(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5798451号
(P5798451)

(45)発行日　平成27年10月21日(2015.10.21)　(24)登録日　平成27年8月28日(2015.8.28)

(51)Int.Cl.		F I		
G06F 13/00	(2006.01)	G06F	13/00	540A
G06F 12/00	(2006.01)	G06F	12/00	547H
H04N 7/173	(2011.01)	H04N	7/173	610Z

請求項の数 13　(全 27 頁)

(21)出願番号　特願2011-250057 (P2011-250057)
(22)出願日　平成23年11月15日 (2011.11.15)
(65)公開番号　特開2012-141954 (P2012-141954A)
(43)公開日　平成24年7月26日 (2012.7.26)
審査請求日　平成26年11月7日 (2014.11.7)
(31)優先権主張番号　特願2010-281012 (P2010-281012)
(32)優先日　平成22年12月16日 (2010.12.16)
(33)優先権主張国　日本国(JP)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(74)代理人　100130409
弁理士　下山　治
(74)代理人　100134175
弁理士　永川　行光

最終頁に続く

(54)【発明の名称】情報処理装置およびその方法

(57)【特許請求の範囲】
【請求項１】
マークアップ言語によって記述されたデータを入力する入力手段と、
前記データから、それぞれフラグメント記述を含む複数のユニフォームリソース識別子を取得する取得手段と、
前記複数のユニフォームリソース識別子が含むフラグメント記述に、接続可能なフラグメント記述があるか否かを判定する判定手段と、
前記接続可能なフラグメント記述があると判定された場合、前記接続可能なフラグメント記述を有するユニフォームリソース識別子を修正して、一つのユニフォームリソース識別子を生成する生成手段と、
前記生成されたユニフォームリソース識別子をサーバに送信する送信手段とを有することを特徴とする情報処理装置。
【請求項２】
さらに、前記データに含まれる、フラグメント記述を含むユニフォームリソース識別子を解析する解析手段と、
前記解析手段の解析結果として、リソースとフラグメント記述の関係を示すレコードを含むテーブルを作成する作成手段と、
前記接続可能なフラグメント記述があると判定された場合、前記接続可能なフラグメント記述を有するレコードを統合するために、前記テーブルを更新する更新手段とを有し、
前記更新されたテーブルは、前記生成手段による前記修正に用いられることを特徴とす

る請求項1に記載された情報処理装置。
【請求項３】
さらに、前記生成されたユニフォームリソース識別子に対応するストリームデータを前記サーバから取得する取得手段と、
前記取得したストリームデータを、前記生成されたユニフォームリソース識別子に対応する前記テーブルのレコードに関連付ける手段とを有することを特徴とする請求項2に記載された情報処理装置。
【請求項４】
前記リソースは動画であり、前記フラグメント記述は前記動画の時間に関する再生の開始点と終了点を示し、
前記判定手段は、前記フラグメント記述が示す終了点と前記フラグメント記述が示す開始点が一致する、前記テーブルにおいて連続するレコードを、前記接続可能なフラグメント記述を有するレコードと判定することを特徴とする請求項2または請求項3に記載された情報処理装置。
【請求項５】
前記リソースは動画であり、前記フラグメント記述は前記動画の時間に関する再生の開始点と終了点を示し、
前記判定手段は、前記フラグメント記述が示す終了点と前記フラグメント記述が示す開始点の差分が閾値以下の、前記テーブルにおいて連続するレコードを、前記接続可能なフラグメント記述を有するレコードと判定することを特徴とする請求項2または請求項3に記載された情報処理装置。
【請求項６】
前記生成手段は、前記連続するレコードの一方のフラグメント記述が示す終了点を、前記連続するレコードの他方のフラグメント記述が示す終了点に変更し、前記連続するレコードの他方を削除することを特徴とする請求項4または請求項5に記載された情報処理装置。
【請求項７】
前記閾値は、前記テーブルに基づいて、前記動画の切り出し時間の総和に対して重み係数を掛け合わせた値であることを特徴とする請求項5に記載された情報処理装置。
【請求項８】
前記リソースは動画であり、前記フラグメント記述は前記動画の空間に関する切り出し範囲を示すことを特徴とする請求項2または請求項3に記載された情報処理装置。
【請求項９】
前記生成手段は、前記テーブルにおいて連続するレコードの一方のフラグメント記述が示す切り出し範囲が、前記連続するレコードの他方のフラグメント記述が示す切り出し範囲を含むように、前記連続するレコードの一方のフラグメント記述を変更して、前記連続するレコードの他方を削除することを特徴とする請求項8に記載された情報処理装置。
【請求項１０】
前記生成手段は、前記フラグメント記述の切り出し範囲に基づき前記連続するレコードを統合する際、統合後の切り出し範囲に占める、統合前の切り出し範囲の割合に基づき、前記切り出し範囲の統合を行うか否かを判定することを特徴とする請求項9に記載された情報処理装置。
【請求項１１】
前記判定手段は、前記テーブルのリソースが時間と空間に関するフラグメント記述を含む場合、前記時間に関して前記フラグメント記述が接続可能か否かを判定し、接続可能と判定した場合は、さらに前記空間に関して前記フラグメント記述が接続可能か否かを判定する請求項2または請求項3に記載された情報処理装置。
【請求項１２】
入力手段が、マークアップ言語によって記述されたデータを入力し、
取得手段が、前記データから、それぞれフラグメント記述を含む複数のユニフォームリ

ソース識別子を取得し、
判定手段が、前記複数のユニフォームリソース識別子が含むフラグメント記述に、接続可能なフラグメント記述があるか否かを判定し、
前記接続可能なフラグメント記述があると判定された場合、生成手段が、前記接続可能なフラグメント記述を有するユニフォームリソース識別子を修正して、一つのユニフォームリソース識別子を生成し、
送信手段が、前記生成されたユニフォームリソース識別子をサーバに送信することを特徴とする情報処理方法。
【請求項１３】
コンピュータを請求項1から請求項11の何れか一項に記載された情報処理装置の各手段として機能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、動画コンテンツの一部を再生する情報処理に関する。
【背景技術】
【０００２】
動画配信サービスによって提供される動画は、ウェブ(Web)ブラウザを用いて閲覧すること可能である。また、ユーザは、自身が撮影中の動画をリアルタイムにストリーム配信することが可能なライブビデオサービスも利用可能であり、動画に関するサービスは、ユーザにとって身近な存在となった。しかし、動画データは一般にデータサイズが大きく、動画の途中部分のみを再生したい場合にコンテンツのダウンロードに時間がかかり、再生の開始に時間がかかる。
【０００３】
そこで、近年、W3C標準のMedia Fragments URI技術（例えば、非特許文献1）が注目されている。この技術によれば、フラグメント記述を含むURI (uniform resource identifier)（以下、フラグメントURI）基づいて、動画コンテンツの一部分（以下、フラグメント動画）のみをサーバから取得することが可能である。フラグメント動画の取得が可能になれば、映画など、データサイズが大きい動画においても、ユーザは、興味がある部分のみを取得して短時間に再生することができる。
【０００４】
ホスト名「sample.org」がHTTPプロトコルを用いて配信する動画v1.ogvのURIは「http://sample.org/v1.ogv」と記述される。このURIが指定されると、Webブラウザは、sample.orgにHTTP GETリクエストを送信し、サーバのレスポンスとして動画v1.ogvのバイナリデータを取得して当該動画を再生する。
【０００５】
フラグメントURIにより、動画の一部分、例えば開始点が10秒で終了点が20秒の一部分を表現すると「http://sample.org/v1.org#t=10,20」と記述される。このURIが指定されると、Webブラウザは、フラグメント記述「#t=10,20」を抽出し、下記のように、HTTPのRangeヘッダにフラグメント記述を設定して、サーバにGETリクエストを送信する。

```
GET /v1.ogb HTTP/1.1
Host: sample.org
Accept: video/*
Range: t:npt=10-20
    :
```

【０００６】
当該リクエストを受信したサーバは、リクエストヘッダのRangeの値を解析して、動画の一部分を示す記述「10-20」を抽出する。そして、下記のように、レスポンスヘッダContent-Range-Mappingに、送信するフラグメント動画の時間間隔と、送信する動画のバイトレンジを設定して、動画のバイナリデータをクライアント（Webブラウザ）に送信する。

```
HTTP/1.1 206 Partial Content
Content-Type: video/ogg
Content-Range-Mapping: {t:npt 10.5-20.5}={bytes 1234-2345/10000}
    :
(バイナリデータ)
```

【０００７】

また、複数のフラグメントURIを組み合わせて、複数のフラグメント動画から構成される集合動画を作成することができる。例えば、下記は、ある集合動画のHTML形式の記述例であり、名前空間http://sample.orgに属するvideos要素をコンテナとして集合動画を定義し、video要素を用いて各フラグメント動画を参照する。

【０００８】

```
<html>
  <head><title>sample</title></head>
  <body>
    <x:videos xmlns:x="http:/sample.org">
      <video src="http://sample.org/v1.ogv#t=10,20"/>
      <video src="http://sample.org/v2.ogv#t=100,200"/>
      <video src="http://sample.org/v1.ogv#t=50,60"/>
      <video src="http://sample.org/v1.ogv#t=60,70"/>
      <a href="http://sample-news.org">
        <video src="http://sample.org/v1.ogv#t=70,80"/>
      </a>
    </x:videos>
  </body>
</html>
```

図1によりフラグメント動画の再生順を示す。Webブラウザは各フラグメント動画をサーバからダウンロードし、図1に示すように、フラグメント動画を順番に再生することで一つの集合動画として再生する。

【０００９】

上記HTML形式の記述例は、同じ動画のリソースであっても、別々にフラグメント動画を記述する。つまり、動画v1.ogvに対してフラグメント記述#10,20、#50,60、#60,70が指定されている。従って、クライアントは、v1.ogvのフラグメント動画を取得するために、HTTPリクエストをサーバに三回送信し、サーバもレスポンスをクライアントに三回送信することになる。

【００１０】

また、サーバは、各リクエストを受信する度に、同じ動画を、そのコーデックに基づいて適切にデコードし、Rangeヘッダに指定されたフラグメント動画を抽出する必要がある。従って、集合動画に、同じ動画リソースを参照する、複数のフラグメント参照が多数含まれる場合、フラグメント動画の取得に時間がかかり、集合動画の再生においてタイムラグが発生する。

【００１１】

なお、フラグメントURIに関連する先行技術として、特許文献1は、フラグメントURIの記述を用いて、指定した矩形領域、時間の部分動画を取得する発明を開示する。また、特許文献2は、動画データのヘッダ部分に次に再生すべき動画の所在を記述し、一つ目の動画の再生終了とともに、予め取得した二つ目の動画を再生することで、スムーズに動画を再生する発明を開示する。

