コンテンツの有効時間を考慮した放送型ハイパーメディアの配信モデルとその時間的一貫性管理

野田玲子[†] 角谷和俊[††] 田中克己[†††]

本論文では，ハイパーメディアの放送型配信モデルと，配信コンテンツの有効時間の一貫性を保つための版管理方式を提案する．対象とするハイパーメディアは，ハイパーリンク構造を有するハイパーテキスト，および複数のマルチメディアコンテンツの再生時間の同期を指定した同期化コンテンツである．特に，同期化コンテンツの配信方式として，サーバ側でコンポーネントをあらかじめ同期化し，パッケージ化して配信するパッケージ型配信方式，およびサーバ側でコンポーネントのみを配信し，クライアント側で動的に同期化するコンポーネント型配信方式の２つの方式を提案する．さらに，ハイパーメディアの放送型配信について，有効時間が付加されたコンテンツの版管理をサーバ側とクライアント側の双方で行うことにより，配信コンテンツの時間的一貫性を保つための版管理方式について述べる．

A Valid-time Based Delivery Model and Temporal Consistency Management for Hypermedia Broadcasting

REIKO NODA[†], KAZUTOSHI SUMIYA [††] and KATSUMI TANAKA [†††]

In this paper, we describe a broadcasting model for hypermedia, and we also describe a version control method for the model. We propose two broadcasting mechanisms for synchronized contents. One is a package delivery mechanism, in which a server synchronizes components and delivers synchronized content to its clients. The other is a component delivery mechanism, in which a server delivers components, and each clients receive and synchronize these components dynamically. Furthermore, we propose a version control method by which the temporal consistency is kept at both the server and its clients.

1. はじめに

近年，インターネットやデジタル放送を用いた情報配信サービスが注目を集めている．特に，WWW(World Wide Web) による情報提供サービスが急激に増加し，様々な情報を共有することが可能になってきている[1]．これらの情報配信システムは，ユーザがあらかじめ興味のあるチャンネルやソースを指定しておくと，自動的に最新の情報が配信されるプッシュ型サービスに基づいている．現在普及しているプッシュ型サービスとして，PointCast Network[2]，Castanet2.0[3] などがある．これらのプッシュ型サービスで配信される情報は，テキスト情報だけでなく，ハイパーテキストやハイパーメディアを含むことが可能である．ハイパーメディアの配信においては，情報が頻繁に更新されるために，情報の価値が時間の経過とともに変化する情報を扱う場合の配信データ，およびその参照情報であるリンクの管理が大きな課題となっている[4]．

一方，１セットの独立したマルチメディアオブジェクトを１つの同期マルチメディア表現に統合することを可能にする同期マルチメディア統合言語 (the Synchronized Multimedia Integration Language) (SMIL1.0)[5][6] を用いた同期化コンテンツの情報提供が注目を集めている．これらのコンテンツは，自動的にユーザに配信されるわけではなく，あらかじめブラウザに設定されているチャンネルを指定するか，コンテンツの場所を指定することでアクセスしなければ閲覧することができない．また，SMIL コンテンツは複数のマルチメディアファイルから構成されているため，ファイルの削除などによりアクセスできない場合や，

† 神戸大学大学院自然科学研究科情報知能工学専攻
Division of Computer and Systems Engineering, Graduate School of Science and Technology, Kobe University
†† 神戸大学都市安全研究センター都市情報システム研究分野
Research Center for Urban Safety and Security, Kobe University
††† 神戸大学大学院 自然科学研究科 情報メディア科学専攻
Division of Infomation and Media Sciences, Graduate School of Science and Technology, Kobe University

時間の経過とともに情報の価値がなくなってしまったコンテンツにアクセスしてしまう場合がある．また，スポーツ中継などのリアルタイム情報の配信においては，あらかじめ同期の記述を行うことが難しい．従って，同期化コンテンツの自動的配信と，その時間的有効性の管理を行う必要がある．

本論文で扱う情報配信システムは，これらの情報の時間的な有効性に着目した情報配信システムである．我々は，ハイパーテキストの配信と，同期化コンテンツの配信の２つの配信方式について，情報の時間的一貫性を保つ方式についての考察を行う．特に，同期化コンテンツの配信方式として，

- サーバ側でコンポーネントをあらかじめ同期化し，パッケージ化して配信する**パッケージ型配信方式**
- サーバ側でコンポーネントのみを配信し，クライアント側で動的に同期化する**コンポーネント型配信方式**

