の構成を示すブロック図である。

【図６】本発明の実施形態２で扱うＭＰＥＧデータの構成を示す概念図である。

【図７】本発明の実施形態２による符号化装置の構成を示すブロック図である。

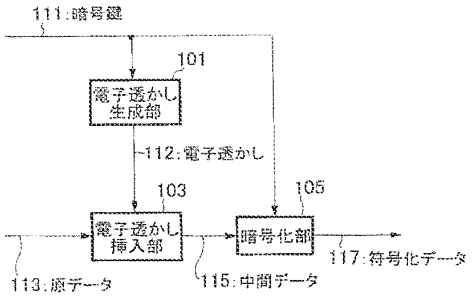
【図８】本発明の実施形態２による符号化装置及び復号化装置の各部におけるデータの構成を示す図である。

【図９】本発明の実施形態２による符号化装置で用いる電子透かし挿入部の構成を示すブ

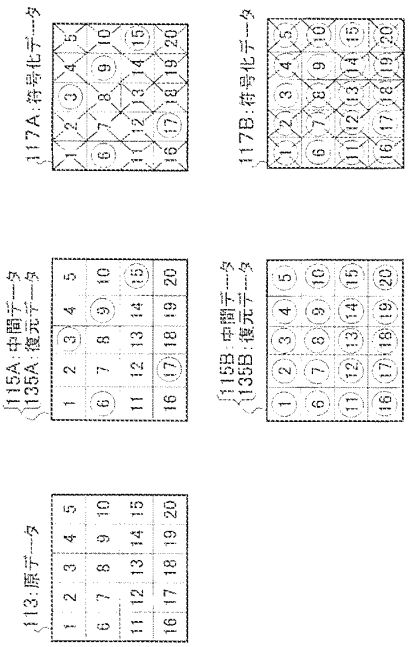
Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0818
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

ロック図である。
【図１０】本発明の実施形態２による符号化装置が出力する符号化データの各バリエーションの構成を示す図である。
【図１１】本発明の実施形態２による符号化装置が出力する符号化データの各バリエーションの構成を示す図である。
【図１２】本発明の実施形態２による復号化装置の構成を示すブロック図である。
【図１３】本発明の実施形態２による復号化装置で用いる電子透かし検出部の構成を示すブロック図である。
【図１４】本発明の実施形態３による符号化装置の構成を示すブロック図である。
【符号の説明】
【００９２】
１０１　電子透かし生成部
１０３　電子透かし挿入部
１０５　暗号化部
１２１　電子透かし検出部
１２３　暗号鍵抽出部
１２５　暗号解読部
２０１　ＭＰＥＧエンコーダ
２０３　電子透かし生成部
２０５　電子透かし挿入部
２０７　暗号化部
２２１　電子透かし検出部
２２２　遅延部
２２３　暗号鍵抽出部
２２５　暗号解読部
２２７　ＭＰＥＧデコーダ

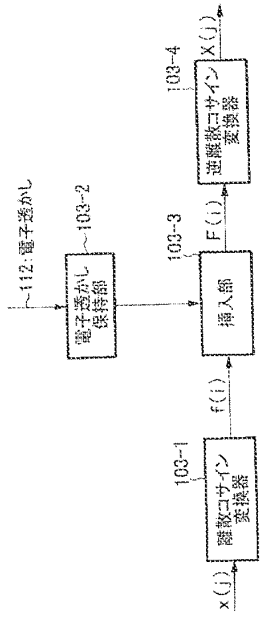
【図１】


111:暗号鍵
101
電子透かし生成部
112:電子透かし
103
105
電子透かし挿入部
暗号化部
117:符号化データ
113:原データ
115:中間データ

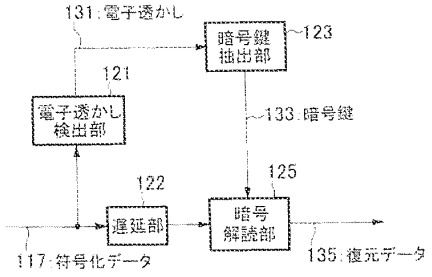
【図２】


117A:符号化データ
117B:符号化データ
115A:中間データ
135A:復元データ
115B:中間データ
135B:復元データ
113:原データ

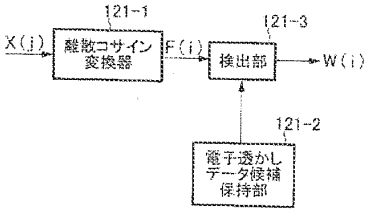
【図３】


112:電子透かし
103-2
電子透かし保持部
103-3
挿入部
F(i)
103-4
逆離散コサイン変換器
X(j)
f(i)
103-1
離散コサイン変換器
x(j)

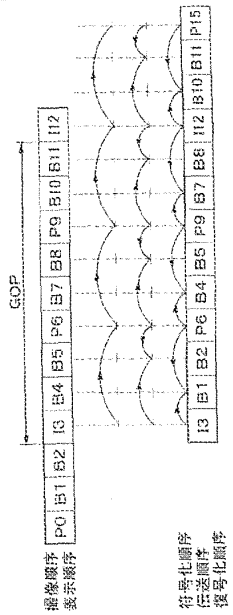
【図４】


131:電子透かし
暗号鍵抽出部
123
121
電子透かし検出部
133:暗号鍵
125
122
遅延部
暗号解読部
117:符号化データ
135:復元データ

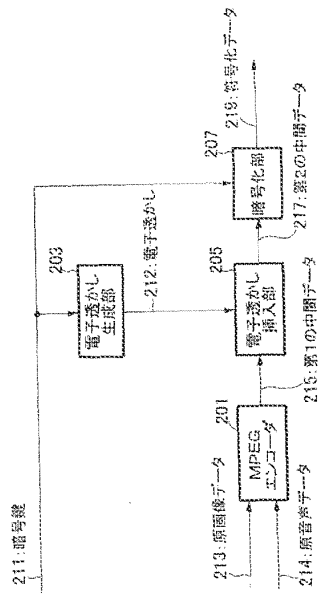
【図５】


121-1
X(j)
離散コサイン変換器
F(i)
121-3
検出部
W(i)
121-2
電子透かしデータ候補保持部

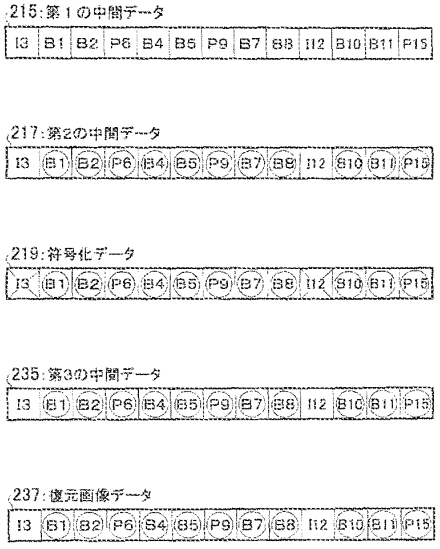
【図６】


GOP
I3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15
I3 B1 B2 P0 B4 B5 P6 B7 B8 I12 B10 B11 P9
撮像順序
表示順序
符号化順序
伝送順序
復号化順序

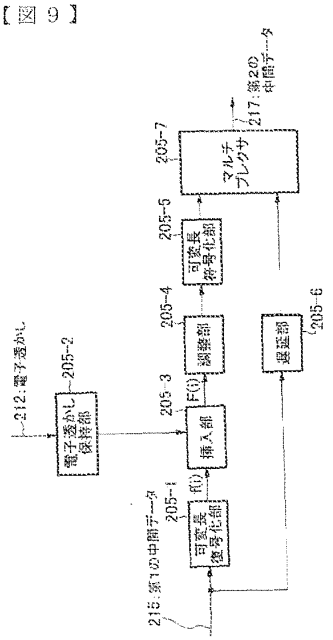
【図７】


電子透かし生成部
203
212:電子透かし
電子透かし挿入部
205
207
符号化部
219:符号化データ
217:第2の中間データ
215:第1の中間データ
MPEGエンコーダ
201
211:暗号鍵
213:原画像データ
214:原音声データ

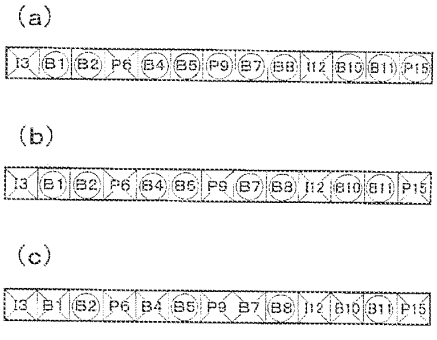
【図８】


215:第1の中間データ
I3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15
217:第2の中間データ
I3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15
219:符号化データ
I3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15
235:第3の中間データ
I3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15
237:復元画像データ
I3 B1 B2 P6 B4 B5 P9 B7 B8 I12 B10 B11 P15

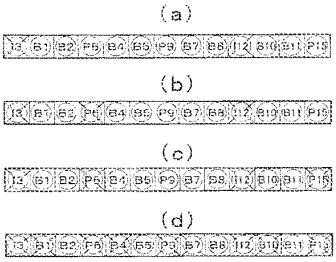
【図９】


212:電子透かし
電子透かし保持部
205-2
205-3
F(i)
挿入部
205-4
減算部
205-5
可変長符号化部
205-7
マルチプレクサ
217:第2の中間データ
205-6
減算部
205-1
F(i)
可変長復号化部
216:第1の中間データ