【先行技術文献】

【非特許文献】

【００１２】

【非特許文献１】http://www.w3.org/TR/media-frags/

【特許文献】
【００１３】
【特許文献１】特開2001-184243公報
【特許文献２】特開2007-295038公報
【発明の概要】
【発明が解決しようとする課題】
【００１４】
本発明は、接続可能なフラグメント記述を有する複数のユニフォームリソース識別子を修正した一つのユニフォームリソース識別子を生成して、動画を取得するためのリクエスト回数を低減することを目的とする。
【課題を解決するための手段】
【００１５】
本発明は、前記の目的を達成する一手段として、以下の構成を備える。
【００１６】
本発明にかかる情報処理は、マークアップ言語によって記述されたデータを入力し、前記データから、それぞれフラグメント記述を含む複数のユニフォームリソース識別子を取得し、前記複数のユニフォームリソース識別子が含むフラグメント記述に、接続可能なフラグメント記述があるか否かを判定し、前記接続可能なフラグメント記述があると判定された場合、前記接続可能なフラグメント記述を有するユニフォームリソース識別子を修正して、一つのユニフォームリソース識別子を生成し、前記生成されたユニフォームリソース識別子をサーバに送信することを特徴とする。
【発明の効果】
【００１７】
本発明によれば、接続可能なフラグメント記述を有する複数のユニフォームリソース識別子を修正した一つのユニフォームリソース識別子を生成して、動画を取得するためのリクエスト回数を低減することができる。
【図面の簡単な説明】
【００１８】
【図１】フラグメント動画の再生順を示す図。
【図２】実施例の情報処理システムの概要を説明するブロック図。
【図３】PCの構成例を説明するブロック図。
【図４】フラグメント動画を取得する処理例を説明するフローチャート。
【図５】videos要素VSiの解析を詳細に説明するフローチャート。
【図６】コンテンツテーブルの一例を説明する図。
【図７】フラグメント範囲のデータ型変換を詳細に説明するフローチャート。
【図８】コンテンツテーブルの更新を詳細に説明するフローチャート。
【図９】フラグメント動画の取得要求を詳細に説明するフローチャート。
【図１０】サーバがフラグメント動画の取得要求データを解析する処理を説明するフローチャート。
【図１１】サーバがフラグメント動画をクライアントに送信する処理を説明するフローチャート。
【図１２】コンテンツテーブルの更新の詳細を説明するフローチャート。
【図１３】実施例2のコンテンツテーブルの一例を説明する図。
【図１４】コンテンツテーブルの更新を詳細に説明するフローチャート。
【図１５】実施例3におけるHTMLデータの記述と空間フラグメントの関係を説明する図、および、クライアントのモニタ上に表示されるフラグメント動画を説明する図。
【図１６】コンテンツテーブルの一例を説明する図。
【図１７】コンテンツテーブルの更新を詳細に説明するフローチャート。
【図１８】動画の再生状態を説明する図。
【図１９】実施例4におけるコンテンツテーブルの更新を詳細に説明するフローチャート

。
【図２０】実施例4における非再生時間に関する閾値を動的に求める処理を説明するフローチャート。
【図２１】実施例5におけるコンテンツテーブルの更新を詳細に説明するフローチャート。
【図２２】コンテンツテーブルの一例を説明する図。
【図２３】図22(a)に示すフラグメント記述に従い動画を再生した様子を示す図。
【図２４】実施例6におけるコンテンツテーブルの更新を詳細に説明するフローチャート。
【発明を実施するための形態】
【００１９】
以下、本発明にかかる実施例の情報処理を図面を参照して詳細に説明する。
【実施例１】
【００２０】
実施例1においては、複数の異なる、フラグメント記述を含むユニフォームリソース識別子（フラグメントURI）を用いて同じ動画をフラグメント参照する場合に、より少ないリクエスト回数で動画を取得する方法を説明する。
【００２１】
［システムの構成］
図2のブロック図により実施例の情報処理システムの概要を説明する。例えばパーソナルコンピュータであるコンピュータ装置(PC)101は、ローカルエリアネットワーク(LAN)のようなネットワーク102に接続され、後述する情報処理を実行する。ネットワーク102には、ディジタルカメラ103、複合機(MFP)104、サーバ105なども接続されている。サーバ105は、クライアントからフラグメントURIに基づくリクエストを受信し、レスポンスデータとしてフラグメント動画を配信するメディアサーバとして機能する。
【００２２】
［装置の構成］
図3のブロック図によりPC101の構成例を説明する。CPU201は、RAM203をワークメモリとして、ROM202やハードディスクドライブ(HDD)204に格納されたオペレーティングシステム(OS)や各種のプログラムを実行し、システムバス211を介して後述する各構成を制御する。CPU201が実行するプログラムには、後述するフラグメント動画の取得に関するプログラムが含まれる。
【００２３】
CPU201は、ビデオカード(VC)206を介してモニタ212にユーザインタフェイス(UI)や動画を含む各種画像を表示する。PC101のオペレータは、キーボード、マウス、タッチパネルなどを有する操作部205を介してUIを操作し、各種指示やデータの入力を行う。
【００２４】
また、ネットワークインタフェイスカード(NIC)207は、PC101とネットワーク102を接続するネットワークインタフェイスである。汎用インタフェイス(I/F)209は、PC101と例えばUSBのようなシリアルバス210を接続するインタフェイスである。
【００２５】
［クライアントの処理］
図4のフローチャートによりフラグメント動画を取得する処理例を説明する。図4に示す処理は、オペレータによってフラグメント動画の取得が指示された場合に、CPU201によって実現される。
【００２６】
CPU201は、オペレータが指示するフラグメント動画に関するマークアップ言語（例えばHTML）のデータ（以下、HTMLデータ）を取得する(S101)。なお、当該HTMLデータのソースは、サーバ105、サーバ105以外のサーバ、HDD204、あるいは、その他のメモリである。そして、取得したHTMLデータを解析し(S102)、body要素の子孫要素として含まれるvideos要

素のリストを取得する(S103)。

【００２７】

次に、CPU201は、videos要素の数を変数Nに設定し(S104)、カウンタiを1に初期化して(S105)、詳細は後述するが、videos要素のリストのi番目のvideos要素VSiを解析し、フラグメント動画を取得する(S106)。そして、カウンタiをインクリメントし(S107)、カウンタiのカウント値と変数Nを比較してvideos要素のリストに含まれるすべてのvideos要素を解析したか否かを判定する(S108)。未解析のvideos要素があれば処理をステップS106に戻し、未解析のvideos要素がなければ処理を終了する。

【００２８】

●videos要素の解析

図5のフローチャートによりvideos要素VSiの解析(S106)を詳細に説明する。また、図6によりコンテンツテーブルの一例を説明する。

【００２９】

CPU201は、変数srcをリスト型に初期化し(S111)、videos要素VSiが含むvideo要素のsrc属性の属性値のURI、および、リンク情報をもつか否かを示すフラグを変数srcの末尾に追加する(S112)。そして、videos要素VSiが含むすべてのvideo要素を解析したか否かを判定し(S113)、未解析のvideo要素がなくなるまで、ステップS112、S113を繰り返す。なお、上述した「集合動画の記述例」における11行目のvideo要素＜video src="http://sample.org/v1.ogv#t=70,80"/＞は、祖先要素を辿ることによってa要素の子孫要素であることがわかる。

【００３０】

CPU201は、videos要素VSiが含むvideo要素の解析が終了すると、変数srcに格納した各URIのプロトコル、ホスト名、パス、フラグメント種別、フラグメント範囲を解析する(S114)。そして、解析結果に基づき、RAM203の所定領域にコンテンツテーブルを作成する(S115)。図6(a)は、上述した「集合動画の記述例」に対応するコンテンツテーブルの一例を示す。

【００３１】

次に、CPU201は、詳細は後述するが、フラグメント種別に基づき、対応するフラグメント範囲のデータ型を変換し(S116)、フラグメント情報に基づきコンテンツテーブルを更新する(S117)。そして、コンテンツテーブルに基づきフラグメント動画の取得要求データをサーバ105に送信し(S118)、サーバ105のレスポンスに基づきコンテンツテーブルを更新する(S119)。

【００３２】

●フラグメント範囲のデータ型変換

図7のフローチャートによりフラグメント範囲のデータ型変換(S116)を詳細に説明する。なお、図7に示す処理は、コンテンツテーブルの一行（1レコード）の処理に対応し、CPU201は、レコードの数分、図7に示す処理を実行する。

【００３３】

CPU201は、フラグメント種別を判定して(S301)、フラグメント種別が「t」（フラグメント範囲が時間を示す）場合、浮動小数点型（float型）の変数listを初期化する(S302)。そして、フラグメント範囲の文字列データを文字「,」で分割し(S303)、各文字列データをfloat型に変換して変数listに順に追加する(S304)。

【００３４】

また、CPU201は、フラグメント種別が空間（例えば「xywh」、フラグメント範囲が空間を示す）の場合、float型の変数listを初期化する(S305)。そして、フラグメント範囲の文字列データを文字「,」で分割し(S306)、各文字列データをfloat型に変換して変数listに順に追加する(S307)。

【００３５】

また、CPU201は、フラグメント種別が時間でも空間でもない場合、フラグメント種別が「トラック」であると見做し、文字列型（char型）の変数listを初期化する(S308)。そし

て、フラグメント範囲の文字列を文字「&」で分割し(S309)、各文字列を文字「=」で分割して、「=」より後の文字列を変数listに順に追加する(S310)。

【００３６】

図6(a)に示すコンテンツテーブルの場合、フラグメント種別はすべて「t」であるから、フラグメント範囲はすべてfloat型のデータに変換される。

【００３７】

●コンテンツテーブルの更新(S117)

図8のフローチャートによりコンテンツテーブルの更新(S117)を詳細に説明する。

【００３８】

CPU201は、コンテンツテーブルのレコード数を変数Mに設定し(S401)、カウンタjを1に初期化する(S402)。そして、カウンタjのカウント値と変数Mを比較して(S403)、j＜Mであれば処理をステップS404に進め、j=Mであれば処理を終了する。

【００３９】

CPU201は、j＜Mの場合、コンテンツテーブルのj行目（レコードj）と、レコードjに連続するj+1行目（レコードj+1）のリンクフラグを判定し(S404)、それらのリンクフラグが肯定(true)の場合は処理をステップS411に進める。

【００４０】

CPU201は、レコードj、j+1のリンクフラグが否定(false)の場合、レコードjとレコードj+1について参照先リソースが同一か否かの判定(S405)、フラグメント種別が「t」か否かの判定(S406)を行う。さらに、レコードjのフラグメント範囲の終了点とレコードj+1のフラグメント範囲の開始点が同一か否かの判定(S407)を行う。なお、参照先リソースはプロトコル、ホスト名、パスの集合によって表される。そして、参照先リソースが異なる、フラグメント種別が「t」以外、フラグメント範囲の終了点と開始点が異なる場合は処理をステップS411に進める。

【００４１】

CPU201は、レコードjとj+1における参照先リソースが同一、フラグメント種別が「t」、フラグメント範囲の終了点と開始点が同一の場合、レコードjのフラグメント範囲の終了点とレコードj+1のフラグメント範囲の開始点は接続可能である。そこで、連続するレコードの一方であるレコードjのフラグメント範囲の終了点を、連続するレコードの他方であるレコードj+1のフラグメント範囲の終了点に変更する(S408)。そして、コンテンツテーブルからレコードj+1を削除し(S409)、変数Mをデクリメントする(S410)。