の２つの方式を提案する．

これらのハイパーメディアの放送型配信において，情報の時間的一貫性を保つ上での問題点は以下の通りである．

- クライアントは常時受信しているとは限らない．
- クライアントが情報を受信しなかった場合，サーバとクライアントでコンテンツの矛盾が生じる場合がある．

本研究では，これらの問題点を解決するために，配信情報の時間的一貫性を保つためのバージョン管理方式を提案する．本方式では，サーバ側とクライアント側でのバージョン管理を行うことで，クライアント側がどんな状態であっても，情報の時間的一貫性を保つことが可能である．

以下，本論文の構成を示す．まず，2 ではハイパーメディア放送型情報配信方式とその問題点について述べる．3 では配信コンテンツの時間的有効性について述べる．4 では 3 で述べた情報の有効時間の一貫性を保つためのバージョン管理方式について述べる．5 ではまとめと今後の課題について述べる．

2. 放送型ハイパーメディアの配信方式

2.1 放送型配信方式

放送型配信方式では，サーバが各クライアントの状況を保持しない．従って，サーバが各クライアントの状況を把握する時に生じるネットワークの負荷や，クライアントの情報を保持するためのコストを削減することができる．

従って，本論文で扱う放送型配信方式とは，以下のような環境を前提としている．

- クライアントはサーバに接続／送信要求を行う．
- サーバはクライアントからの接続・送信要求を受け，その時点での最新のバージョンを送信する．

本論文で取り扱う放送型配信は，大きく次の二つに分けられる．一つがハイパーリンクにより関連づけられたハイパーテキストの配信を行う**ハイパーテキストの放送型配信**，もう一つが動画像，静止画像，音声，テキストなどのマルチメディアコンテンツを同期化したコンテンツを配信する**同期化コンテンツの放送型配信**である．

2.2 ハイパーテキストの放送型配信

時間の経過に伴って変更される情報や時系列データ，およびリアルタイム性を持つ情報をハイパーテキストとして配信する方式である．放送型情報配信システム *Mille-feuille*[4),7),8)] は，サーバでリンク定義とコンテンツを別に管理し，クライアント側で動的リンクを生成するハイパーテキスト配信システムである．このシステムでは，リンクは有効時間が付与された時間依存リンク[4)]であり，文書やアンカーに有効時間を付加し，リンクの時間的有効性を管理することができる．すなわち，単純に動的リンクを生成するだけでなく，リンク元とリンク先のアンカー，およびリンク元とリンク先のコンテンツの時間的有効性を検査し，時間的矛盾のないリンクを生成することが可能である．

また，このシステムでは，サーバ側で一部の情報が更新される毎に，新しいバージョンをクライアントに配信し，クライアントは受信したバージョンを蓄積する．これにより，クライアントは古いバージョンの有効時間をチェックした上で古いバージョンを参照することが可能である．

2.2.1 ハイパーテキストの放送型配信の問題点

ハイパーテキストの放送型配信において，クライアント側での情報の時間的一貫性を保つためには，文書やアンカーに付加された有効時間が正しくなければならない．

ここで，情報の時間的な有効性は，内容が更新された場合に変更される可能性がある．例えば，内容が更新され，新しいバージョンが作成された際に，古いバージョンの有効時間が修正される場合や，情報が削除される場合などである．クライアントでは，このような有効時間の変更は，新しいバージョンを受信することによってのみ修正可能である．しかし，放送型配信ではサーバからクライアントに対して一方的にコンテンツが配信されるため，クライアントがなんらかの理由により，配信された情報を受信できない場合に，

蓄積しているバージョンに矛盾が生じる可能性がある．例えば，クライアント側が受信状態でない場合や，配送エラーが起こった場合，配送順序が前後する場合である．

従って，クライアントがどのような状態の場合でも，蓄積されている情報の有効時間に矛盾が生じないようにするための機構が必要である．すなわち，有効時間の一貫性を保つためのバージョン管理を行う機構が必要である．

これまでのハイパーテキストにおける時制管理は，Dexter ハイパーテキスト参照モデル[9]に時間概念を導入した Amsterdam ハイパーメディアモデル[10]に代表されるように，主にメディアの同期をとるための枠組みについて議論されてきた．このモデルでは，ハイパーテキストの放送型配信におけるデータの時間的有効性を考慮した枠組みについては検討されていない．我々は，ハイパーテキストの放送型配信におけるデータの関連性と，時間的一貫性を保つためのバージョン管理方式を提案する．