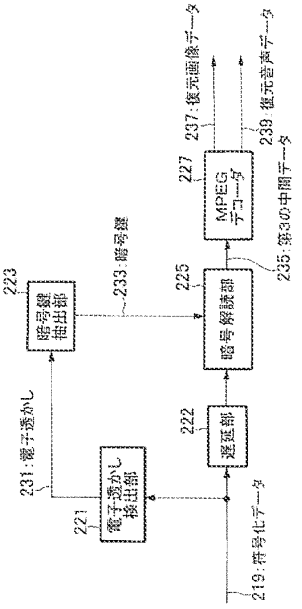
【図１０】



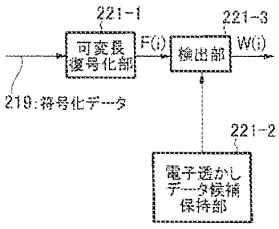
【図１１】



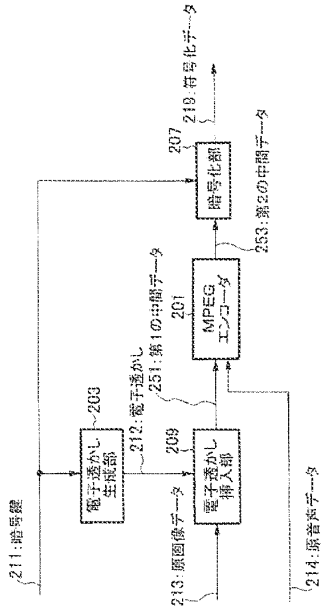
【図１２】



【図１３】



【図１４】



フロントページの続き

(51)Int.Cl.[7]	F I		テーマコード（参考）
H04N 7/24	H04N 7/13	Z	
	H04N 7/167	Z	

Fターム(参考) 5B057 CA12 CA18 CB12 CB18 CE08 CG07
5C059 KK43 MA00 MA14 MA23 PP05 PP06 PP07 RB01 RC35 SS13
UA02 UA05
5C064 CA14 CA18 CB01 CC01
5C076 AA14 BA06 BA09
5J104 AA14 EA17

Electronic Patent Application Fee Transmittal

Application Number:	13340623
Filing Date:	29-Dec-2011
Title of Invention:	Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information
First Named Inventor/Applicant Name:	Michael George Kiefer
Filer:	Jason Daniel Wolfe/Katrina Lee
Attorney Docket Number:	D1:02359A

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Request for Continued Examination	1801	1	1200	1200
		Total in USD ($)		**1200**

Electronic Acknowledgement Receipt

EFS ID:	20080035
Application Number:	13340623
International Application Number:	
Confirmation Number:	1094
Title of Invention:	Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information
First Named Inventor/Applicant Name:	Michael George Kiefer
Customer Number:	71897
Filer:	Jason Daniel Wolfe/Katrina Lee
Filer Authorized By:	Jason Daniel Wolfe
Attorney Docket Number:	D1:02359A
Receipt Date:	08-SEP-2014
Filing Date:	29-DEC-2011
Time Stamp:	20:44:59
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$1200
RAM confirmation Number	7766
Deposit Account	504407
Authorized User	BAILEY, DAVID

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Continued Examination (RCE)	02359arce.pdf	76782 7bdaaae172c371ccba8b04c749efe73eaedfcc1c	no	2
Warnings:					
This is not a USPTO supplied RCE SB30 form.					
Information:					
2	Transmittal Letter	02359aidsxtml.pdf	91384 96f819f3ddc1796f3e9905a0f1638b10e1eb76fc	no	2
Warnings:					
Information:					
3	Information Disclosure Statement (IDS) Form (SB08)	02359aids1449.pdf	175244 4366edeeaf159fe2b4bb3fa9b216e804a57147f9	no	10
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
4	Non Patent Literature	informationtechnology.pdf	10449205 b67ea7bcfec5bb8d444ad49e2ec8dd18342de3ed	no	724
Warnings:					
Information:					
5	Non Patent Literature	cloakwareprotectingdigitalcontent.pdf	10106070 89e25abea8af5303ccf4a29191e2ccdc0f541954	no	10
Warnings:					
Information:					
6	Other Reference-Patent/App/Search documents	esr11855103.pdf	8679470 f1975d17d64656591871b242cb7b2946204feaf0	no	9
Warnings:					
Information:					
7	Other Reference-Patent/App/Search documents	esr118552371.pdf	9574333 2a6568dd3fa8bc5a4dc19b0f699e1095c878f8a2	no	9
Warnings:					
Information:					

8	Non Patent Literature	fedweektoolspeedsinfotovehicl esfedcom.pdf	667614 be5e70ef770361d448312f83398eaa4f9125 6db9	no	5
Warnings:					
Information:					
9	Non Patent Literature	httplivestreamingingnetwork. pdf	544658 9dda7c96aa17a76c13ed8a50b5c6b959c3a bbd30	no	38
Warnings:					
Information:					
10	Other Reference-Patent/App/Search documents	intlprelimrprt68276.pdf	17679480 7ac8b9239df6703f4ac153d3fad62d3f8d28 6a85	no	23
Warnings:					
Information:					
11	Other Reference-Patent/App/Search documents	intlsearchreport56733.pdf	1028691 341178042667423d31230aaed229613dae3 3e8b2	no	9
Warnings:					
Information:					
12	Other Reference-Patent/App/Search documents	isrwo066927.pdf	3711899 0a0889e6dd01724add5d9dfbd3067f9967c 3df2d	no	14
Warnings:					
Information:					
13	Other Reference-Patent/App/Search documents	isrwo067167.pdf	1160661 2bc51a5303ba8211456e20e93bbbc2ab6af 30a77	no	11
Warnings:					
Information:					
14	Other Reference-Patent/App/Search documents	intlsearchrprt25845.pdf	2094044 fe16b479e75cd141ee0abe891f2ee1963b9 57a87	no	3
Warnings:					
Information:					
15	Other Reference-Patent/App/Search documents	intlsearchrprt63950.pdf	2691694 f0e2b4bad221bdedcd9f267f27c930e51d6c 461e	no	3
Warnings:					
Information:					
16	Non Patent Literature	itsfleetsystemoptsformobile. pdf	110053 228a1879da5e55dba9cb3f0145255f9d1ec 5ed00	no	1
Warnings:					
Information:					

17	Non Patent Literature	microsoftmediaplatformv11.pdf	106175 26ed297be732bdb88460ddd87580cebc2b57536f	no	2
Warnings:					
Information:					
18	Non Patent Literature	microsoftmediaplatformv25.pdf	80873 9d4a4aae9c828be8e0be749b0917bcfefe4034ec	no	2
Warnings:					
Information:					
19	Non Patent Literature	microsoftsmoothstreamingclient.pdf	386881 ff0e42bbf436798af29d6b2cfb1e327c3d17cb8d	no	4
Warnings:					
Information:					
20	Other Reference-Patent/App/Search documents	writtenopinionpct0763950.pdf	1042006 fa05ffe3979e80a367185507a8ddb9476dee92d3	no	5
Warnings:					
Information:					
21	Other Reference-Patent/App/Search documents	writtenopinionpct05025845.pdf	383433 43c343448aeebe01a4266f885e9cd543e05e7756	no	5
Warnings:					
Information:					
22	Non Patent Literature	adaptivestreamingcompa.pdf	122315 04ce194308d7da6468bc61453fe682a145f29cbc	no	5
Warnings:					
Information:					
23	Non Patent Literature	bestpracticesformultidevi.pdf	2988083 9ce3bd1d5fbba7025fbef30892b447f71fc87337	no	13
Warnings:					
Information:					
24	Non Patent Literature	informationweekfrontend3pgs.pdf	242661 bf1de111dbf6d565c3e586f8e5a486a190098126	no	3
Warnings:					
Information:					
25	Non Patent Literature	netflixturnssubtitles.pdf	322961 8ed38f2218c15728ff6be521f205c6f1aa347be9	no	3
Warnings:					
Information:					

26	Other Reference-Patent/App/Search documents	epsuppsearchreport10834935.pdf	372951 00182ab596879aa50db14b7ad91cb3f9ce88bdff	no	9
Warnings:					
Information:					
27	Non Patent Literature	supportedmediaformats.pdf	87610 3dec8ba9b2e2f3001600554eb1facaaa2192808e3	no	3
Warnings:					
Information:					
28	Non Patent Literature	threadssmeconfigxml.pdf	220324 387a1b5490afcd1b231720becd7ca750327ddc95	no	3
Warnings:					
Information:					
29	Non Patent Literature	transcodingbestpractices.pdf	107602 096fc6ce3c88f951b334db9a94a956fa2f53419a	no	5
Warnings:					
Information:					
30	Non Patent Literature	usinghttplivestreaming.pdf	286791 bdd71ba47e9b05c650d330432a3cca36cf5114e5	no	10
Warnings:					
Information:					
31	Non Patent Literature	anexperimentalevaluation.pdf	251058 a3166ee8c470cdc8448ab40ff002708203ff728d	no	12
Warnings:					
Information:					
32	Foreign Reference	EP813167A2.pdf	1731589 073f6d99becf3ba0a4ce2e79efe78a0bf5461328	no	15
Warnings:					
Information:					
33	Foreign Reference	EP936812A1.pdf	3293769 08eaafaf7e51e03c90a6185237a61233313dca2f	no	31
Warnings:					
Information:					
34	Foreign Reference	WO0165762A2.pdf	19941894 2858920deb0e845263a2be57c6f4b61ae1a10b66	no	17
Warnings:					
Information:					