【００４２】

次に、CPU201は、カウンタjをインクリメントして(S411)、処理をステップS403に戻す。

【００４３】

つまり、参照先のリソースが同一、フラグメント種別が「t」、フラグメント範囲の終了点と開始点が同一の場合、レコードj+1のフラグメント範囲をレコードjのフラグメント範囲に含め、コンテンツテーブルからレコードを一つ削除する。

【００４４】

例えば、図6(a)に示すコンテンツテーブルにおいて、レコード3と4において、リンクフラグは否定であり、参照先リソースが同一、フラグメント種別が「t」、フラグメント範囲の終了点と開始点が同一である。従って、上記の処理により、図6(b)に示すように、レコード3のフラグメント範囲が[50,60]から[50,70]に変更され、レコード4が削除される。また、図6(b)におけるレコード3と4において、参照先リソースが同一、フラグメント種別が「t」、フラグメント範囲の終了点と開始点が同一である。しかし、レコード4のリンクフラグが肯定（リンク情報を有する）であるため、それらのフラグメント範囲は統合されない。

【００４５】

●フラグメント動画の取得要求

図9のフローチャートによりフラグメント動画の取得要求(S118)を詳細に説明する。

【００４６】
CPU201は、コンテンツテーブルのレコード数を変数Mに設定し(S501)、カウンタiを1に初期化し(S502)、各レコードの参照先リソースに対応する送信済みフラグを否定に初期化する(S503)。そして、カウンタiのカウント値と変数Mを比較して(S504)、i＜Mであれば処理をステップS505に進め、i=Mであれば処理を終了する。
【００４７】
CPU201は、i＜Mの場合、変数indexをリスト型に初期化し(S505)、否定を示す送信済みフラグの数を変数jに設定し(S506)、レコードjのフラグメント範囲を変数indexに追加する(S507)。そして、変数kにj-1を設定し(S508)、変数kを判定して(S509)、k=0であれば処理をステップS515に進める。
【００４８】
k＞0の場合、CPU201は、レコードkに対応する送信済みフラグの判定(S510)、レコードkとレコードjの参照先リソースの判定(S511)を行う。レコードkに対応する送信済みフラグが肯定、または、レコードkとレコードjの参照先リソースが異なる場合は、処理をステップS514に進める。
【００４９】
レコードkに対応する送信済みフラグが否定、かつ、レコードkとレコードjの参照先リソースが同一の場合、CPU201は、レコードkのフラグメント範囲を変数indexに追加する(S512)。そして、レコードkに対応する送信済みフラグを反転し（肯定にして）(S513)、変数kをデクリメントし(S514)、処理をステップS509に戻す。
【００５０】
k=0の場合、CPU201は、変数indexに格納した各レコードのフラグメント範囲に基づき、フラグメント動画の取得要求データをサーバ105に送信する(S515)。そして、カウンタiをインクリメントし(S516)、処理をステップS504に戻す。
【００５１】
図6(b)に示すコンテンツテーブルの場合、動画v1.ogvのフラグメント動画の取得要求データの内容は下記のようになる。つまり、Rangeヘッダを用いて、動画v1.ogvのフラグメント範囲として10-20秒、50-70秒、70-80秒を指定する。

```
GET /v1.ogv HTTP/1.1
Host: sample.org
Accept: video/*
Range: t:npt=10-20,50-70,70-80
    :
```

【００５２】
［サーバの処理］
●フラグメント動画の取得要求データの解析
図10のフローチャートによりサーバ105がフラグメント動画の取得要求データを解析する処理を説明する。なお、図10に示す処理はサーバ105のCPUが行う処理であるが、以下では、サーバ105が行うとして記述する。
【００５３】
サーバ105は、フラグメント動画の取得要求データを受信すると、HTTPのリクエストメソッドの解析(S601)、パス情報の解析(S602)、Rangeヘッダの解析(S603)、フラグメント種別の判定(S604)を行う。
【００５４】
フラグメント種別が時間に関する場合、サーバ105は、変数listをfloat型のリストに初期化する(S605)。そして、フラグメント範囲の文字列データを文字「-」で分割し(S606)、各文字列データをfloat型に変換して変数listに順に追加する(S607)。
【００５５】
また、フラグメント種別が空間に関する場合、サーバ105は、変数listをfloat型のリストに初期化する(S608)。そして、フラグメント範囲の文字列データを文字「,」で分割し(

S609)、各文字列データをfloat型に変換して変数listに順に追加する(S610)。

【００５６】

また、フラグメント種別が時間でも空間でもない場合、サーバ105は、フラグメント種別が「トラック」であると見做し、変数listを文字列型のリストに初期化する(S611)。そして、フラグメント範囲の文字列データを文字「&」で分割し(S612)、各文字列データを変数listに順に追加する(S613)。

【００５７】

以上の処理により、サーバ105は、クライアントが要求するフラグメント動画を特定することができる。

【００５８】

●フラグメント動画のクライアントへの送信処理

図11のフローチャートによりサーバ105がフラグメント動画をクライアントに送信する処理を説明する。

【００５９】

サーバ105は、フラグメント動画の取得要求データから抽出したフラグメント種別、フラグメント範囲、バイトレンジ、メディアタイプ、フラグメント動画の入力ストリームを集合とするリストをリスト型の変数mappingに格納する(S701)。そして、マルチパート送信用の識別子を生成し、レスポンスヘッダにContent-Typeを設定する(S702)。本実施例において、マルチパート送信用の識別子を「BOUNDARY」と仮定し、Content-Typeヘッダの値は「multipart/byteranges; boundary=BOUNDARY」になる。

【００６０】

次に、サーバ105は、レスポンスヘッダにContent-Range-Mappingを設定し(S703)、変数mappingのリストサイズを変数Mに設定し(S704)、カウンタjを1に初期化する(S705)。そして、変数mappingの、カウンタjに対応するリストに含まれるフラグメント種別、フラグメント範囲、バイトレンジをContent-Range-Mappingヘッダの値として追加する(S706)。

【００６１】

次に、サーバ105は、カウンタjをインクリメントし(S707)、カウンタjのカウント値と変数Mを比較し(S708)、j≦Mの場合は処理をステップS706に戻す。つまり、ステップS706の繰り返しによって、変数mappingに格納された全フラグメント情報がContent-Range-Mappingヘッダに設定される。

【００６２】

j＞Mになると、サーバ105は、カウンタjを1に初期化し(S709)、レスポンスボディの出力ストリームを初期化する(S710)。そして、文字列「--」、マルチパートの識別子「BOUNDARY」を出力ストリームに設定する(S711)。さらに、「Content-Type」ヘッダとともにカウンタjに対応する動画データのメディアタイプを出力ストリームに設定し(S712)、続けて、動画データのバイナリデータを出力ストリームに設定する(S713)。つまり、ステップS712、S713の繰り返しによって、変数mappingに格納された全フラグメント動画が出力ストリームに設定される。

【００６３】

次に、サーバ105は、カウンタjをインクリメントし(S714)、カウンタjのカウント値と変数Mを比較し(S715)、j≦Mの場合は処理をステップS711に戻す。そして、j＞Mになると、マルチパートの終了識別子として文字列「--」、識別子「BOUDNARY」、文字列「--」を出力ストリームに設定して(S716)、出力ストリームを閉じる(S717)。

【００６４】

上述した動画v1.ogvのフラグメント動画の取得要求データに対するサーバのレスポンスデータの一例を下に示す。

```
HTTP/1.1 206 Partial Content
Content-Type: multipart/byteranges; boundary=BOUNDARY
Content-Range-Mapping:
{t:npt 10-20}={bytes 1000-2000/10000}
```

```
{t:npt 50-70}={bytes 5000-7000/10000}
{t:npt 70-80}={bytes 7000-8000/10000}
--BOUNDARY
Content-Type: video/ogg
Content-Range: 1000-2000/10000
(バイナリデータ)
--BOUNDARY
Content-Type: video/ogg
Content-Range: 5000-7000/10000
(バイナリデータ)
--BOUNDARY
Content-Type: video/ogg
Content-Range: 7000-8000/10000
(バイナリデータ)
--BOUNDARY--
```

【００６５】

［クライアントの処理（つづき）］

●コンテンツテーブルの更新(S119)

CPU201は、上記のサーバレスポンスを解析し、コンテンツテーブルを更新する(S119)。図12のフローチャートによりコンテンツテーブルの更新(S119)の詳細を説明する。

【００６６】

CPU201は、サーバレスポンスのContent-Range-Mappingヘッダを解析し(S801)、各行のフラグメント種別、フラグメント範囲、バイトレンジを集合とするリストをリスト型の変数rangeに格納する(S802)。そして、変数indexに格納したフラグメント範囲と変数rangeのリストに基づき、図6(b)に示すコンテンツテーブルにバイトレンジ情報を追加する(S803)。さらに、変数indexに格納したフラグメント範囲に基づき、フラグメント範囲にレスポンスボディのストリームデータを関連付ける(S804)。

【００６７】

図6(c)は、サーバレスポンスに基づきコンテンツテーブル（図6(b)）を更新(S119)した後のコンテンツテーブルを示す。つまり、CPU201は、動画v1.ogvのフラグメント範囲[10,20][50,70][70,80]に対してそれぞれバイトレンジ1000-2000/10000、5000-7000/10000、7000-8000/10000を関連付ける。また、ストリームインスタンスとして、レスポンスボディのストリームを示すstream1を関連付ける。

【００６８】

なお、別の参照先リソースに対応する動画v2.ogvのフラグメント動画の取得要求(S515)、サーバ105の処理について、上記では説明を省略した。動画v2.ogvのフラグメント動画の取得要求データは下記のようになる。

```
GET /v2.ogv HTTP/1.1
Host: sample.org
Accept: video/*
Range: t:npt=100-200
    :
```

【００６９】

また、サーバのレスポンスデータは下記のようになる。

```
HTTP/1.1 206 Partial Content
Content-Type: video/ogg
Content-Range-Mapping: {t:npt 100-200}={bytes 10000-20000/100000}
    :
(バイナリデータ)
```

【００７０】

CPU201は、動画v1.ogvと同様に、動画v2.ogvのフラグメント範囲[100,200]に対してバイトレンジ10000-20000/100000を関連付ける。さらに、ストリームインスタンスとして、レスポンスボディのストリームを示すstream2を関連付ける。

【００７１】

CPU201は、図6(c)に示すコンテンツテーブルに基づき、各ストリームインスタンスから動画データを取得し、モニタ212に表示した例えばウェブブラウザのウィンドウ内において、それら動画データを順番に表示することで、集合動画を再生する。

【００７２】

図6の例では低減される、動画を取得するためのリクエスト回数は一回であるが、取得すべきフラグメント動画の構成によっては、さらにリクエスト回数を低減することができる。つまり、クライアントからサーバにフラグメント動画を要求する際に、フラグメント動画のリクエスト回数を減らすことができる。その結果、クライアントーサーバ間のリクエスト/レスポンスの回数を減らして、集合動画の再生におけるタイムラグの発生を防ぎ、ビデオストリーミングのパフォーマンスを向上させることができる。