2.3 同期化コンテンツの放送型配信

近年，１セットの独立したマルチメディアオブジェクトを１つの同期マルチメディア表現に統合することを可能にする同期マルチメディア統合言語 (the Synchronized Multimedia Integration Language) (SMIL1.0)[5][6] が提案されている．この言語を用いることで，テレビ番組レベルのマルチメディアプレゼンテーションを作成することが可能である．

SMIL は，２つの主要なタグ<seq>と<par>からなる．タグ<seq>は２つのマルチメディアファイルを順に再生する．タグ<par>は２つのメディアを同時に再生する．この２つを組み合わせることにより，複雑なメディアの同期を記述することができる[6]．

ニュースや天気予報，コンサートのライブ中継，名所の紹介など，さまざまな種類のコンテンツがこの SMIL により作成され，インターネット上で公開されており，RealSystemG2[11] などのブラウザで閲覧可能である．これらの同期化コンテンツによる情報提供における問題点は以下のとおりである．

- 同期化コンテンツ (SMIL コンテンツ)☆は，自動的に配信されるわけではなく，あらかじめブラウザに設定されているチャンネルを指定するか，Web 上などでリンクをたどったり，コンテンツの場所を指定することでアクセスしなければ閲覧することができない．
- 同期化コンテンツは複数のマルチメディアファイルから構成されているため，ファイルの削除などによりアクセスできない場合や，時間の経過とともに情報の価値がなくなってしまったコンテンツにアクセスしてしまう場合がある．
- ニュースのようなプレゼンテーション形式のコンテンツはあらかじめサーバ側で同期の記述を行い，配信することが好ましい．それに対して，スポーツ中継などのリアルタイムな情報を同期化して配信するときは，サーバ側であらかじめ同期の記述を行うことが難しい．

従って，様々な情報に対応した同期化コンテンツの配信方式，及び一貫性管理の機構が必要となる．

我々は，(1) サーバ側で同期を行うパッケージ型配信方式と，(2) クライアント側で動的に同期化を行うコンポーネント型配信方式の２つの方式を提案する．コンポーネントは同期化コンテンツを構成するマルチメディアファイルである．この２つの配信方式を提案することにより，様々な情報を同期化コンテンツとして配信することが可能になる．

パッケージ型配信方式は，コンポーネントを同期化コンテンツとしてパッケージ化して配信する方式であり，パッケージ単位での更新が行われる．この方式は，ニュースなどのプレゼンテーション形式のコンテンツの配信に適している．

コンポーネント型配信方式は，クライアント側でサーバから配信されたコンポーネントを動的に同期化コンテンツとして同期化する方式で，Ｆ１などのスポーツ中継などのあらかじめサーバ側で SMIL を記述できないリアルタイムな情報配信に適している．

パッケージ型配信方式ではサーバ側でコンポーネントを同期化した SMIL ソース☆☆を生成するのに対し，コンポーネント型配信方式ではクライアント側で動的に SMIL ソースを生成する．

また，パッケージ型配信方式では，パッケージとなる SMIL ソースに対して有効時間が与えられ，この時間外に再生されることを防ぐことができる．一方のコンポーネント型配信方式では，各コンポーネントに有効時間を付与し，この有効時間からクライアント側で生成された同期化コンテンツの有効時間を計算することが可能である．

☆ 以下，SMIL コンテンツのことを同期化コンテンツと呼ぶ．

☆☆ SMIL ソースとは，コンポーネントの同期の記述を行ったものである．従って，同期化コンテンツは SMIL ソースとコンポーネントから構成される．

2.3.1 同期化コンテンツの定義

[I] パッケージ型配信方式における同期化コンテンツの定義

同期化コンテンツ S はいくつかのコンポーネント c_i の集合である．従って，パッケージ型配信方式における同期化コンテンツを記述する SMIL ソース S_p およびコンポーネント c_i は以下のように定義される．

$$S_p = (ID, Version, \{c_i | i = 1, 2, \cdots, n\}, Tr, [V_{S_1}, V_{S_2}])$$

$$c_i = (id, version, tr, p_i, d_i)$$

ID,id は識別子，$Version$,$version$ はバージョン番号を表す．バージョン番号は，メジャー番号とマイナー番号の２つの組で表される．メジャー番号が内容の更新，マイナー番号が有効時間の更新を表す．バージョン番号が (i, j) のコンポーネントを c_{ij} と表す．