35	Foreign Reference	wo0237210.pdf	1846951 c393b363de4ff08afef0c04361f9b5b8dc92d227	no	45
Warnings:					
Information:					
36	Foreign Reference	wo02054196.pdf	1864599 68a2c2a68edb6438d9f0eece34a6c7b181478c4d	no	34
Warnings:					
Information:					
37	Foreign Reference	WO2004102571A1.pdf	24094361 d18c18b83e3d5a111c3575794f09738a986e7893	no	25
Warnings:					
Information:					
38	Foreign Reference	jp2006524007wenglishtranslationr.pdf	8184075 f98ea922a42343b3cbcf3d760811ecb040c7814d	no	56
Warnings:					
Information:					
39	Foreign Reference	WO2008010275A1.pdf	6761737 8d3cc507394a32167280bf522071d68b42bd95eb	no	56
Warnings:					
Information:					
40	Foreign Reference	WO2009065137A1.pdf	3385362 26bc47cf24bca9fde1cb24fa4e92b6e26b1076f2	no	35
Warnings:					
Information:					
41	Foreign Reference	wo2010060106A1.pdf	3120538 02496cc209ffd3b7ce1e08873066aae45d907661	no	67
Warnings:					
Information:					
42	Foreign Reference	WO2010122447A1.pdf	20063745 754a77a43a83afc3fbef6ea045bba8e576254764	no	22
Warnings:					
Information:					
43	Foreign Reference	wo2011068668.pdf	1237998 c12b3f7aaeb35e9ba87013705eb17b0533d05eac	no	34
Warnings:					
Information:					

44	Foreign Reference	wo2012094181A2.pdf	2723071 21371be78a99626b24d13bc01e435ce3ca2979b9	no	61
Warnings:					
Information:					
45	Foreign Reference	wo2012094189a1.pdf	2531347 b237734fa2d49349f5a58b4e1ea1f37a935eef46	no	67
Warnings:					
Information:					
46	Foreign Reference	wo2012094171a1.pdf	8042471 31a07670d5fa477a7e38333a6dafe0b1d7043301	no	57
Warnings:					
Information:					
47	Foreign Reference	jp2005080204wtranslationabstract.pdf	23787301 66414b8cf0bc3c89faa0f625e7dbbc10c0b410cb	no	22
Warnings:					
Information:					
48	Fee Worksheet (SB06)	fee-info.pdf	30621 5096492402858034a61d56b71efd378568c36eec	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	208484435	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	13/340,623
	Filing Date	Dec 29, 2011
	First Named Inventor	Michael George Kiefer
	Art Unit	2434
	Examiner Name	Abiy Getachew
	Attorney Docket Number	D1-02359A

NON-PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T 5
		ANONYMOUS, "Method for the encoding of a compressed video sequence derived from the same video sequence compressed at a different bit rate without loss of data", ip.com, ip.com number: IPCOM000008165D, May 22, 2012, pp. 1-9.	
		AUTHOR UNKNOWN, "Entropy and Source Coding (Compression)", TCOM 570, 1999-9, pp. 1-22,	
		AUTHOR UNKNOWN, "MPEG-4 Video Encoder: Based on International Standard ISO/IEC 14496-2", Patni Computer Systems, Ltd., Publication date unknown, 15 pgs.	
		AUTHOR UNKNOWN, "Tunneling QuickTime RTSP and RTP over HTTP", Published by Apple Computer, Inc.: 1999 (month unknown), 6 pages.	
		BLASIAK, PH.D., DAREK, Video Transrating and Transcoding: Overview of Video Transrating and Transcoding Technologies", Ingenient Technologies, TI Developer Conference, August 6-8, 2002, 22 pgs.	
		DEUTSCHER, "IIS Transform Manager Beta - Using the MP4 to Smooth Task", Retrieved from: https://web.archive.org/web/20130328111303/http://blog.johndeutscher.com/category/smooth-streaming, Blog post of April 17, 2010, 14 pgs.	
		GANNES, "The Lowdown on Apple's HTTP Adaptive Bitrate Streaming", GigaOM, June 10, 2009, 12 pgs.	
		GHOSH, "Enhancing Silverlight Video Experiences with Contextual Data", Retrieved from: http://msdn.microsoft.com/en-us/magazine/ee336025.aspx, 15 pgs.	
		INLET TECHNOLOGIES, "Adaptive Delivery to iDevices", 2 pages.	
		INLET TECHNOLOGIES, "Adaptive delivery to iPhone 3.0", 2 pgs.	

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	13/340,623
	Filing Date	Dec 29, 2011
	First Named Inventor	Michael George Kiefer
	Art Unit	2434
	Examiner Name	Abiy Getachew
	Attorney Docket Number	D1-02359A

		INLET TECHNOLOGIES, "HTTP versus RTMP", 3 pages.	
		INLET TECHNOLOGIES, "The World's First Live Smooth Streaming Event: The French Open", 2 pages.	
		KIM, KYUHEON, "MPEG-2 ES/PES/TS/PSI", Kyung-Hee University, October 4, 2010, 66 pages.	
		KURZKE ET AL., "Get Your Content Onto Google TV", Google, Retrieved from: http://commondatastorage.googleapis.com/io2012/presentations/live%20to%20website/1300.pdf, 58 pgs.	
		LANG, "Expression Encoder, Best Practices for live smooth streaming broadcasting", Microsoft Corporation, 20 pgs.	
		LEVKOV, "Mobile Encoding Guidelines for Android Powered Devices", Adobe Systems Inc., Addendum B, source and date unknown, 42 pgs.	
		MSDN, "Adaptive streaming, Expression Studio 2.0", 2 pgs.	
		NELSON, "Smooth Streaming Deployment Guide", Microsoft Expression Encoder, August 2010, 66 pgs.	
		NELSON, MARK, "Arithmetic Coding + Statistical Modeling = Data Compression: Part 1 - Arithmetic Coding", Doctor Dobb's Journal, February 1991, printed from http://www.dogma.net/markn/articles/arith/part1.htm; printed July 2, 2003, 12 pgs.	
		Noé, ALEXANDER, "Matroska File Format (under construction !)", June 24, 2007, XP002617671, Retrieved from: http://web.archive.org/web/20070821155146/www.matroska.org/technical/specs/matroska.pdf, Retrieved on January 19, 2011, 1-51.	
		OZER, "The 2012 Encoding and Transcoding Buyers' Guide", Streamingmedia.com, Retrieved from: http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-2012-Encoding-and-Transcoding-Buyers-Guide-84210.aspx, 2012, 8 pgs.	

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	13/340,623
	Filing Date	Dec 29, 2011
	First Named Inventor	Michael George Kiefer
	Art Unit	2434
	Examiner Name	Abiy Getachew
	Attorney Docket Number	D1-02359A

		PANTOS, "HTTP Live Streaming, draft-pantos-http-live-streaming-10", IETF Tools, October 15, 2012, Retrieved from: http://tools.ietf.org/html/draft-pantos-http-live-streaming-10, 37 pgs.	
		PANTOS, R., "HTTP Live Streaming: draft-pantos-http-live-streaming-06", Published by the Internet Engineering Task Force (IETF), March 31, 2011, 24 pages.	
		PHAMDO, NAM, "Theory of Data Compression", printed from http://www.data-compression.com/theoroy.html on October 10, 2003, 12 pgs.	
		RGB NETWORKS, "Comparing Adaptive HTTP Streaming Technologies", November 2011, Retrieved from: http://btreport.net/wp-content/uploads/2012/02/RGB-Adaptive-HTTP-Streaming-Comparison-1211-01.pdf, 20 pgs.	
		SCHULZRINNE, H et al., "Real Time Streaming Protocol 2.0 (RTSP): draft-ietfmmusic-rfc2326bis-27", MMUSIC Working Group of the Internet Engineering Task Force (IETF), March 9, 2011, 296 pages.	
		SIGLIN, "HTTP Streaming: What You Need to Know", streamingmedia.com, 2010, 16 pages.	
		SIGLIN, "Unifying Global Video Strategies, MP4 File Fragmentation for Broadcast, Mobile and Web Delivery", Nov. 16, 2011, 16 pgs.	
		WU, FENG et al., "Next Generation Mobile Multimedia Communications: Media Codec and Media Transport Perspectives", In China Communications, October 2006, pp. 30-44.	
		ZAMBELLI, ALEX, "IIS Smooth Streaming Technical Overview", Microsoft Corporation, March 2009.	

PTO/SB/08a (01-10)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	13/340,623
	Filing Date	Dec 29, 2011
	First Named Inventor	Michael George Kiefer
	Art Unit	2434
	Examiner Name	Abiy Getachew
	Attorney Docket Number	D1-02359A

EXAMINER SIGNATURE			
Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. [2] Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). [3] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document.
[4] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [5] Applicant is to place a check mark here if English language translation is attached.