【実施例２】

【００７３】

以下、本発明にかかる実施例2の情報処理を説明する。なお、実施例2において、実施例1と略同様の構成については、同一符号を付して、その詳細説明を省略する。

【００７４】

取得したフラグメント動画に関するHTMLデータが下記のような記述だったとする。つまり、コンテナであるvideos要素の子要素に関して、三番目と四番目のvideo要素は同一の参照先リソースを示し、それらのフラグメント範囲には非再生時間の五秒間が存在する。実施例1において説明した処理によれば、video要素のフラグメント範囲の終了点と、続くvideo要素のフラグメント範囲の開始点が一致すれば、それらvideo要素は統合される。しかし、下記のHTMLデータにおいては非再生時間の五秒間によって、三番目と四番目のvideo要素は統合されない。

【００７５】

```
<html>
  <head><title>sample</title></head>
  <body>
    <x:videos xmlns:x="http:/sample.org">
      <video src="http://sample.org/v1.ogv#t=10,20"/>
      <video src="http://sample.org/v2.ogv#t=100,200"/>
      <video src="http://sample.org/v1.ogv#t=50,60"/>
      <video src="http://sample.org/v1.ogv#t=65,70"/>
    </x:videos>
  </body>
</html>
```

そこで、実施例2においては、非再生時間が閾値以下の場合は、それらvideo要素を統合する例を説明する。なお、非再生時間に関する閾値は、固定でもよいし、ユーザが設定することもできる。

【００７６】

図13により実施例2におけるコンテンツテーブルの一例を説明するが、図13は上記の「集合動画の記述例」に対応するコンテンツテーブルの一例である。

【００７７】

図14のフローチャートによりコンテンツテーブルの更新(S117)を詳細に説明する。なお、図8に示す実施例1と同様の処理には同一符号を付して、その詳細説明を省略する。

【００７８】

CPU201は、非再生時間の閾値を変数Dに設定し(S901)、実施例1と同様のステップS401からS406の処理を行う。レコードjとj+1について、リンクフラグが否定、参照先リソースが

同一、フラグメント種別が「t」の場合、レコードjのフラグメント範囲の終了点$T_j e$とレコードj+1のフラグメント範囲の開始点$T_{j+1} b$の差分を計算する(S902)。なお、差分は非再生時間Tdに相当する。そして、非再生時間Tdと変数Dを比較して(S903)、Td＞Dならば処理をステップS411に進める（レコードjとj+1を統合しない）。また、Td≦Dならば、レコードjのフラグメント範囲の終了点とレコードj+1のフラグメント範囲の開始点は接続可能であるとして、処理をステップS408に進める（レコードjとj+1を統合する）。

【００７９】

非再生時間の閾値を五秒間として、図14に示す処理を行えば、上記のHTMLデータにおける三番目と四番目のvideo要素は統合されることになり、動画を取得するためのリクエスト回数が低減される。つまり、参照先リソースが同一の複数のフラグメント動画が短い非再生間隔で細切れ状態に存在する場合、非再生時間の閾値を適切に設定することで、それら複数のフラグメント動画を統合して、リクエスト回数を低減することができる。

【実施例３】

【００８０】

以下、本発明にかかる実施例3の情報処理を説明する。なお、実施例3において、実施例1、2と略同様の構成については、同一符号を付して、その詳細説明を省略する。

【００８１】

実施例3においては、空間に関するフラグメント（以下、空間フラグメント）が指定された複数の動画について、それら空間を包含する取得要求データをサーバ105に送信することで、より少ないリクエスト回数で動画を取得する例を説明する。

【００８２】

空間フラグメントが指定された動画を含むHTMLデータの一例を下に示す。図15(a)により実施例3におけるHTMLデータの記述と空間フラグメントの関係を説明する。

【００８３】

```
<html>
  <head>sample</head>
  <body>
    <video src="http://sample.org/v1.ogv#xywh=160,120,320,240"/>
    <video src="http://sample.org/v1.ogv#xywh=170,360,320,240"/>
  </body>
</html>
```

記述#xywh=x,y,w,hは、画像の左上を原点として、座標(x, y)からX方向に幅w、Y方向に高さhの矩形画像を切り出すことを意味する。従って、このようなURIに基づくリクエストを受信したサーバは、図15(a)に示すように、動画v1.ogvから、左上座標が(x, y)、右下座標が(x+w, y+h)の矩形領域の動画を切り出してクライアントに送信する。

【００８４】

図15(b)によりクライアントのモニタ上に表示されるフラグメント動画を説明する。つまり、クライアントのモニタに表示されたウェブブラウザのウィンドウ内に各video要素に指定されたフラグメント動画が表示される。

【００８５】

図16によりコンテンツテーブルの一例を説明するが、図16(a)は上記の「集合動画の記述例」に対応するコンテンツテーブルの一例である。

【００８６】

図17のフローチャートによりコンテンツテーブルの更新(S117)を詳細に説明する。なお、図8に示す実施例1と同様の処理には同一符号を付して、その詳細説明を省略する場合がある。

【００８７】

CPU201は、コンテンツテーブルのレコード数を変数Mに設定し(S401)、カウンタjを1に初期化する(S402)。そして、カウンタjのカウント値と変数Mを比較して(S403)、j＜Mであれば処理をステップS404に進め、j=Mであれば処理を終了する。

【００８８】
CPU201は、j＜Mの場合、コンテンツテーブルのレコードjとレコードj+1について参照先リソースが同一か否かの判定(S405)、フラグメント種別が「xywh」か否かの判定(S911)を行う。そして、参照先リソースが異なる、フラグメント種別が「xywh」以外の場合は処理をステップS411に進める。
【００８９】
CPU201は、レコードjとj+1における参照先リソースが同一、および、フラグメント種別が「xywh」の場合、次の処理（ステップS912からS920）を行う。

if ($X_{j+1} < X_j$){； …(S912)
    $X = X_{j+1}$；
    $W = X_j + W_j - X_{j+1}$； …(S913)
} else {
    $X = X_j$；
    $W = X_{j+1} + W_{j+1} - X_j$； …(S914)
}
if ($Y_{j+1} < Y_j$){； …(S915)
    $Y = Y_{j+1}$；
    $H = Y_j + H_j - Y_{j+1}$； …(S916)
} else {
    $Y = Y_j$；
    $H = Y_{j+1} + H_{j+1} - Y_j$； …(S917)
}

ここで、X_jはレコードjのフラグメント範囲のX座標、
W_jはレコードjのフラグメント範囲の幅、
H_jはレコードjのフラグメント範囲の高さ、
X_{j+1}はレコードj+1のフラグメント範囲のX座標、
W_{j+1}はレコードj+1のフラグメント範囲の幅、
H_{j+1}はレコードj+1のフラグメント範囲の高さ。

【００９０】
そして、CPU201は、レコードjのフラグメント範囲を変数X、Y、W、Hそれぞれの値で書き換え(S918)、コンテンツテーブルからレコードj+1を削除し(S919)、変数Mをデクリメントする(S920)。
【００９１】
次に、CPU201は、カウンタjをインクリメントして(S411)、処理をステップS403に戻す。
【００９２】
つまり、参照先のリソースが同一、フラグメント種別が「xywh」の場合、レコードj+1のフラグメント範囲をレコードjのフラグメント範囲に含め、コンテンツテーブルからレコードを一つ削除する。
【００９３】
図16(a)に示すコンテンツテーブルに上記の処理を行うと、レコード1の範囲とレコード2のフラグメント範囲が統合されて、図16(b)に示すコンテンツテーブルに変換される。そして、下記のフラグメント動画の取得要求データがサーバ105に送信される。

```
GET /v1.ogv HTTP/1.1
Host: sample.org
Accept: video/*
Range: xywh=160,120,330,480
    :
```

【００９４】
上記のフラグメント動画の取得要求データに対するサーバのレスポンスデータの一例を

下に示す。

```
HTTP/1.1 206 Partial Content
Content-Type: multipart/byteranges; boundary=BOUNDARY
Content-Range: bytes 1000-2000/10000
Content-Range-Mapping: xywh=160,120,330,480
    :
(バイナリデータ)
```

【００９５】

CPU201は、上記のサーバレスポンスを解析し、コンテンツテーブルを更新する(S119)。図16(c)は、サーバレスポンスに基づきコンテンツテーブル（図16(b)）を更新(S119)した後のコンテンツテーブルを示す。つまり、CPU201は、動画v1.ogvのフラグメント範囲[160,120,330,480]に対してバイトレンジ1000-2000/10000を関連付ける。さらに、ストリームインスタンスとして、レスポンスボディのストリームを示すstream1を関連付ける。

【００９６】

図18により動画の再生状態を説明する。CPU201は、図16(c)に示すコンテンツテーブルに基づき、ストリームインスタンスから動画データを取得する。そして、モニタ212に表示した例えばウェブブラウザのウィンドウ内のサイズ330×480に対応する領域に動画データを表示することで、動画を再生する。

【００９７】

このように、空間フラグメントが指定された複数の動画について、それら空間を包含する取得要求データをサーバ105に送信することで、より少ないリクエスト回数で動画を取得することができる。

【実施例４】

【００９８】

以下、本発明にかかる実施例4の情報処理を説明する。なお、実施例4において、実施例1、2と略同様の構成については、同一符号を付して、その詳細説明を省略する。

【００９９】

実施例2では、非再生時間に関する閾値を固定またはユーザが設定する例を説明した。実施例4では、非再生時間に関する閾値をフラグメント動画の内容に基づいて決定する例を説明する。

【０１００】

図19のフローチャートにより実施例4におけるコンテンツテーブルを更新する手順を詳細に説明する。なお、図8に示す実施例1と同様の処理、図14に示す実施例2と同様の処理には同一符号を付して、その詳細説明を省略する。

【０１０１】

CPU201は、コンテンツテーブルの各リソースに関して、非再生時間に関する閾値を示すマップ(Map)を作成する(S1001)。図20のフローチャートによりマップの作成(S1001)の処理を詳細に説明する。

【０１０２】

CPU201は、コンテンツテーブルのレコード数を変数Lに設定する(S1101)。そして、コンテンツテーブルにおけるリソースのホスト名とパスを組み合わせた文字列をキーとし、そのリソースの時間を値とするMapを初期化する(S1102)。なお、マップは、一意の重複しない値であるキーに対して、その組となるように値を管理するテーブルである。マップに対してキーを入力すると、その組となる値を取り出すことができる。また、マップに対してキーと値を入力すると、マップとして管理されているテーブルを、そのキーと値で更新することができる。

【０１０３】

次に、CPU201は、カウンタjを1に初期化し(S1103)、カウンタjの値が変数L以下か否かを判定する(S1104)。カウンタjの値が変数L以下（j≦L）の場合、j行目のリンクフラグを判定し(S1105)、j行目のフラグメント種別が時間「t」か否かを判定する(S1106)。