サーバ側で SMIL ソース及び各コンポーネントは識別子 ID, id で管理されており，ある id を持つコンテンツがバージョンアップされた場合，新しいバージョンと古いバージョンは同じ id を付与されて送信される．従って，あるコンポーネントのバージョンは，コンポーネントの id とバージョン番号 ver の組で一意に決まる．また，コンポーネントの各バージョンには有効時間を付与し，価値がなくなった情報や，サーバ側で削除されたコンポーネントにアクセスすることを防ぐことが可能である．

Tr, tr はそれぞれ SMIL ソース及びコンポーネントの各バージョンがサーバ側のデータベースに登録された時刻を表すトランザクション時間である．

p_i は c_i の再生時間であり，テキストや画像などの静止メディアの再生時間は SMIL ソース内に記述されている．

d_i は c_i の再生開始時間であり，S が再生されてから c_i が再生を開始するまでの時間である．これらは SMIL ソース内の記述により求めることができる．

$[V_{S_1}, V_{S_2}]$ は SMIL ソース S_p に与えられた有効時間であり，同期化コンテンツが再生を開始することが可能な時区間を表す．有効時間はサーバ側において絶対時間で指定する．

[II] コンポーネント型配信方式における同期化コンテンツの定義

コンポーネント型配信方式における SMIL ソース S_c およびコンポーネント c_i は以下のように定義される．

$$S_c = (ID, Version, \{c_i | i = 1, 2, \cdots, n\}, [V_{S_1}, V_{S_2}])$$

$$c_i = (id, version, p_i, d_i, tr, [v_{i_1}, v_{i_2}])$$

$[V_{S_1}, V_{S_2}]$ は S の有効時間である．コンポーネント型配信方式モデルにおいては，クライアント側でコンポーネントの有効時間より計算される．

$[v_{i_1}, v_{i_2}]$ は c_i の有効時間である．各コンポーネントの有効期限であり，同期化コンテンツ及びコンポーネントが再生を開始することが可能な時区間を表す．有効時間は絶対時間で指定する．

p_i は c_i の再生時間であり，コンポーネント自身が保持している値である．

d_i は c_i の再生開始時間である．この値は，クライアントでの SMIL ソース生成時にコンポーネント自身が保持している時間情報やコンポーネント間の時間的関連情報などの同期情報をもとに計算される．同期情報に関しては 2.3.3 節で詳述する．

ID,id,$Version$,$version$,tr はパッケージ型配信方式と同様である．

2.3.2 パッケージ型配信方式

パッケージ型配信方式は，サーバ側で，複数のマルチメディアファイルを SMIL により同期化し，一つのプレゼンテーションとしてクライアントに配信する方式である．各コンポーネントのあるバージョンを同期化コンテンツとしてパッケージ化して配信するため，コンポーネントが更新されると，それを含む同期化コンテンツも更新され，更新されたコンポーネントとそれを含む SMIL ソースのバージョンがクライアント側にパッケージとして配信される．図 1 は，パッケージ型配信方式を示している．

(1) 図 1 において，サーバ側でコンポーネント a_1, b_1, c_1, d_1 を同期化しパッケージ化した同期化コンテンツ S_1 をクライアント側に配信する．

(2) サーバ側において，コンポーネント b_1 が更新され，b_2 が作成されると，a を含む同期化コンテンツも S_2 に更新され，クライアントに配信される．図 1 の場合は，この時点ではクライアントがシステムを起動していないため，配信されない．

(3) コンポーネント a_1 が更新され，a_2 が作成されると，これを含む同期化コンテンツも S_2 に更新される．クライアントは，システムを起動している間にサーバに送信要求を行い，それを受けたサーバはクライアントに最新のバージョンの同期化コンテンツ及びコンポーネントを配信する．

図 1 において，クライアント S_3 を再生する場合，すでに受信している c_1, d_1 及び，新たに受信した a_2, b_2 を再生することになる．また，S_3 を受信した後に S_1 を再生すると，S_1 が作成されたときのバージョン

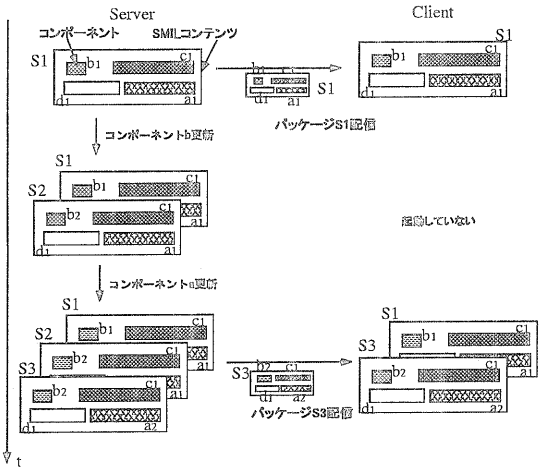


図 1 パッケージ型配信方式

a_1, b_1, c_1, d_1 が再生されることになる.