Electronic Acknowledgement Receipt

EFS ID:	20080158
Application Number:	13340623
International Application Number:	
Confirmation Number:	1094
Title of Invention:	Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information
First Named Inventor/Applicant Name:	Michael George Kiefer
Customer Number:	71897
Filer:	Jason Daniel Wolfe/Katrina Lee
Filer Authorized By:	Jason Daniel Wolfe
Attorney Docket Number:	D1:02359A
Receipt Date:	08-SEP-2014
Filing Date:	29-DEC-2011
Time Stamp:	21:09:04
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	02359aidsxtml.pdf	103132 58f37ee58aa7e544e0304bb9f7f990b943cb7341	no	2

Warnings:	
Information:	

2	Non Patent Literature	methodfortheencoding.pdf	1890587 2bb121487856353f2acf49b1dffea2565d491bbc	no	9
Warnings:					
Information:					
3	Non Patent Literature	entropyandsourcecoding.pdf	4839853 8495ed3b099d54721ff1bba6879dee6eafa3adaf	no	22
Warnings:					
Information:					
4	Non Patent Literature	mpeg4videoencoder.pdf	1561848 7ddc0005b213d55362f4af5ffe37c555824cce5e	no	15
Warnings:					
Information:					
5	Non Patent Literature	tunnelingquicktimertsp.pdf	144154 1444ef4b14772175f35c41ec9a58021e94287d0d	no	6
Warnings:					
Information:					
6	Non Patent Literature	videotransratingandtranscoding.pdf	7991343 0ebfcc8beb9a946424c41e1180c7c652f6b230b2	no	22
Warnings:					
Information:					
7	Non Patent Literature	iistransformmanager.pdf	1064216 b83a17b889436461eb1a8b3b5c1e91653a098534	no	14
Warnings:					
Information:					
8	Non Patent Literature	gannesthelowdownonapple.pdf	608734 412edc5c182dfea985c801ca8f743d36e7c26186	no	12
Warnings:					
Information:					
9	Non Patent Literature	ghoshenhancingsilverlightvideo.pdf	879482 c68dc12c3c08af0a5c58420b6e9c485491162937	no	15
Warnings:					
Information:					
10	Non Patent Literature	inletfrenchopenpdf.pdf	733061 092d685f2ba064e91804ff68b2d9627a47409154	no	2
Warnings:					
Information:					

11	Non Patent Literature	inlethttpcsrtmp.pdf	254318 ccd9074668c471629f8239c26d804c7fb0287e20	no	3
Warnings:					
Information:					
12	Non Patent Literature	inletidevicesoultion.pdf	280152 1f8aebe5512aff75438791e0007a18434bf50f08	no	2
Warnings:					
Information:					
13	Non Patent Literature	mpeg2espestspsi.pdf	5387515 fbf80a4fe1524132857d453fb2aef30564ee72b1	no	66
Warnings:					
Information:					
14	Non Patent Literature	googletv.pdf	5936174 72567184c7a628bc7a4e744f6a2353efd40de25c	no	58
Warnings:					
Information:					
15	Non Patent Literature	langexpressionencoder.pdf	1664576 8e4be64d6928827cd4a9110e6eaf1f7cdc900bf3	no	20
Warnings:					
Information:					
16	Non Patent Literature	levkovmobileencodingguidelinesforandroid.pdf	4807496 0fc2b9950886d5ded6e89ac4c5fe92404c535c3a	no	42
Warnings:					
Information:					
17	Non Patent Literature	msdnadaptivestreaming.pdf	40339 95260cb4996c906d735097c43ad904e4ae02adeb	no	2
Warnings:					
Information:					
18	Non Patent Literature	nelsonsmoothstreamingdep.pdf	19744963 afa585099baa5c27a3b853f2b23ad0ff77a17a6a	no	66
Warnings:					
Information:					
19	Non Patent Literature	arithmeticcoding.pdf	12034679 aeb00f11787fbf55373e06fb94406a1065343f5d	no	12
Warnings:					
Information:					

20	Non Patent Literature	noematroskafileformakunderconst.pdf	657072 b06dcf638b6d64e2c717efbd1f5b5b1af700640b	no	51
Warnings:					
Information:					
21	Non Patent Literature	ozerthe2012encoding.pdf	1154277 d66ebeff99ff5d490a06b9a3b1c0ac6dabe3ab54	no	8
Warnings:					
Information:					
22	Non Patent Literature	pantoshttplivestreamingdraft10.pdf	70810 ea69d853ab13a16a542a43709c307906e046bc54	no	37
Warnings:					
Information:					
23	Non Patent Literature	httplivestreamingdraft06.pdf	51488 5f2f07b05ea2bab0662dc5fb8f8be6e2a1d4ca40	no	24
Warnings:					
Information:					
24	Non Patent Literature	theoryofdataacompression.pdf	1671273 45dadad05efe79b965727f067e80daa36f9f0d7f	no	12
Warnings:					
Information:					
25	Non Patent Literature	rgbnetworkscomparingadaptive.pdf	1980675 e5d8a46580503a0914baff0cf2123afdd466a59c	no	20
Warnings:					
Information:					
26	Non Patent Literature	realtimestreamingprotocol2rtspdraft27.pdf	488413 680e07d7325809cc47205ff7480d004c00176e48	no	296
Warnings:					
Information:					
27	Non Patent Literature	siglinunifyingglobalvideo.pdf	290993 52de0de2621f85937a61e34b80e7e78ff85ee83d7	no	16
Warnings:					
Information:					
28	Non Patent Literature	siglinhttpstreamingwhatyou.pdf	1321101 9d597990f5d1920a25d71b57054313d801ec36e2	no	12
Warnings:					
Information:					

29	Non Patent Literature	nextgenerationmobilemultimediacommun.pdf	531664 90d5c50477ab9105fec69b5430cc905fefece34e	no	15
Warnings:					
Information:					
30	Non Patent Literature	zambelliiissmoothstreaming.pdf	1141278 6723e490d4df0d9c42d4164b8e18ac271dbcd2bc	no	17
Warnings:					
Information:					
31	Non Patent Literature	inletadaptivedeliveriphone.pdf	460973 c898acadd4149bb8972ba8e894658267411 29506	no	2
Warnings:					
Information:					
32	Information Disclosure Statement (IDS) Form (SB08)	02359aids1449.pdf	105211 21387108a96de9d042267c2f03e5d2fa66ab8976	no	4
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
		Total Files Size (in bytes):	79891850		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being electronically filed with the United States Patent and Trademark Office on September 8, 2014 at or before 11:59 p.m. Pacific Time under the Rules of 37 C.F.R. 1.8.

/Katrina Lee/
Katrina Lee

Appl No.	: 13/340,623	Confirmation No. 1094
Applicant	: Michael George Kiefer et al.	
Filed	: December 29, 2011	
Title	: Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information	
TC/A.U.	: 2434	
Examiner	: Abiy Getachew	
Docket No.	: D1-02359A	
Customer No.	: 71897	

INFORMATION DISCLOSURE STATEMENT § 1.97(b)

MAIL STOP AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

2400 E. Katella, Suite 1050
Anaheim, CA 92806
September 8, 2014

Commissioner

In compliance with the duty of disclosure under 37 C.F.R. §§ 1.56, 1.97 and 1.98, and in accordance with the provisions in the MPEP §§ 609 and 707.05(b), enclosed is FORM PTO/SB/08A/B with a listing of references that are known to applicant. Copies of each of the references listed under "Non-Patent Literature Documents" are enclosed.

It is respectfully requested that these references be considered in the examination of this application and identified on the list of references cited on the patent issuing on this application. Applicant also requests that an initialed copy of said FORM PTO/SB/08A/B be entered in the application file, and returned to the Applicant with the next official communication from the Office in accordance with MPEP § 609.

It is believed that there are no fees due in connection with this filing. If there are any fees due in connection with this filing, the Commissioner is authorized to charge Deposit Account 50-4407. Please show our docket number with any charge to the Deposit Account.

Respectfully submitted,

KPPB LLP

/Jason D. Wolfe/
Jason D. Wolfe
Registration No. 71,327
949.852.0000

JDW/kl

Enclosure(s)

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0843
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

71897 7590 06/06/2014

KPPB LLP
2400 EAST KATELLA AVENUE
SUITE 1050
ANAHEIM, CA 92806

EXAMINER
GETACHEW, ABIY

ART UNIT	PAPER NUMBER
2434	

DATE MAILED: 06/06/2014

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/340,623	12/29/2011	Michael George Kiefer	D1:02359A	1094

TITLE OF INVENTION: Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	UNDISCOUNTED	$960	$0	$0	$960	09/08/2014

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the ENTITY STATUS shown above. If the ENTITY STATUS is shown as SMALL or MICRO, verify whether entitlement to that entity status still applies.

If the ENTITY STATUS is the same as shown above, pay the TOTAL FEE(S) DUE shown above.

If the ENTITY STATUS is changed from that shown above, on PART B - FEE(S) TRANSMITTAL, complete section number 5 titled "Change in Entity Status (from status indicated above)".

For purposes of this notice, small entity fees are 1/2 the amount of undiscounted fees, and micro entity fees are 1/2 the amount of small entity fees.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0844
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

71897 7590 06/06/2014
KPPB LLP
2400 EAST KATELLA AVENUE
SUITE 1050
ANAHEIM, CA 92806

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/340,623	12/29/2011	Michael George Kiefer	D1:02359A	1094

TITLE OF INVENTION: Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	UNDISCOUNTED	$960	$0	$0	$960	09/08/2014

EXAMINER	ART UNIT	CLASS-SUBCLASS
GETACHEW, ABIY	2434	713-160000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) The names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) The name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 _____
2 _____
3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:
- ❑ Issue Fee
- ❑ Publication Fee (No small entity discount permitted)
- ❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
- ❑ A check is enclosed.
- ❑ Payment by credit card. Form PTO-2038 is attached.
- ❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credits any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
- ❑ Applicant certifying micro entity status. See 37 CFR 1.29 — NOTE: Absent a valid certification of Micro Entity Status (see forms PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.
- ❑ Applicant asserting small entity status. See 37 CFR 1.27 — NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.
- ❑ Applicant changing to regular undiscounted fee status. — NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: This form must be signed in accordance with 37 CFR 1.31 and 1.33. See 37 CFR 1.4 for signature requirements and certifications.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

PTOL-85 Part B (10-13) Approved for use through 10/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0845
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/340,623	12/29/2011	Michael George Kiefer	D1:02359A	1094

71897 7590 06/06/2014

KPPB LLP
2400 EAST KATELLA AVENUE
SUITE 1050
ANAHEIM, CA 92806

EXAMINER	
GETACHEW, ABIY	
ART UNIT	PAPER NUMBER
2434	

DATE MAILED: 06/06/2014

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(Applications filed on or after May 29, 2000)

The Office has discontinued providing a Patent Term Adjustment (PTA) calculation with the Notice of Allowance.