【０１０４】
j行目のリンクフラグが否定を示し、かつ、j行目のフラグメント種別が時間「t」の場合、CPU201は、Mapからj行目のリソースに関する時間Tjを取得する(S1107)。そして、時間Tjに対して、j行目のリソースのフラグメント時間を加算する(S1108)。そして、j行目のリソースと時間Tjの組を用いてMapを更新し(S1109)、カウンタjをインクリメントし(S1110)、処理をステップS1104に戻す。
【０１０５】
また、j行目のリンクフラグが肯定、または、j行目のフラグメント種別が時間「t」以外の場合、CPU201は、カウンタjをインクリメントし(S1110)、処理をステップS1104に戻す。
【０１０６】
ステップS1104において、カウンタjの値が変数L以下ではない（j＞L）場合、CPU201は、Mapの各リソースに関連付けられた時間Tjに、0以上1以下の重み係数を乗算して、Mapの内容を更新する(S1111)。
【０１０７】
図20に示す以上の処理が終了すると、コンテンツテーブルの各リソースに関する非再生時間に関する閾値を示すMapの作成が完了する。次に、処理は図19のフローに戻るが、ステップS401からS902の処理は実施例2と同様であり、その詳細説明を省略する。
【０１０８】
ステップS902の処理が終了すると、CPU201は、参照先リソースのホスト名とパスを組み合わせた文字列をキーとして、Mapからそのリソースの閾値を取得し、変数Dに設定する(S1002)。そして、設定した変数Dと非再生時間Tdを比較する(S903)。そして、実施例1、2と同様に、非再生時間Tdが変数D以下（Td≦D）の場合はステップS408からS410の処理を実行してレコードjとj+1を結合し、Td＞Dの場合はレコードjとj+1を結合しない。
【０１０９】
なお、非再生時間に関する閾値である変数Dは、テーブルに基づいて、動画の切り出し時間の総和に対して重み係数を掛け合わせた値になる。
【０１１０】
以上のように、非再生時間に関する閾値をフラグメント動画の内容に基づいて決定することにより、フラグメント動画の見栄えに大きな影響がない場合に、リクエスト回数を低減することができる。
【実施例５】
【０１１１】
以下、本発明にかかる実施例5の情報処理を説明する。なお、実施例5において、実施例1-3と同様の構成については、同一符号を付して、その詳細説明を省略する。
【０１１２】
実施例3においては、空間フラグメントが指定された複数の動画について、それらの空間を包含する取得要求データをサーバに送信し、リクエスト回数を低減した。実施例5では、元のフラグメント動画の空間領域（図15(a)）と、統合を行った場合のフラグメント動画の空間領域（図18）との違いが、所定の範囲内（フラグメント動画の見栄えに大きな影響がない）の場合のみ、リクエスト回数を低減する例を説明する。
【０１１３】
言い替えれば、実施例3において、複数のフラグメント領域をまとめて取得する際に、オリジナルのフラグメント動画の見栄えに対して、再生時の見栄えが大きく変わってしまう場合がある。その場合、実施例5では、その複数のフラグメント領域をまとめて取得しないことにより、見栄えを保持する。
【０１１４】
図21のフローチャートにより実施例5におけるコンテンツテーブルの更新(S117)を詳細に説明する。なお、図17に示す実施例3と同様の処理には同一符号を付して、その詳細説明を省略する。

【０１１５】
CPU201は、ステップS401-S405、S911-S917の処理を行った後、S1=Wj×Hj+Wj+1×Hj+1とS2=W×Hを計算し(S1201)、値S1と値k・S2を比較する(S1202)。なお、kは値が0以上1以下の重み係数である。そして、S1＞k・S2の場合は実施例3と同様にステップS918からS920の処理を行って、レコードjのフラグメント範囲を変数X、Y、W、Hそれぞれの値で書き換える。他方、S1≦k・S2の場合はレコードjのフラグメント領域を更新せずに、処理をステップS411に進める。
【０１１６】
つまり、元（統合前）のフラグメント動画の空間領域と、統合を行った場合（統合後）のフラグメント動画の空間領域の違いが、所定の範囲内の場合のみ、リクエスト回数を低減することが可能になる。
【実施例６】
【０１１７】
以下、本発明にかかる実施例6の情報処理を説明する。なお、実施例6において、実施例1-5と同様の構成については、同一符号を付して、その詳細説明を省略する。
【０１１８】
実施例1は時間のフラグメントが指定された複数の動画を対象とし、実施例3は空間のフラグメントが指定された複数の動画を対象とした。実施例6では、フラグメント動画が時間のフラグメント記述と空間のフラグメント記述を併用したURIを用いる場合に、より少ないリクエスト回数で動画を取得する方法を説明する。
【０１１９】
図22(a)によりフラグメント動画に含まれる各動画リソースに関するコンテンツテーブルの一例を示す。図22(a)に示すフラグメント記述においては、時間のフラグメント記述と空間のフラグメント記述が併用されている。
【０１２０】
図23により図22(a)に示すフラグメント記述に従い動画を再生した様子を示す。つまり、時間のフラグメント記述と、空間のフラグメント記述に基づいた部分動画から構成された動画が再生される。なお、図23に太線で示す矩形領域は、オリジナルの動画リソースに対する、抽出された部分動画の占める位置とその割合を示す。図22(a)におけるレコード2の「v2.ogv」とレコード5の「v1.ogv」は空間のフラグメント記述を含まず、オリジナルの動画リソースの全空間を参照する。
【０１２１】
図22(a)に示すコンテンツテーブルは図24に示す処理によって更新される。図24により実施例6におけるコンテンツテーブルの更新(S117)を詳細に説明する。
【０１２２】
CPU201は、実施例1と同様にステップS401-S407の処理を行う。ただし、実施例6においてステップS406の処理はない。
【０１２３】
レコードj、j+1について、リンクフラグが否定、参照先リソースが同一、フラグメント範囲の終了点と開始点が同一の条件を満たす場合、CPU201は、レコードｊとレコードj+1が空間フラグメントをもつか否か判定する(S1301)。そして、レコードｊとレコードj+1が空間フラグメントをもたない場合は処理をステップS408に進める。
【０１２４】
また、レコードｊとレコードj+1が空間フラグメントをもつ場合、CPU201は、実施例3と同様にステップS912-S918の処理を行う。そして、レコードjのフラグメント範囲を変数X、Y、W、Hそれぞれの値で書き換え、処理をステップS408に進める。
【０１２５】
そして、CPU201は、実施例1と同様に、連続するレコードの一方であるレコードjのフラグメント範囲の終了点を、連続するレコードの他方であるレコードj+1のフラグメント範囲の終了点に変更する(S408)。そして、コンテンツテーブルからレコードj+1を削除する(

S409)。

【０１２６】

図22(a)に示すコンテンツテーブルに、図24の処理を施すと、図22(b)に示す更新されたコンテンツテーブルが得られる。つまり、図22(a)におけるレコード3と4は、図22(b)におけるレコード3に統合される。また、図22(a)におけるレコード1と3は、空間的に隣接するが、時間的に隣接しないので、一つのレコードには統合されない。また、レコード4と5は、時間的に隣接するが、空間の統合を行うとオリジナルのフラグメント動画に対して見栄えが変わるので統合しない。

【０１２７】

そして、図22(b)に示すコンテンツテーブルに対して、実施例1と同様に、図9のフローチャートに示す処理によりコンテンツテーブルを更新する。更新されたコンテンツテーブルは図22(c)に示すようになる。

【０１２８】

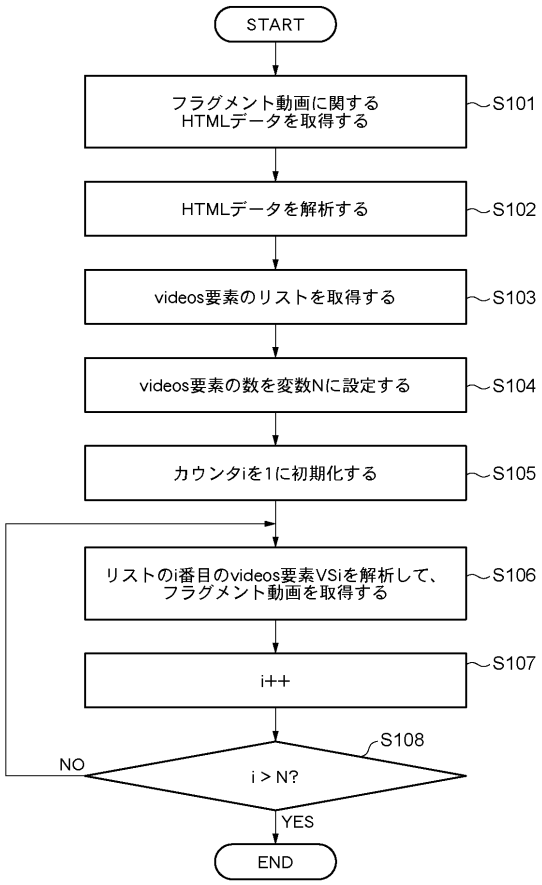
CPU201は、図22(c)のコンテンツテーブルを基に、各ストリームインスタンスから動画データを取得し、動画データを順番に表示することで、集合動画を再生する。

【０１２９】

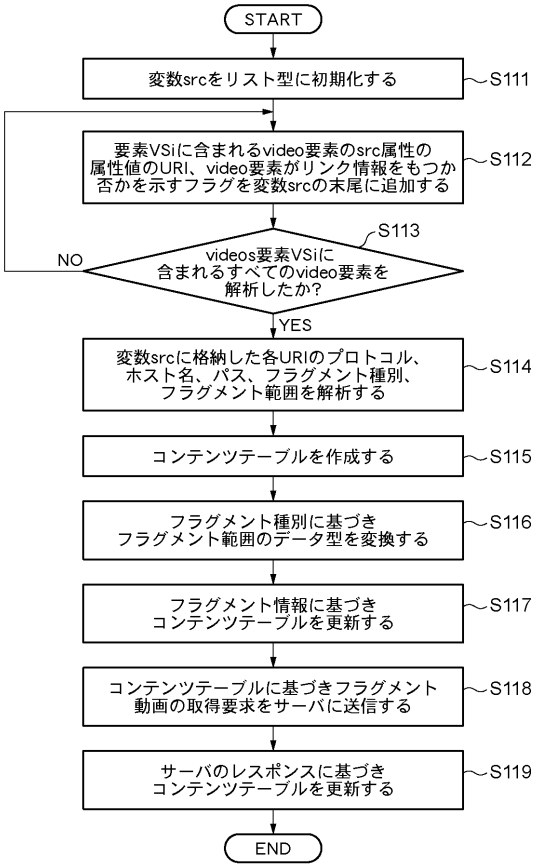
［その他の実施例］

また、本発明は、以下の処理を実行することによっても実現される。即ち、上述した実施形態の機能を実現するソフトウェア（プログラム）を、ネットワーク又は各種記憶媒体を介してシステム或いは装置に供給し、そのシステムあるいは装置のコンピュータ（又はCPUやMPU等）がプログラムを読み出して実行する処理である。