SMIL ソースには有効時間が付加されており，配信された同期化コンテンツが再生される際にこの有効時間のチェックを行うことで，価値のなくなった同期化コンテンツが再生されることを防ぐことができる.

パッケージ型配信方式の例として，ニュース番組の配信があげられる. SMIL を用いたニュースの配信を考えると，ニュースはいくつかの動画やテキストなどのコンポーネントから構成されている. それらを同期化した SMIL ソースに対して有効時間を与えることにより，ユーザに対して適切な時刻にそのコンテンツを呈示することができる.

図 2 はオーケストラのコンサートの模様を同期化コンテンツとしてパッケージ化して配信した例である. この同期化コンテンツには有効時間が付与されており，コンサートの開催期間のみ閲覧可能である. コンサート開催期間が終わり，次のコンサートが始まると，新しいバージョンの同期化コンテンツが作成され，前のコンサートのコンテンツの有効時間が終了する. これによって，常に最新のコンサートの情報のみをユーザに提供することが可能である.

この同期化コンテンツの SMIL ソースは以下のようになっている.

```
1.<smil>
2. <head>
3.  <layout>
4.   <root-layout background-color="#000000"
      height="349" width="576"/>
5.   <region id="shoukai" left="345"
      top="212" height="133" width="225"/>
6.   <region id="program" left="346"
      top="6" height="200" width="226"/>
7.   <region id="setumei" left="9"
      top="213" height="132" width="333"/>
8.   <region id="video" left="6"
      top="10" height="213" width="332"/>
9.  </layout>
10. </head>
11. <body>
12.  <par>
13.   <seq>
14.    <par>
15.     <video
         src="unmei1.mpg" region="video"/>
16.     <textstream
         src="unmei1.rt" region="shoukai"/>
17.    </par>
18.    <par>
19.     <video
         src="unmei4.mpg" region="video"/>
20.     <textstream
         src="unmei4.rt" region="shoukai"/>
21.    </par>
22.   </seq>
23.    <textstream
         src="pro_un.rt" region="program"/>
24.    <textstream
         src="unmei.rt" region="setumei"/>
25.  </par>
26. </body>
27.</smil>
```

このSMIL ソースにおいて，3 行目から 9 行目が同期化コンテンツのレイアウト指定である. ここでは 4 つの領域 (video,program,setumei,shoukai) が指定されている. また，11 行目から 26 行目までが同期の記述である. <par>タグはタグ内に記述された要素を同時に再生することを示す. また，<seq>タグはタグ内の要素を順に再生することを示す.

図 2 において，15 行目で指定されたコンサートの映像，23 行目で指定されたプログラム，16 行目で指定された演奏中の曲および楽章に関するデータ，24 行で指定された及び演奏中の楽章を示すデータを同期化し，それぞれ左上，右上，左下，右下の矩形の領域に表示している.

このようにコンサートに関係する様々な形式のデータを同期化コンテンツとしてパッケージ化し配信することで，現在開催中のコンサートの模様をわかりやす

図 2 パッケージ型配信方式の表示例

くユーザに提供することが可能である．

2.3.3 コンポーネント型配信方式

コンポーネント型配信方式では，サーバ側で作成／更新され配信されるコンポーネントを，クライアント側で動的に同期化を行い，同期化コンテンツに合成する．更新単位はコンポーネントとなる．クライアント側で合成される同期化コンテンツはクライアント側に蓄積され，参照することが可能である．各コンポーネントはそれぞれ更新間隔が異なり，それぞれに対して有効時間を付与することで，価値のないコンポーネントにアクセスすることを防ぐことができる．