Section 1(h)(2) of the AIA Technical Corrections Act amended 35 U.S.C. 154(b)(3)(B)(i) to eliminate the requirement that the Office provide a patent term adjustment determination with the notice of allowance. See Revisions to Patent Term Adjustment, 78 Fed. Reg. 19416, 19417 (Apr. 1, 2013). Therefore, the Office is no longer providing an initial patent term adjustment determination with the notice of allowance. The Office will continue to provide a patent term adjustment determination with the Issue Notification Letter that is mailed to applicant approximately three weeks prior to the issue date of the patent, and will include the patent term adjustment on the patent. Any request for reconsideration of the patent term adjustment determination (or reinstatement of patent term adjustment) should follow the process outlined in 37 CFR 1.705.

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

PTOL-85 (Rev. 02/11)

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0846
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

OMB Clearance and PRA Burden Statement for PTOL-85 Part B

The Paperwork Reduction Act (PRA) of 1995 requires Federal agencies to obtain Office of Management and Budget approval before requesting most types of information from the public. When OMB approves an agency request to collect information from the public, OMB (i) provides a valid OMB Control Number and expiration date for the agency to display on the instrument that will be used to collect the information and (ii) requires the agency to inform the public about the OMB Control Number's legal significance in accordance with 5 CFR 1320.5(b).

The information collected by PTOL-85 Part B is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450. Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No. 13/340,623	Applicant(s) KIEFER ET AL.	
	Examiner ABIY GETACHEW	Art Unit 2434	AIA (First Inventor to File) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *05/27/2014*.
   ☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on_____.
2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.
3. ☒ The allowed claim(s) is/are *1-2, 4-6, 8-10, 12-19, 21-23, 25-29 and 32-38* . As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov .
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   Certified copies:

   a) ☐ All b) ☐ Some *c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date 05/22/2014
3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
4. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
5. ☐ Examiner's Amendment/Comment
6. ☐ Examiner's Statement of Reasons for Allowance
7. ☒ Other *Examiner's Statement*.

/ABIY GETACHEW/ Examiner, Art Unit 2434	/TESHOME HAILU/ Primary Examiner, Art Unit 2434

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0848
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

DETAILED ACTION

1. The present application is being examined under the pre-AIA first to invent provisions.

Response to Arguments

2. Applicant's arguments/Amendments, filed 05/27/2014 have been fully considered and are persuasive.

Information Disclosure Statement

3. The information disclosure statement (IDS) submitted on 05/22/2014 are being considered by the examiner.

Allowable Subject Matter

4. Claims 1-2, 4-6, 8-10, 12-19, 21-23, 25-29 and 32-38 are allowed. No reason for allowance is needed as the record is clear in light of applicant's arguments and specification.

5. According to MPEP 1302.14 (I): "In most cases, the examiner's actions and the applicant's replies make evident the reasons for allowance, satisfying the "record as a whole" proviso of the rule. This is particularly true when applicant fully complies with 37 CFR 1.111 (b) and (c) and 37 CFR 1.133(b). Thus, where the examiner's actions clearly point out the reasons for rejection and the applicant's reply explicitly presents reasons why claims are patentable over the reference, the reasons for allowance are in all probability evident from the record and no statement should be necessary."

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to ABIY GETACHEW whose telephone number is (571)272-6932. The examiner can normally be reached on Monday to Friday 8Am to 4:30Pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kambiz Zand can be reached on (571)272-3811. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/ABIY GETACHEW/
Examiner, Art Unit 2434

A.G.
May 30, 2014

/TESHOME HAILU/
Primary Examiner, Art Unit 2434

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13340623	KIEFER ET AL.
	Examiner	**Art Unit**
	ABIY GETACHEW	2434

CPC							
Symbol						**Type**	**Version**
H04L		2209	/	603		A	2013-01-01
H04L		9	/	0819		F	2013-01-01
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				
			/				

CPC Combination Sets								
Symbol					**Type**	**Set**	**Ranking**	**Version**
			/					
			/					

/ABIY GETACHEW/ Examiner.Art Unit 2434	05/29/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	31	
/TESHOME HAILU/ Primary Examiner.Art Unit 2434	06/02/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	12a

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13340623	KIEFER ET AL.
	Examiner	**Art Unit**
	ABIY GETACHEW	2434

US ORIGINAL CLASSIFICATION		INTERNATIONAL CLASSIFICATION										
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED					
713	160	H	0	4	L	29 / 06 (2006.01)						

CROSS REFERENCE(S)					
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				

/ABIY GETACHEW/ Examiner.Art Unit 2434	05/29/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	31	
/TESHOME HAILU/ Primary Examiner.Art Unit 2434	06/02/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	12a

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13340623	KIEFER ET AL.
	Examiner	**Art Unit**
	ABIY GETACHEW	2434

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	13	17	23	33										
2	2	15	18	24	34										
	3	16	19	25	35										
3	4		20	26	36										
4	5	17	21	30	37										
5	6	27	22	31	38										
	7	28	23		39										
6	8		24												
7	9	29	25												
8	10	18	26												
	11	19	27												
9	12	20	28												
10	13	21	29												
11	14		30												
12	15		31												
14	16	22	32												

/ABIY GETACHEW/ Examiner.Art Unit 2434	05/29/2014	**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	31	
/TESHOME HAILU/ Primary Examiner.Art Unit 2434	06/02/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	12a

U.S. Patent and Trademark Office

Part of Paper No. 20140530

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	13340623	KIEFER ET AL.
	Examiner	Art Unit
	ABIY GETACHEW	2434

CPC- SEARCHED		
Symbol	Date	Examiner

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
713	160	10/16/2013	A.G.

SEARCH NOTES		
Search Notes	Date	Examiner
Class/subclass, Text search and google keyword search	10/16/2013	A.G.
Search has been updated.	05/29/2014	A.G.
Inventourship search has been updated.	05/29/2014	A.G

INTERFERENCE SEARCH			
US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner
713	160	05/29/2014	A.G.
	713/160.ccls. and ((((access$ play$ play$1back playback) with (Common "same" identical$5) with Key) and((access$ play$ play$1back playback) with (common "same" identical$5) with key) and ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)).clm.	05/29/2014	A.G.

/ABIY GETACHEW/ Examiner.Art Unit 2434	

INTERFERENCE SEARCH			
US Class/ CPC Symbol	**US Subclass / CPC Group**	**Date**	**Examiner**
	processor and (memory with client with application) and ((((access$ play$ play$1back playback) with (Common "same" identical$5) with Key) and((access$ play$ play$1back playback) with (common "same" identical$5) with key) and ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)).clm.	05/29/2014	A.G.

/ABIY GETACHEW/ Examiner.Art Unit 2434	

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	2	("20090013195").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/07 11:23
S2	8	(("20090132721") or ("20090290706") or ("20090328228") or ("20130061040")).PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/07 11:27
S3	1	(13/340623).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/07 11:28
S4	18	(playback with device) with (processor with memory with client with application)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 09:49
S5	2	("20070024706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 10:03
S8	2	("20090290706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 10:38
S9	8	(("20090132721") or ("20090290706") or ("20090328228") or ("20130061040")).PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 10:56
S10	17797	(playback with device)	USPAT	OR	OFF	2013/10/08

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0856
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

						11:02
S14	1	(processor) with (memory with client with application) same (index with file) same (Streams with content with encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 11:05
S15	3	(processor) near20 (index with file) same (Streams with content with encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 11:06
S16	1	((plurality with streams) with (encrypt$3 adj key)) same (index$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/08 12:26
S17	4	((plurality with streams) with (encrypt$3 adj key)) same (index$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2013/10/08 12:27
S18	11	((plurality with streams) with (encrypt$3 adj key)) same (index$3 list$3 or index)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2013/10/09 11:02
S22	4	((plurality with streams) with (encrypt$3 adj key)) same (index$3 list$3 or index) and (playback with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2013/10/09 11:04
S23	2	((plurality with streams) near20 (index$3 list$3 or index)) near20 (identify or identifying or identified) same (plurality with alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2013/10/09 11:18
S25	2	("20130032518").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 11:32
S26	2	("20090290706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO;	OR	OFF	2013/10/09 11:48

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0857
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			JPO; DERWENT; IBM_TDB			
S27	2	("20130061040").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 11:51
S29	2	("20130061040").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 11:55
S30	2	("20130061045").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 11:55
S32	2	(request with index with file with identifying with alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 12:33
S33	4	("2003030324").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 12:39
S34	0	("20090387258").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 12:39
S35	0	("200900387258").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 12:39
S36	2	("20090307258").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 12:40
S37	2	("20090132599").PN.	US-PGPUB; USPAT;	OR	OFF	2013/10/09 12:41

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0858
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			
S38	0	(index with file) with (alternative streams of protected content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 12:55
S39	1	(index with file) with (alternative streams protected content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 12:55
S40	1	(index) with (alternative streams protected content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 12:55
S41	9	(index) with (streams protected content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 12:55
S42	2	("20080301734").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 12:59
S43	5	(index) near20 (streams protected content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:02
S44	2	(index) near20 (streams protected content).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:04
S46	5	(index or indexing or indexed) near20 (streams with content with protect$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:41

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0859
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

S47	2	(index or indexing or indexed) near20 (streams with content with protect$3).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:42
S48	30	(index or indexing or indexed) near20 (streams with content).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:43
S49	2	S48 and (streams with protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:44
S50	2	S48 and (streams same encrypted same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:45
S51	3	S48 and (encrypted same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 13:46
S52	4	(index or indexing or indexed) near20 (streams with content) near20 (protected with content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:05
S53	2	("20030078930").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 14:06
S54	4	(index or indexing or indexed) near20 (protected with content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:08
S55	60	(index or indexing or indexed) near20 (content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO;	WITH	OFF	2013/10/09 14:08