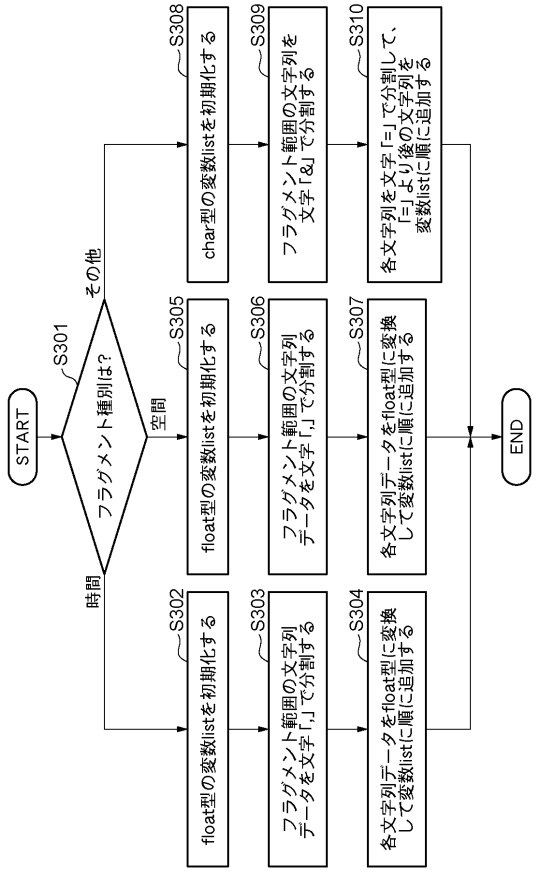
【図４】



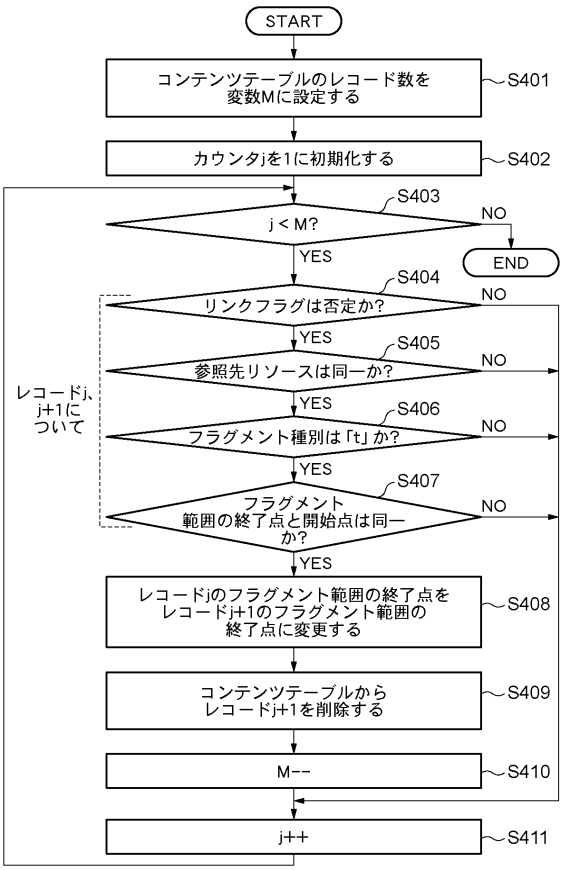
【図５】



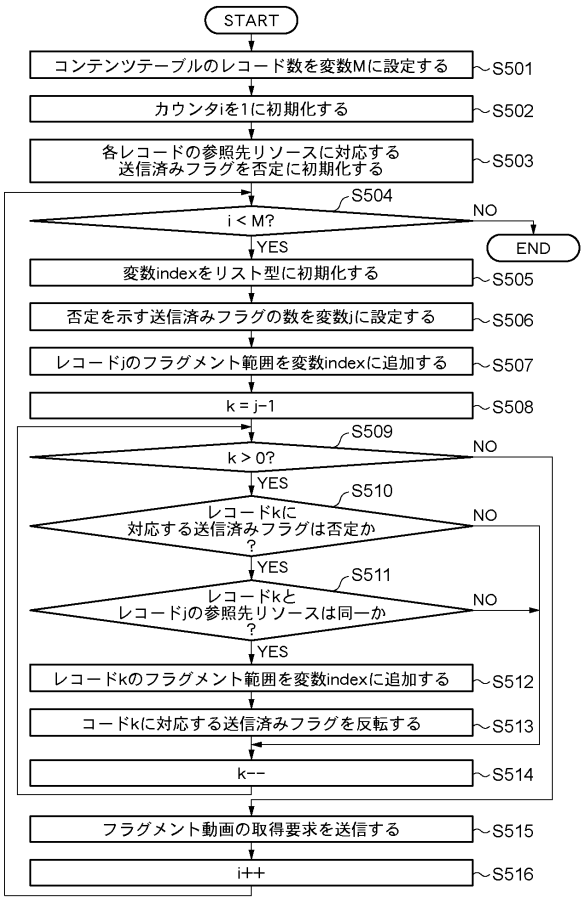
【図７】


START
フラグメント種別は？
その他
空間
時間
S301
S302
float型の変数listを初期化する
S303
フラグメント範囲の文字列データを文字「,」で分割する
S304
各文字列データをfloat型に変換して変数listに順に追加する
S305
float型の変数listを初期化する
S306
フラグメント範囲の文字列データを文字「,」で分割する
S307
各文字列データをfloat型に変換して変数listに順に追加する
S308
char型の変数listを初期化する
S309
フラグメント範囲の文字列を文字「&」で分割する
S310
各文字列を文字「=」で分割して、「=」より後の文字列を変数listに順に追加する
END

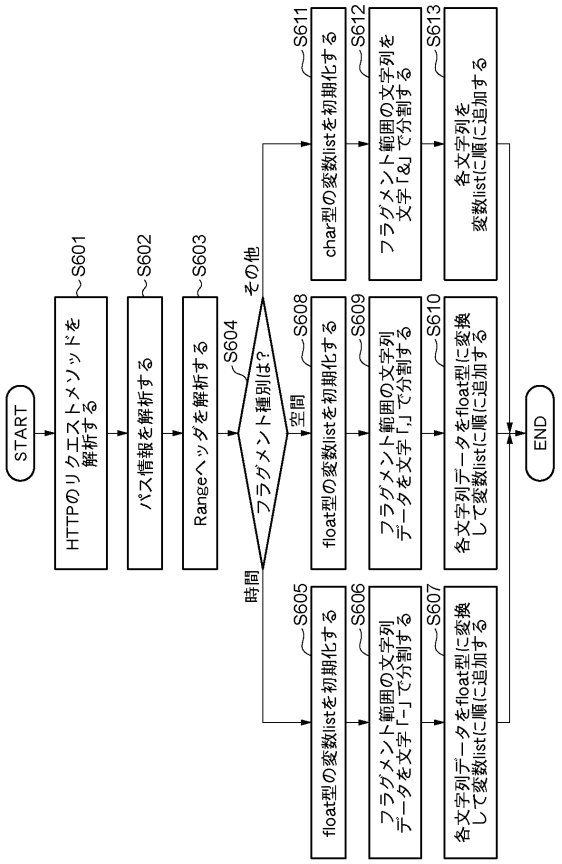
【図８】


START
コンテンツテーブルのレコード数を変数Mに設定する
S401
カウンタjを1に初期化する
S402
j < M?
S403
NO
YES
END
リンクフラグは否定か?
S404
参照先リソースは同一か?
S405
フラグメント種別は「t」か?
S406
フラグメント範囲の終了点と開始点は同一か?
S407
レコードjのフラグメント範囲の終了点をレコードj+1のフラグメント範囲の終了点に変更する
S408
コンテンツテーブルからレコードj+1を削除する
S409
M--
S410
j++
S411
レコードj+1について

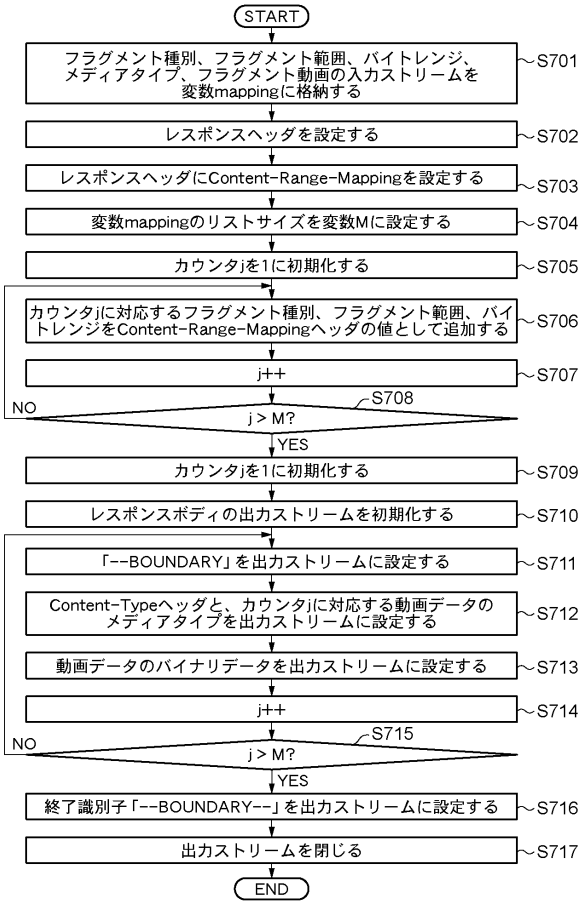
【図９】


START
コンテンツテーブルのレコード数を変数Mに設定する
S501
カウンタiを1に初期化する
S502
各レコードの参照先リソースに対応する送信済みフラグを否定に初期化する
S503
i < M?
S504
NO
YES
END
変数indexをリスト型に初期化する
S505
否定を示す送信済みフラグの数を変数jに設定する
S506
レコードjのフラグメント範囲を変数indexに追加する
S507
k = j-1
S508
k > 0?
S509
レコードkに対応する送信済みフラグは否定か?
S510
レコードkとレコードjの参照先リソースは同一か?
S511
レコードkのフラグメント範囲を変数indexに追加する
S512
コードkに対応する送信済みフラグを反転する
S513
k--
S514
フラグメント動画の取得要求を送信する
S515
i++
S516

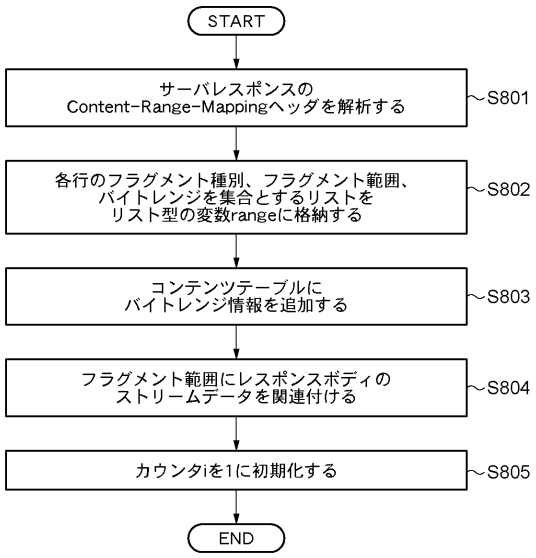
【図１０】


START
HTTPのリクエストメソッドを解析する
S601
パス情報を解析する
S602
Rangeヘッダを解析する
S603
フラグメント種別は?
S604
その他
空間
時間
S605
float型の変数listを初期化する
S606
フラグメント範囲の文字列データを文字「,」で分割する
S607
各文字列データをfloat型に変換して変数listに順に追加する
S608
float型の変数listを初期化する
S609
フラグメント範囲の文字列データを文字「,」で分割する
S610
各文字列データをfloat型に変換して変数listに順に追加する
S611
char型の変数listを初期化する
S612
フラグメント範囲の文字列を文字「&」で分割する
S613
各文字列を変数listに順に追加する
END

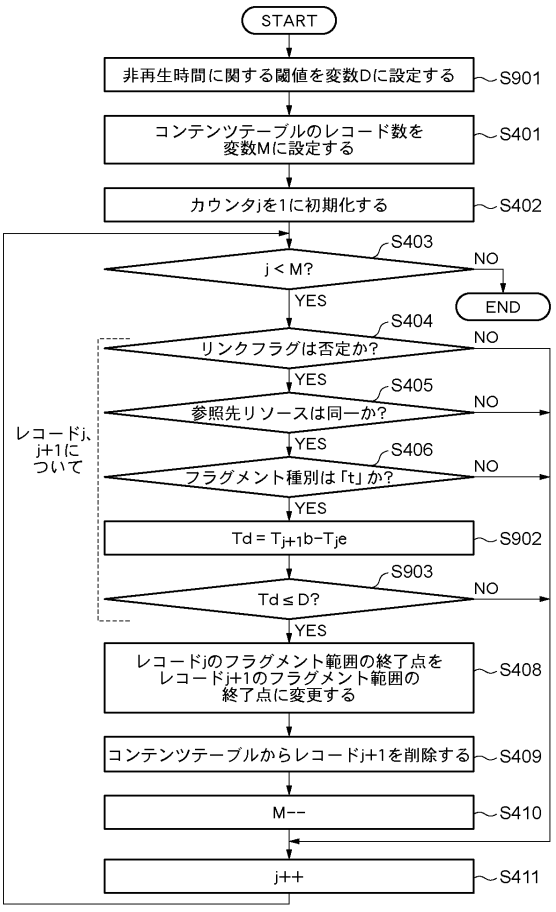
【図１１】



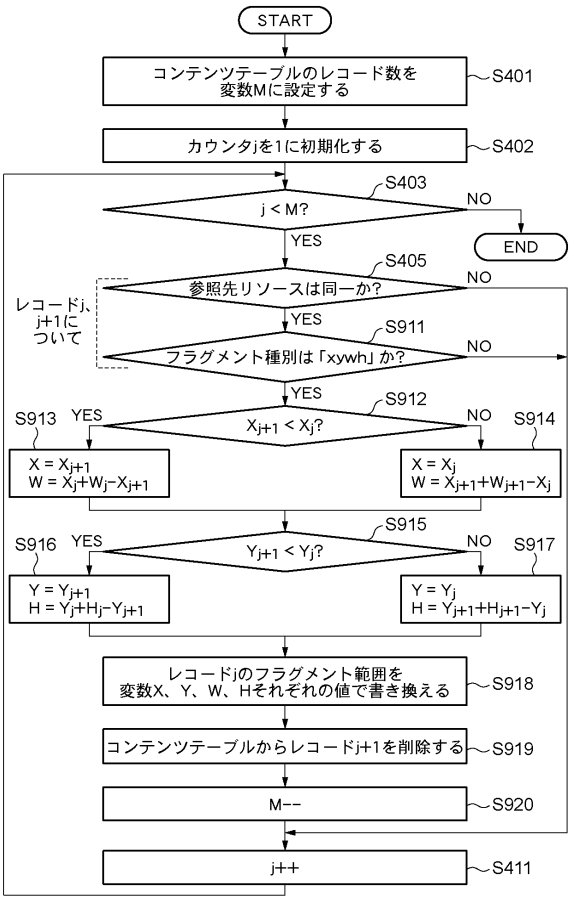
【図１２】



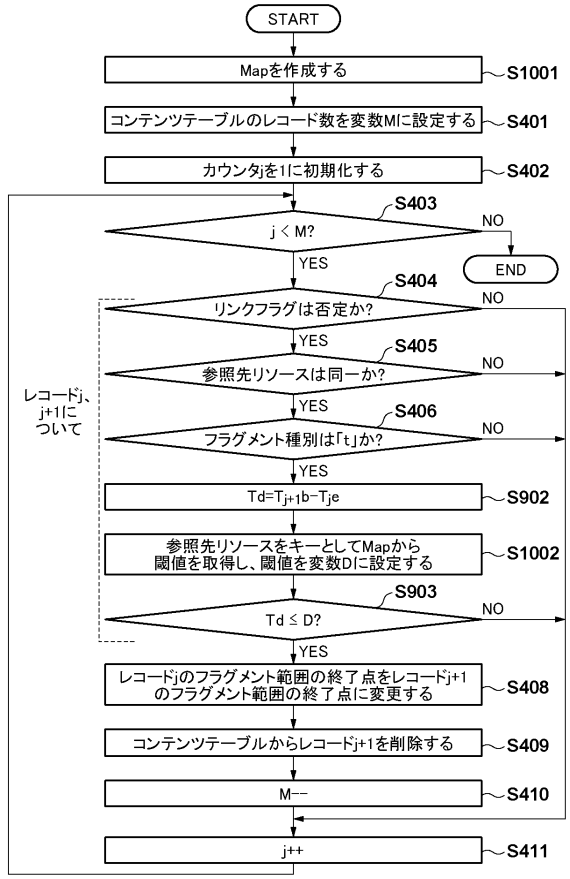
【図１４】



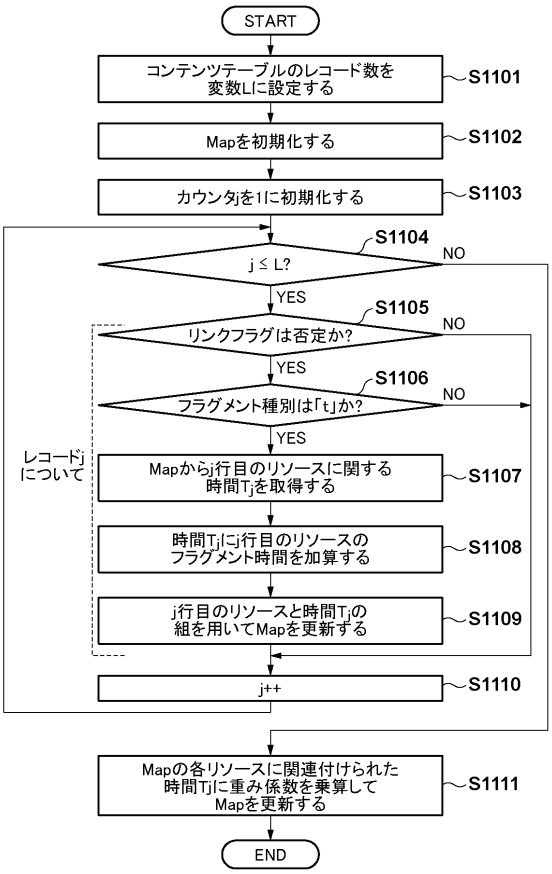
【図１７】



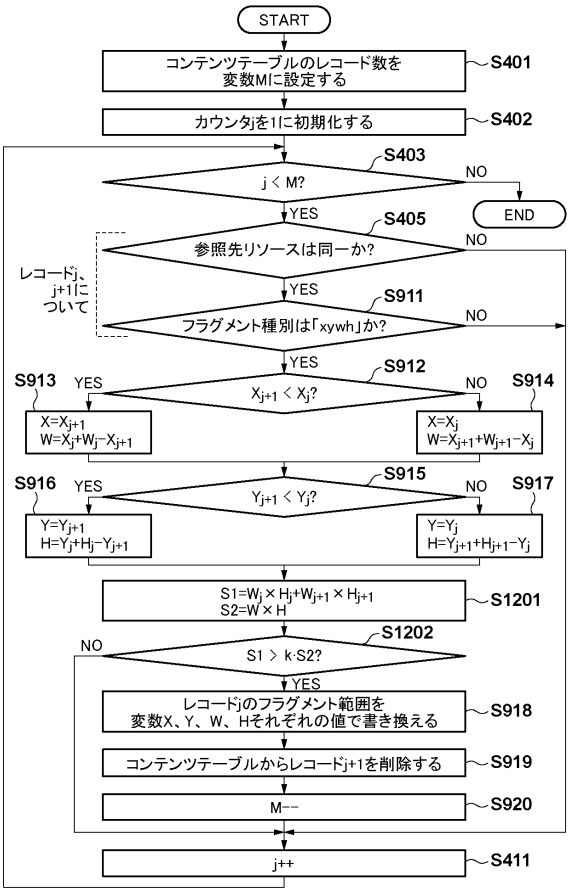
【図１９】



【図２０】



【図２１】



【図２２】

(a)

プロトコル	ホスト名	パス	時間フラグメント	空間フラグメント	リンクフラグ
http	sample.org	/v1.ogv	[10, 20]	[170, 360, 320, 240]	false
http	sample.org	/v2.ogv	[100, 200]		false
http	sample.org	/v1.ogv	[50, 60]	[160, 120, 320, 240]	false
http	sample.org	/v1.ogv	[60, 70]	[170, 360, 320, 240]	false
http	sample.org	/v1.ogv	[70, 80]		false

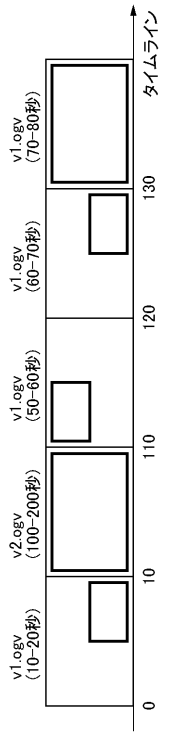
(b)

プロトコル	ホスト名	パス	時間フラグメント	空間フラグメント	リンクフラグ
http	sample.org	/v1.ogv	[10, 20]	[170, 360, 320, 240]	false
http	sample.org	/v2.ogv	[100, 200]		false
http	sample.org	/v1.ogv	[50, 70]	[160, 120, 330, 480]	false
http	sample.org	/v1.ogv	[70, 80]		false

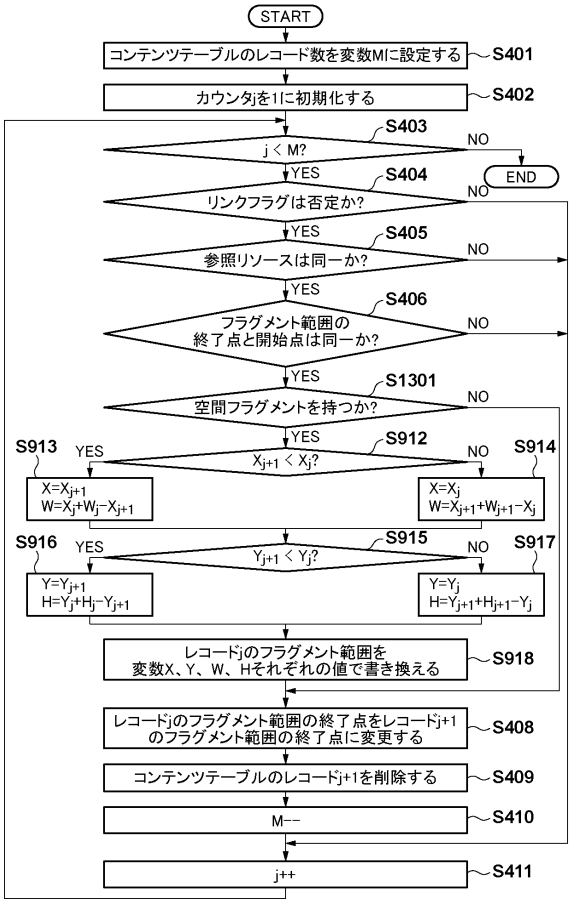
(c)