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0860
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			DERWENT; IBM_TDB			
S56	4	(index or indexing or indexed) near20 (streams with content) near20 (content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:08
S57	4	(index or indexing or indexed) near20 (streams) near20 (content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:08
S58	4	(index or indexing or indexed) near20 (streams) near20 (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:09
S59	4	(index or indexing or indexed) near20 (streams) with (content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:10
S60	13	(index or indexing or indexed) near20 (streams same protect$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:11
S61	58	("5465353" | "5613004" | "5664016" | "5689639" | "5809498" | "6101602" | "6363463").PN. OR ("6671407").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2013/10/09 14:21
S62	6	(streams with content) near20 (protected with content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 14:41
S63	20490	(client with application same processor)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/09 15:13
S64	1	(client with application same processor) with (index or indexing or indexed) near20 (Streams with content with encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/10/09 15:16

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0861
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S65	48	(client with application same processor) with (index or indexing or indexed)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:16
S66	1	(client with application same processor) with (index or indexing or indexed) near20 (Streams with encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:17
S67	1	(client with application same processor) with (index or indexing or indexed) near20 (Streams with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:17
S68	2	(client with application same processor) with (index or indexing or indexed) near20 (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:18
S69	12	(client with application same processor) with (index or indexing or indexed) near20 (Stream$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:19
S70	1	(client with application same processor) with (index or indexing or indexed) near20 (protect with Stream$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:19
S71	1	(client with application same processor) with (index or indexing or indexed) near20 (protect same Stream$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:20
S72	2	(client with application same processor) with (index or indexing or indexed) near20 (Stream$3) and (cryptographic with information)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:21
S73	2	(client with application same processor) with (index or indexing or indexed) near20 (Stream$3) and (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO;	OR	OFF	2013/10/09 15:21

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0862
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			JPO; DERWENT; IBM_TDB			
S74	6	(application same processor) with (index or indexing or indexed) near20 (Stream$3) and (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:21
S75	3	(processor) with (index or indexing or indexed) near20 (Stream$3) same (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:23
S76	4	(processor) with (index or indexing or indexed) near20 (Stream$3) same (protected)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:24
S77	69	(index or indexing or indexed) near20 (Stream$3) same (protected)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:24
S78	5	S55 and S77	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 15:25
S79	2	(plurality with alternative with streams with protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:50
S80	308	(plurality with alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:50
S81	2	(plurality with alternative with streams)same (cryptographic with information)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:51
S82	2	(plurality with alternative with streams)same (cryptographic)	US-PGPUB; USPAT;	OR	OFF	2013/10/09 17:51

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0863
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			
S83	2	(plurality with alternative with streams) near20 (content with encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:52
S84	2	(plurality with alternative with streams) near20 (encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:52
S85	4	(plurality with alternative with streams) near20 (key or encryption or decryption or code or encryp$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:54
S86	2	(plurality with alternative with streams) near20 (protect$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:56
S87	49731	(plurality with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:56
S88	51	(plurality with streams)same (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:56
S89	27	(plurality with streams) near20 (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:56
S90	2	(plurality with streams) near20 (protect$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:57

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0864
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

S91	27	S88 and S89	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 17:57
S92	9	(Request) with (index) near20 (plurality with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:01
S93	1	(Request) with (index) near20 (plurality with streams)same (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:02
S94	1	(index) near20 (plurality with streams)same (cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:02
S95	3	(Request) with (index) near20 (protect$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:03
S96	17	(index) near20 (protect$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:03
S97	1	(plurality with streams) near20 (index) near20 (protect$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:03
S98	4	(streams) near20 (index) near20 (protect$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/09 18:04
S99	18	(streams) near20 (protect$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO;	OR	OFF	2013/10/09 18:04

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0865
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			DERWENT; IBM_TDB			
S101	13	(streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:15
S102	9	(streams with protected with video same DRM) same (DRM same encrypt$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:15
S103	3	(streams with protected with video same DRM) near20 (encrypt$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:16
S104	0	(memory with client with application) near20 (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:18
S105	0	(memory same application) near20 (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:19
S106	1	(client with application) near20 (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:19
S107	1	(Application) near20 (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:20
S108	3	(Application) same (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:20
S109	3	(processor) same (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR;	OR	OFF	2013/10/10 10:23

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0866
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			FPRS; EPO; JPO; DERWENT; IBM_TDB			
S111	1	(application with configures with processor) same (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:24
S112	1	(application same processor) same (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:25
S113	4	(application same processor) and (streams with protected with video same DRM)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 10:25
S114	2	"13235729"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 12:26
S115	2	(streams with protected with video same DRM) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:14
S116	2	(streams same video same DRM) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:15
S117	20	(streams same video) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:15
S118	1	S117 and (DRM with Header same Tracks)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:16
S119	10	(DRM with Header same Tracks)	US-PGPUB;	OR	OFF	2013/10/10

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0867
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			14:16
S120	1	S119 and (streams same video) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:17
S121	1	S119 and (streams) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:17
S122	1	S119 and (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:17
S123	2	("20060245727").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:21
S124	2	DRM same (streams same video) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:27
S125	9	index same (streams same video) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:28
S126	20	(Matroska with container with files)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:34
S127	2	S126 and (index or indexing or indexed) near20 (content with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	WITH	OFF	2013/10/10 14:35

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0868
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S128	17	S126 and (index or indexing or indexed)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	WITH	OFF	2013/10/10 14:35
S129	20	S117 and (streams same video) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:52
S130	3	S129 and (index or indexing or indexed) near20 (Stream$3) same (protected)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:55
S131	3	S129 and (index or indexing or indexed) same (Stream$3) same (protected)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:56
S132	3	S129 and (Stream$3) same (protected)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:56
S133	16	S129 and (index or indexing or indexed)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:56
S134	7	S129 and (processor) with (index or indexing or indexed)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:57
S135	2	(streams with protected) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:57
S136	9	S125 and (streams same video) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO;	OR	OFF	2013/10/10 14:58

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0869
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			JPO; DERWENT; IBM_TDB			
S137	9	S125 and (streams) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 14:59
S138	21	(streams) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:00
S139	2	(DRM with Header) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:01
S140	2	(DRM) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:01
S141	3	(DRM) same (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:02
S142	7	(DRM with Header same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:49
S143	20	(DRM same Header same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:50
S144	2	(streams) near20 (Matroska) near20 (DRM same Header same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:50
S145	2	(Matroska) near20 (DRM same Header same cryptographic)	US-PGPUB; USPAT;	OR	OFF	2013/10/10 15:51

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0870
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			
S146	2	(DRM same Header same cryptographic) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:51
S147	2	(DRM same cryptographic) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:51
S148	2	(DRM same Header same cryptographic) same (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:51
S149	4	(DRM same Header same cryptographic) and (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/10 15:52
S150	54	(bitrate with stream$4) same (alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/15 10:46
S151	6	(bitrate with stream$4) same (alternative with streams) same (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/15 10:46
S152	8	(bitrate with stream$4) same (alternative with streams) same (protected cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/15 10:47
S153	6	(bitrate with stream$4) same (alternative with streams) same (protected crypto$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/15 10:47

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0871
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

S154	1	713/160.ccls. and (bitrate with stream$4) same (alternative with streams) same (protected crypto$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/10/16 15:16
S155	1	(13/340623).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 12:47
S156	6105	("Digital Rights Management" or DRM)	USPAT	OR	OFF	2013/11/05 12:51
S158	3456	("Digital Rights Management" or DRM) same (DRM with (encryption or decryption or code))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 12:54
S159	2	("Digital Rights Management" or DRM) same (DRM with (encryption or decryption or code)) and (Matroska with container with files) with (Hypertext with Transfer)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 12:56
S160	4	(("20090132721") or ("20130061045")).PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 12:58
S162	4	("Digital Rights Management" or DRM) same (DRM with (encryption or decryption or code)) and ("top level index file")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:02
S163	35	("top level index file")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:06
S164	3	("Digital Rights Management" or DRM) near20 ("top level index file")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:09
S165	10	("Digital Rights Management" or DRM) and ("top level index file")	US-PGPUB; USPAT; USOCR;	OR	OFF	2013/11/05 13:24

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0872
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			FPRS; EPO; JPO; DERWENT; IBM_TDB			
S166	5	(("20040039916") or ("20100138903")).PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:29
S167	35	("top level index file")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:48
S168	9	("top level index file").clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:52
S169	2	("20060245727").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 13:55
S170	2	("20090132599").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 14:05
S171	6	("2009/0132599").URPN.	USPAT	OR	OFF	2013/11/05 14:17
S172	27	processor and memory and ("top level index file")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 14:19
S173	20	processor and memory and ("top level index file")and (alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 14:19
S175	16	S173 and (client with application)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/11/05 14:20

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0873
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S176	6	(("20130046902") or ("8386621") or ("20090293116")).PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 14:24
S177	203241	(streams witrh protected with content with encrypted with common with cryptographic)	USPAT	OR	OFF	2013/11/05 16:21
S178	484775	(streams witrh protected with content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:21
S179	2	(streams with protected with content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:21
S180	40	(content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:23
S181	2	(streams) near20 (content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:24
S182	2	(playback) near20 (content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:24
S183	1015	(content with encrypted with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:24
S184	2	(streams with protected with content with encrypted with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:24

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0874
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

S185	20	(streams same content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:25
S186	2	(plurality with streams same content with encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:25
S187	2	(streams same content with encrypted with common with cryptographic) and (access with protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:26
S188	20	(streams same content with encrypted with common with cryptographic) and (access with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:26
S189	6	S188 and (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:27
S190	2	S188 and (alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:33
S191	2	(streams same content with encrypted with common with cryptographic) and (access with content) and (alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:34
S192	2	(content with encrypted with common with cryptographic) and (alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:34
S193	5972	(alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO;	OR	OFF	2013/11/05 16:34