プロトコル	ホスト名	パス	時間フラグメント	FRAGMENT RANGE	BYTE RANGE	STREAM INSTANCE
http	sample.org	/v1.ogv	[10, 20]	[170,360,320,240]	1000−2000/10000	stream1
http	sample.org	/v2.ogv	[100, 200]		10000−20000/100000	stream2
http	sample.org	/v1.ogv	[50, 70]	[160,120,330,480]	5000−7000/10000	stream1
http	sample.org	/v1.ogv	[70, 80]		7000−8000/10000	stream1

【図２３】


タイムライン
v1.ogv (70-80秒)
v1.ogv (60-70秒)
v1.ogv (50-60秒)
v2.ogv (100-200秒)
v1.ogv (10-20秒)
0
10
110
120
130

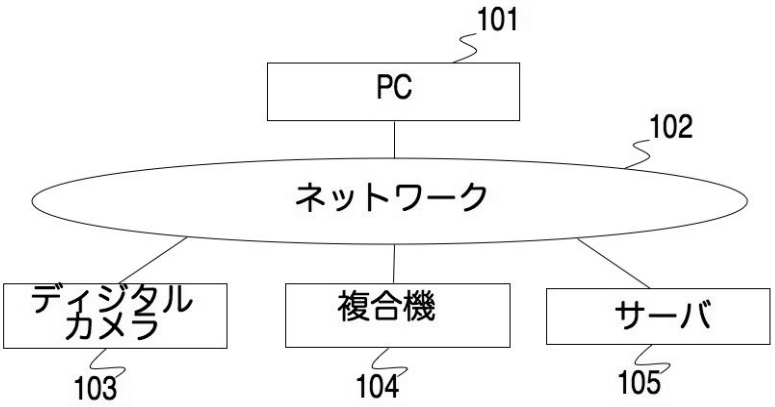
【図２４】


START
コンテンツテーブルのレコード数を変数Mに設定する
S401
カウンタjを1に初期化する
S402
S403
j < M?
NO
END
YES
S404
リンクフラグは否定か?
NO
YES
S405
参照リソースは同一か?
NO
YES
S406
フラグメント範囲の終了点と開始点は同一か?
NO
YES
S1301
空間フラグメントを持つか?
NO
YES
S913
S912
S914
Xj+1 < Xj?
X=Xj+1
W=Xj+Wj−Xj+1
X=Xj
W=Xj+1+Wj+1−Xj
S916
S915
S917
Yj+1 < Yj?
Y=Yj+1
H=Yj+Hj−Yj+1
Y=Yj
H=Yj+1+Hj+1−Yj
レコードjのフラグメント範囲を変数X、Y、W、Hそれぞれの値で書き換える
S918
レコードjのフラグメント範囲の終了点をレコードj+1のフラグメント範囲の終了点に変更する
S408
コンテンツテーブルのレコードj+1を削除する
S409
M−−
S410
j++
S411

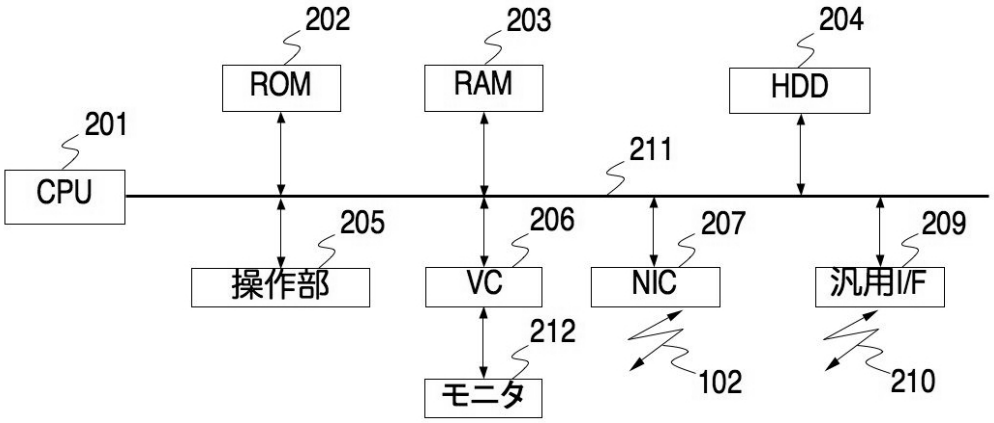
【図１】


v1.ogv (10-20秒)
v2.ogv (100-200秒)
v1.ogv (50-60秒)
v1.ogv (60-70秒)
v1.ogv (70-80秒)
0
10
110
120
130
タイムライン

【図２】


101
PC
102
ネットワーク
ディジタル
カメラ
103
複合機
104
サーバ
105

【図３】


202
ROM
203
RAM
204
HDD
201
CPU
211
205
操作部
206
VC
207
NIC
209
汎用I/F
212
モニタ
102
210

【図６】

(a)

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲	リンクフラグ
http	sample.org	/v1.ogv	t	[10,20]	false
http	sample.org	/v2.ogv	t	[100,200]	false
http	sample.org	/v1.ogv	t	[50,60]	false
http	sample.org	/v1.ogv	t	[60,70]	false
http	sample.org	/v1.ogv	t	[70,80]	true

(b)

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲	リンクフラグ
http	sample.org	/v1.ogv	t	[10,20]	false
http	sample.org	/v2.ogv	t	[100,200]	false
http	sample.org	/v1.ogv	t	[50,70]	false
http	sample.org	/v1.ogv	t	[70,80]	true

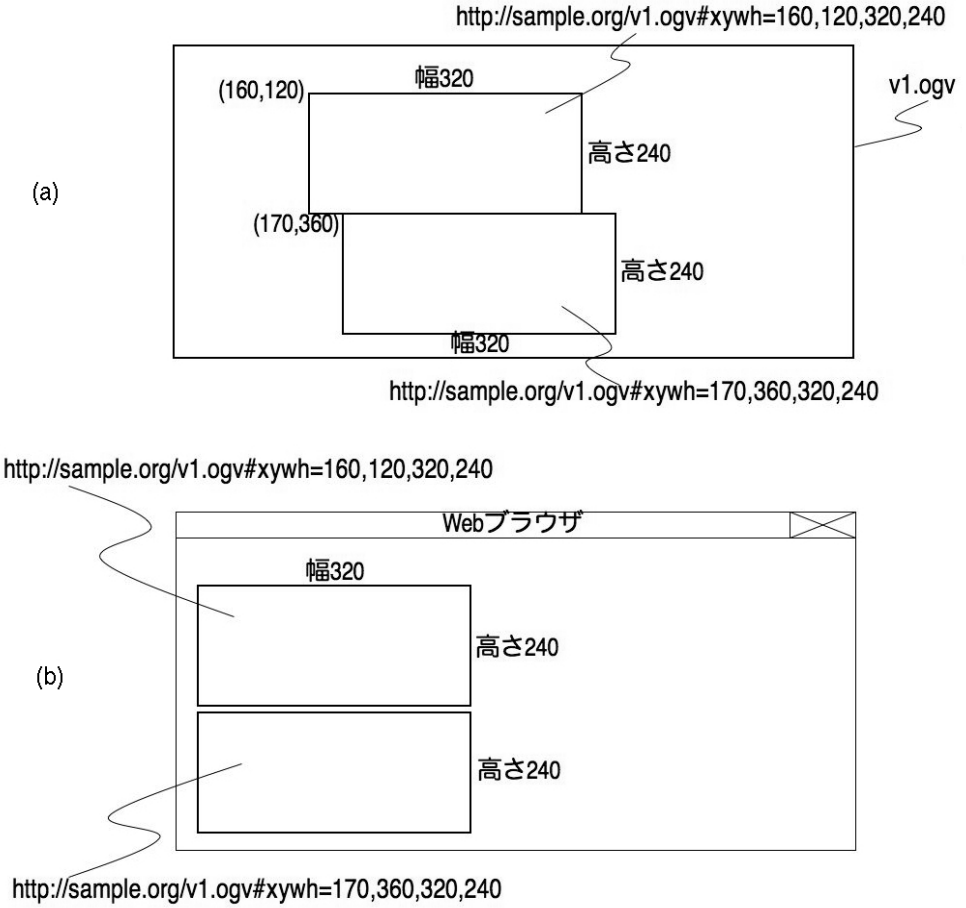
(c)

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲	バイトレンジ	ストリームインスタンス
http	sample.org	/v1.ogv	t	[10,20]	1000-2000/10000	stream1
http	sample.org	/v2.ogv	t	[100,200]	10000-20000/100000	stream2
http	sample.org	/v1.ogv	t	[50,70]	5000-7000/10000	stream1
http	sample.org	/v1.ogv	t	[70,80]	7000-8000/10000	stream1

【図１３】

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲
http	sample.org	/v1.ogv	t	[10,20]
http	sample.org	/v2.ogv	t	[100,200]
http	sample.org	/v1.ogv	t	[50,60]
http	sample.org	/v1.ogv	t	[65,70]

【図１５】



【図１６】

(a)

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲
http	sample.org	/v1.ogv	xywh	[160,120,320,240]
http	sample.org	/v1.ogv	xywh	[170,360,320,240]

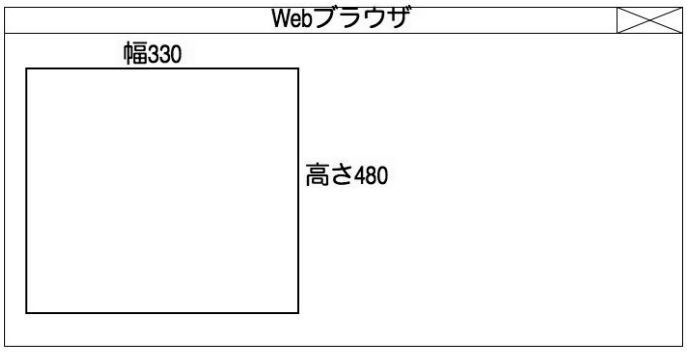
(b)

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲
http	sample.org	/v1.ogv	xywh	[160,120,330,480]

(c)

プロトコル	ホスト名	パス	フラグメント種別	フラグメント範囲	バイトレンジ	ストリームインスタンス
http	sample.org	/v1.ogv	xywh	[160,120,330,480]	1000-2000/10000	stream1

【図１８】



フロントページの続き

(72)発明者　内田　均
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　古河　雅輝

(56)参考文献　特開２０１０－２２５００３（ＪＰ，Ａ）
特開２００６－１８５３８４（ＪＰ，Ａ）
特開２００４－１８０２５８（ＪＰ，Ａ）
特開２００４－１３９５６５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　１３／００
Ｈ０４Ｎ　７／１０
Ｈ０４Ｎ　７／１４－　７／１７３
Ｈ０４Ｎ　７／２０－　７／６８
Ｈ０４Ｎ　２１／００－２１／８５８