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0875
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			DERWENT; IBM_TDB			
S194	2	(alternative with streams) near20 (content with encrypted with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:35
S195	2	(alternative with streams) near20 (encrypted with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:35
S196	2	(alternative with streams) near20 (content with encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:35
S197	2	(alternative with streams) near20 (encrypted with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:35
S198	23	(alternative with streams) near20 (encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:36
S199	3	(alternative with streams) near20 (encrypted) same common	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:36
S200	2	(alternative with streams) near20 (encrypted) same (cryptographic with information)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:37
S201	1907	(encrypted) near20 (cryptographic with information)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:38
S202	2	S201 and (alternative with streams) near20 (encrypted)	US-PGPUB; USPAT; USOCR;	OR	OFF	2013/11/05 16:38

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0876
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			FPRS; EPO; JPO; DERWENT; IBM_TDB			
S203	516	(encrypt$3 with common with cryptographic with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:39
S204	1	(alternative with streams) near20 (encrypt$3 with common with cryptographic with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:39
S205	1	(streams) near20 (encrypt$3 with common with cryptographic with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:39
S206	176	(streams data content) near20 (encrypt$3 with common with cryptographic with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:41
S207	11	(streams data content) near20 (encrypt$3 with common with cryptographic with key) and (playback with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:41
S208	5	(streams data content) near20 (encrypt$3 with common with cryptographic with key) and (playback with content) and (index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:42
S209	6	(streams data content) near20 (encrypt$3 with common with cryptographic with key) and (playback with content).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:47
S210	29	(streams data content) near20 (encrypt$3 with common with cryptographic with key).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:48
S211	2	("20040193876").PN.	US-PGPUB;	OR	OFF	2013/11/05

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0877
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			16:51
S212	2	("20020078353").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 16:58
S213	1	("Digital Rights Management" or DRM) near20 (encrypt$3 with common with cryptographic with key).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 17:01
S214	1	(DRM) near20 (encrypt$3 with common with cryptographic with key).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 17:01
S215	3	("Digital Rights Management" or DRM) and (encrypt$3 with common with cryptographic with key).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 17:02
S216	1	("Digital Rights Management" or DRM) near20 (encrypt$3 with common with cryptographic with key).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 17:03
S217	1	(DRM) near20 (encrypt$3 with common with cryptographic with key).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 17:03
S218	2	(DRM) near20 (encrypt$3 with common with cryptographic).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/05 17:03
S219	1120	(hierarchical with index with structure)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/11/06 15:13

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0878
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S220	3	(hierarchical with index with structure) and Soroushian	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:13
S221	4	(hierarchical with index with structure) and (top with level with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:18
S224	2	(hierarchical with index with structure) and (top with level with index with file) and (content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:25
S227	2	(hierarchical with index with structure) same (top with level with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:26
S230	13	(streams) same (top with level with index with file)near20 (content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:47
S231	12	(streams) same (top with level with index with file)near20 (playback with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:47
S232	3	(streams) same (top with level with index with file)near20 (playback with content).clm.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:49
S233	2	("7295752").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:51
S234	1	(10/067550).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO;	OR	OFF	2013/11/06 15:53

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0879
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			JPO; DERWENT; IBM_TDB			
S235	2	("20020150243").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 15:57
S236	14	(top with level with index with file)near20 (playback with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:01
S237	2	("20080301734").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:09
S238	2	("20070074254").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:13
S239	35	("top level index file")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:17
S240	9	("top level index file") and (encrypted same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:18
S241	1	(level with index) near20 (encrypted same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:19
S242	1	(level with index) near20 (common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:20
S243	45	(level with index) near20 (encrypted)	US-PGPUB; USPAT;	OR	OFF	2013/11/06 16:21

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0880
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			
S244	1	(top with level with index) near20 (encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:22
S245	72	(top with level with index) and (encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:23
S246	41	(top with level with index) and (encrypted same using)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:23
S247	19	(top with level with index) and (encrypted same using with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:25
S248	1	(top with level with index) near20 (encrypted same encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:48
S249	45	(level with index) near20 (encrypted same encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:48
S250	6063	(alternative with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:51
S251	2	(alternative with streams) near20 (encrypted with using with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:52

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0881
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

S252	2	(alternative with streams) near20 (encrypted with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:52
S253	2	(alternative with streams) near20 (encrypt$5 with common with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:52
S254	3	(alternative with streams) near20 (encrypt$5 with cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:53
S255	23	(alternative with streams) near20 (encrypted with encrypt$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:53
S256	117	(("top level index file") or (top with level with index with file))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:54
S257	2	S255 and S256	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:55
S258	8	(alternative with streams) near20 (encrypted with using with encrypt$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/06 16:55
S259	2	("20080046369").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 09:51
S261	117	(top with level with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO;	OR	OFF	2013/11/07 10:30

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0882
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			DERWENT; IBM_TDB			
S265	3	((top high) with (level layer) with (file$1 document$1 stream content)) near20 (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:39
S266	1	((top) with (level) with (stream content)) near20 (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:44
S269	35	(top with level) near20 (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:46
S270	1	S269 and (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:46
S273	32	((top high) with (index file)) near20 (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:48
S274	1	((top high) with (level) with (index)) near20 (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:49
S277	2	((top) with (level) with (index)) same (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:51
S278	8154	((encrypted with using) with (Symmetric common secret) near3 (key))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:52
S279	11092	((encrypted with using) with (Symmetric common secret))	US-PGPUB; USPAT; USOCR;	OR	OFF	2013/11/07 10:52

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0883
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			FPRS; EPO; JPO; DERWENT; IBM_TDB			
S280	1	((top with level with index with file) near20 (Symmetric common secret))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:55
S285	15	((top with index) near20 (Symmetric))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:56
S286	3	"380"/$.ccls. and (top with level with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:57
S287	2	"380"/$.ccls. and ((top high) with (level layer) with (file$1 document$1 stream content)) near20 (Symmetric common secret) near3 (key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 10:58
S290	742	streams near20 ((top high) with (level layer) with (file$1 document$1 stream content))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 11:40
S291	7	streams near20 ((top high) with (level layer) with (file$1 document$1 stream content)) near20 (Symmetric common secret)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 11:41
S292	1	streams near20 ((top) with (level layer) with (file$1)) near20 (Symmetric common secret)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 11:43
S294	17	streams and ((top) with (level layer) with (file$1)) near20 (Symmetric secret)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 11:43
S295	1	streams and ((top) with (level layer)	US-PGPUB;	OR	OFF	2013/11/07

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0884
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

		with (file$1)) near20 (Symmetric encryp$3)	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			12:19
S296	11	(streams frame packet stream bitstream) and ((top) with (level layer) with (file$1)) near20 (Symmetric encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:22
S297	3	(top with level with index with file) and (streams frame packet stream bitstream) near20 (Symmetric encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:24
S298	49	(top with index with file) and (streams frame packet stream bitstream) near20 (Symmetric encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:24
S299	1	(top with index with file) near20 ((common with cryptographic) and (playback with content))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:33
S300	1	(top with index with file) near20 ((Symmetric encrypt$3) and (playback with content))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:34
S301	1	(top with index with file) near20 ((Symmetric encrypt$3) and (content))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:35
S302	2	(top with index with file) near20 (Symmetric encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:35
S303	452	(common with cryptographic with information)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/11/07 12:37

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0885
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S304	5	(top with level with index with file) and S303	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:37
S305	5	(top with level with index) and S303	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:37
S306	16	(top with level with file) and S303	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:38
S308	4	(top with file) near20 (encrypted with using with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:41
S309	1	(11/611188).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 12:43
S310	91230	(content with encrypt$3 with using encrypt$3)	USPAT	OR	OFF	2013/11/07 12:59
S312	1	(top with level with index with file) near20 (content with encrypt$3 with using encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:01
S313	22	(top with level with index with file) and (content with encrypt$3 with using encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:01
S314	7	(top with level with index with file) and (content with encrypt$3 with using encrypt$3) and (plurality with streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:02
S315	14	(top with level with index with file)	US-PGPUB;	OR	OFF	2013/11/07

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0886
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

		and (content with encrypt$3 with using encrypt$3) and (streams)	USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			13:02
S316	22	(top with level with index with file) and (content with encrypt$3 with using encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:18
S317	22	(top with level with index with file) and (encrypt$3 with using encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:22
S318	600	((top high) with (level layer) with (file$1 document$1 stream content)) near20 (Symmetric common secret)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:23
S319	210	S318 and (encrypt$3 with using encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:24
S320	1	S319 and (top with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:24
S321	1	S319 and (top with index)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:24
S322	32	S319 and (index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:24
S323	20	S319 and (index with file) and streams	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/11/07 13:25

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0887
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S324	2	DRM and ((encryption decrption) with (content)) and (matroska with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:29
S325	2	("20090158318").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:41
S326	254	(distribute with common with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:46
S329	3	(index with file) and (distribute with common with key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 13:47
S330	2	("20090290706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/07 15:27
S331	2	("20090328228").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 13:21
S332	8	(("20090132721") or ("20090290706") or ("20090328228") or ("20130061040")).PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 13:55
S333	77792	(plurality same streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:10
S334	41	(plurality same streams) same (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO;	OR	OFF	2013/11/08 14:11

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0888
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			JPO; DERWENT; IBM_TDB			
S335	8	(plurality same streams) near10 (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:11
S336	1	(index with file) near10 (plurality same streams) near10 (protected with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:12
S337	31	(plurality same streams) and (top with level with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:15
S338	117	(top with level with index with file)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:22
S340	1	(top with level with index with file) near20 (common crypto$6)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:23
S345	14	(top with level with index with file) near20 (playback with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:25
S346	1	(top with level with index with file) near20 encrypt$4 same (playback with content)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:26
S348	1	(top with level with index with file) near20 encrypt$4	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 14:27
S349	4	(top with level with index with file) near20 (playback with content).clm.	US-PGPUB; USPAT;	OR	OFF	2013/11/08 14:28

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0889
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB			
S350	117	(encrypt$ with stream$ with (common "same" identical$5) with (encrypt$ with Stream$ with (common "same" identical$5) with key)"ti.key").clm. (encrypt$ with stream$ with (common "same" identical$5) with key).ab.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 16:18
S351	110	S350 and @ad<"20110901"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 16:19
S353	30	S351 and ("713"/$ "726"/$380/$).ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 16:21
S354	7	S353 and ((access$ play$ play$1back playback) with (Common "same" identical$5) with Key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 16:22
S355	0	S354 and (((access$ play$ play$1back playback) with (Common "same" identical$5) with Key).ti. ((access$ play$ play$1back playback) with (common "same" identical$5) with key).ab. ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/08 16:28
S356	2	("20070067646").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/11 12:10
S357	2	("20130046902").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/11 12:22
S358	22	(multimedia with digital with data with transmission with procedure)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/11/11 13:06

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0890
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S359	12733	((access$ play$ play$1back playback) with (Common "same" identical$5) with Key)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/11 13:30
S360	1	(top with level with index with file) near20 encrypt$4 same (playback)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/11 14:32
S361	2	("20080172341").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 10:12
S362	2	("20130046902").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:43
S363	87	(stream$4 same bitrate with stream$3 with encoded with media)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:51
S364	11	(stream$4 same bitrate with stream$3 with encoded with media) same (Matroska same container ame files same Hypertext with Transfer with Protocol)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:52
S365	10	(stream$4 same bitrate with stream$3 with encoded with media) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:53
S366	87	(stream$4 same bitrate with stream$3 with encoded with media)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:53
S367	4127	(stream$3 with encoded with media)	US-PGPUB; USPAT; USOCR; FPRS; EPO;	OR	OFF	2013/11/12 11:53

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0891
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			JPO; DERWENT; IBM_TDB			
S368	12	(stream$3 with encoded with media) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:53
S369	42	(stream$3) near20 (Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:55
S370	9	S369 and processor and (memory with client with application)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 11:56
S371	2	("6005621").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 12:00
S372	1	(stream$3 with encoded with media with Matroska)	USPAT	OR	OFF	2013/11/12 14:28
S373	0	(stream$3 with encoded with media near20 Matroska)	USPAT	OR	OFF	2013/11/12 14:29
S374	1	(stream$3 with encoded with media same Matroska)	USPAT	OR	OFF	2013/11/12 14:30
S375	3	(stream$3 with encoded same Matroska)	USPAT	OR	OFF	2013/11/12 14:30
S377	11	("top level index file" near20 Matroska)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 14:32
S378	2	("20070140647").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 14:35
S379	159	Matroska	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 15:52

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0892
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

S380	143	Matroska and (plurality adj$3 streams)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 15:52
S381	0	Matroska and (plurality adj$3 streams) near20 (common with crypt$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 15:53
S382	2	Matroska and (plurality adj$3 streams) near20 (encrypt$3 with use$3 same cryptographic)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 15:54
S383	6	Matroska and (plurality adj$3 streams) near20 (encrypt$3 with use$3 same encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 15:55
S384	6	Matroska and (streams) near20 (encrypt$3 with use$3 same encrypt$3)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 16:07
S385	2	("20100107260").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 16:53
S386	2	("20110135090").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 17:10
S387	2	("20040081333").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/12 17:21
S388	2	("20090290706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO;	OR	OFF	2013/11/14 16:27

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0893
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			DERWENT; IBM_TDB			
S389	1	(13/258981).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/14 16:33
S390	2	("20110283361").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/15 14:33
S391	2	("2009290706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 11:31
S392	2	("20090290706").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 11:31
S393	2	("20070067646").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 11:50
S394	1	(13/340623).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 14:10
S395	4	("2011103364").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 14:51
S396	1	(13/579020).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 14:51
S397	2	("20090328228").PN.	US-PGPUB; USPAT; USOCR;	OR	OFF	2013/11/16 14:52

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0894
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			FPRS; EPO; JPO; DERWENT; IBM_TDB			
S398	2	("20090132721").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/16 14:53
S401	3	"20090328228" and (protect$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 10:41
S403	36482	(protect$4)same (encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 10:48
S405	2	("20090328228").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 10:48
S411	0	S405 and (encrypted)	USPAT	OR	OFF	2013/11/17 10:56
S412	10	("digital rights management" or "DRM") near20 (protect$4 with frame with video)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 11:08
S413	2	("digital rights management" or "DRM") near20 (protect$4 with frame with video) same (video same encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 11:11
S414	9	("digital rights management" or "DRM") near20 (protect$4 with frame with video) and (video same encrypted)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 11:13
S415	4	("20111103364").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT;	OR	OFF	2013/11/17 12:45

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0895
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

			IBM_TDB			
S418	4	("2011103364").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 14:39
S419	1	(13/579020).APP.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 14:39
S420	2	("20110103364").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 15:19
S429	4	("2011103364").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 15:27
S430	1	WO-2011103364-$.did.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 15:28
S431	1	WO-2011103364-$.did.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 15:30
S432	2	("20090132721").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 16:42
S433	4	("2011103364").PN.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2013/11/17 17:08
S435	97	((Michael) near2 (Kiefer)).INV.	US-PGPUB; USPAT	OR	OFF	2014/05/30 13:57
S436	7	((Eric) near2 (Grab)).INV.	USPAT	OR	OFF	2014/05/30

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0896
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

						13:59

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S439	5	("digital rights management" or "DRM") and (protect$4 with frame with video) and (video same encrypted).clm.	USPAT; UPAD	OR	OFF	2014/05/30 13:50
S440	609	((((access$ play$ play$1back playback) with (Common "same" identical$5) with Key).ti. ((access$ play$ play$1back playback) with (common "same" identical$5) with key).ab. ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)).clm.	USPAT; UPAD	OR	OFF	2014/05/30 13:51
S441	511	((((access$ play$ play$1back playback) with (Common "same" identical$5) with Key) and((access$ play$ play$1back playback) with (common "same" identical$5) with key) and ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)).clm.	USPAT; UPAD	OR	OFF	2014/05/30 13:52
S442	6	processor and (memory with client with application) and ((((access$ play$ play$1back playback) with (Common "same" identical$5) with Key) and((access$ play$ play$1back playback) with (common "same" identical$5) with key) and ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)).clm.	USPAT; UPAD	OR	OFF	2014/05/30 13:53
S444	6	713/160.ccls. and ((((access$ play$ play$1back playback) with (Common "same" identical$5) with Key) and((access$ play$ play$1back playback) with (common "same" identical$5) with key) and ((access$ play$1back playback) with (common "same" identical$5) with key).ab.)).clm.	USPAT; UPAD	OR	OFF	2014/05/30 13:54

5/30/2014 3:34:14 PM
C:\Users\agetachew\Documents\EAST\Workspaces\13340623.wsp

Netflix, Inc. and Hulu, LLC - Ex. 1004, Page 0897
IPR2020-00558 (Netflix, Inc. and Hulu, LLC v. DivX, LLC)

Doc code: IDS
Doc description: Information Disclosure Statement (IDS) Field

PTO/SB/08a (01-10)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	13/340,623
	Filing Date	Dec 29, 2011
	First Named Inventor	Michael George Kiefer
	Art Unit	2434
	Examiner Name	Abiy Getachew
	Attorney Docket Number	D1-02359A

U.S.PATENTS

Examiner Initial*	Cite No	Patent Number	Kind Code[1]	Issue Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
/A.G./		5715403	A	1998-02-03	Stefik, Mark	
/A.G./		6141754	A	2000-10-31	Choy, David	

NON-PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[5]
/A.G./		"IBM Closes Cryptolopes Unit," December 17, 2007, CNET News, Retrieved from http://news.cnet.com/IBM-closes-Cryptolopes-unit/2100-1001_3206465.html, 3pgs.	
/A.G./		"IBM Spearheading Intellectual Property Protection Technology for Information on the Internet; Cryptolope Containers Have Arrived," May 1, 1996, Business Wire, Retrieved from http://www.thefreelibrary.com/IBM+Spearheading+Intellectual+Property+Protection+Technology+for...-a018239381, 1pg.	
/A.G./		Nelson, Michael, "IBM's Cryptolopes," Complex Objects in Digital Libraries Course, Spring 2001, Retrieved from http://www.cs.odu.edu/~mln/teaching/unc/inls210/?method=display&pkg_name=cryptolopes.pkg&element_name=cryptolopes.ppt, 12 pgs.	

EXAMINER SIGNATURE

Examiner Signature	/Abiy Getachew/ (05/30/2014)	Date Considered	05/30/2014

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. [2] Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). [3] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document.
[4] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [5] Applicant is to place a check mark here if English language translation is attached.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 1094

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
13/340,623	12/29/2011	713	2434	D1:02359A

APPLICANTS

INVENTORS

Michael George Kiefer, Scottsdale, AZ;
Eric William Grab, San Diego, CA;
Jason Braness, San Diego, CA;

**** CONTINUING DATA ***************************

This appln claims benefit of 61/530,305 09/01/2011

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

01/13/2012

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☑ No Verified and Acknowledged /ABIY GETACHEW/ Examiner's Signature	☐ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		AZ	19	31 ~~39~~	3

ADDRESS

KPPB LLP
2400 EAST KATELLA AVENUE
SUITE 1050
ANAHEIM, CA 92806
UNITED STATES

TITLE

Systems and Methods for Playing Back Alternative Streams of Protected Content Protected Using Common Cryptographic Information

FILING FEE RECEIVED		
2520	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

BIB (Rev. 05/07).