例えば，天気予報は「天気図」（画像）や「明日の天気」（テキスト）「降水確率」（テキスト）などをクライアントで動的に同期化したコンテンツだと考えられるが，「天気図」は6時間に1回更新，明日の天気は1日1回更新など，更新間隔が異なるため，コンポーネントごとの版管理を行う必要がある．また，6月1日の「明日の天気」という情報は6月1日中に見ないと意味がない情報であり，「6月1日9時現在の天気図」という情報は「6月1日12時現在の天気図」が発表されるまでに見ないと意味のない情報である．従って，「明日の天気」というコンポーネントには，[6/1 0:00-6/1 23:59] という有効時間を指定し，「6月1日9時現在の天気図」という情報には [6/1 9:00-6/1 11:59] という有効時間を指定する必要があり，この時間外に再生されることのないようにしなければならない．

コンポーネントは更新され，バージョンアップが行われるごとに配信される．ここでコンポーネントの各バージョンとはクライアント側で同期化するときに再生される順番を決定するためのものである．すなわち，バージョン2のコンポーネントはバージョン1のコンポーネントの後に再生されることになる．例えば，

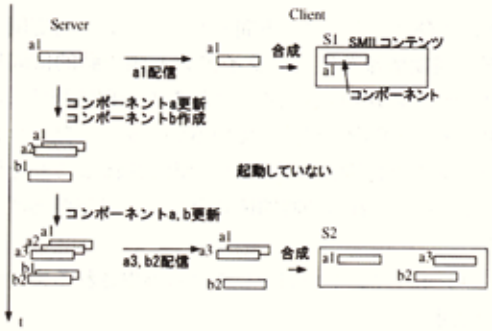


図 3 コンポーネント型配信方式

F1の周回タイムデータというコンポーネントを考えた場合に，1周目のタイムデータの次に2周目のタイムデータが再生されることになる．すなわち，1週目のデータが，周回タイムデータのバージョン1であり，2周目のデータが周回タイムデータのバージョン2ということになる．サーバ側で各コンポーネントは識別子（id）で管理されており，ある id を持つコンテンツがバージョンアップされた場合，新しいバージョンと古いバージョンは同じ id を付与されて送信される．従って，あるコンポーネントのバージョンは，コンポーネント id とバージョン番号 ver の組で一意に決まる．また，コンポーネントの各バージョンには有効時間を付与し，価値がなくなった情報や，サーバ側で削除されたコンポーネントにアクセスすることを防ぐことが可能である．

図3はコンポーネント型配信を示している．

(1) 図3において，サーバ側でコンポーネント a_1 が作成され，クライアント側に配信される．クライアント側では到着したコンポーネントを動的に同期化し，同期化コンテンツを生成する．

(2) サーバ側において，コンポーネント a_1 が更新され a_2 が作成される．また新たなコンポーネント b_1 が作成される．図3の場合は，この時点ではクライアントがシステムを起動していないため，配信されない．

(3) さらに，コンポーネント a_2, b_1 が a_3, b_2 に更新される．この時点で，クライアントがシステムを起動すると，サーバに送信要求を行い，それを受けたサーバは最新のバージョンのコンポーネントである a_3, b_2 を配信する．クライアントは最新バージョンを受け取ると，同期化コンテンツを再合成する．

コンポーネント型配信方式では，コンポーネント自身がもつ時間情報や，コンポーネント間の時間的関連

情報，及びコンポーネント間のバージョンの対応関係などの同期情報を元にクライアント側で動的に同期化コンテンツが生成される．これらの同期情報は，コンポーネント自身が保持しているか，サーバ側でコンポーネントとは別に作成されて配信されることを仮定している．これらの時間情報として，以下があげられる．

- コンポーネントが到着してから再生されるまでの時間
- コンポーネントの到着時間
- 前のバージョンが再生開始あるいは終了してから再生されるまでの時間
- 別のコンポーネントのあるバージョンが再生開始あるいは終了してから再生されるまでの時間

これらの時間情報を用いて，クライアント側で動的に同期化を行う．しかし，これらの情報の配信手段及び，関連づけられたバージョンが再生開始時間を過ぎても到着しなかった場合や，逆に再生中に次のバージョンが到着した場合などの同期化の制御方式に関しては別途定める必要があるが，本論文では２つの配信方式の提案と，両者に共通して使用できる版管理方式の提案が主目的であり，これらの話題については，今回の論文では対象としていない．

また，一つのSMILコンテンツを構成するコンポーネントの種類や数はあらかじめ固定されており，コンポーネントが追加・削除されることはないことを前提としている．この前提のもとで、あらかじめコンポーネントの空間配置を行っている．

ここでクライアント側におけるコンポーネントのバージョン間に以下の関連づけを行う．この関連づけはコンポーネントのバージョンの対応関係により，以下の３つに類別される．

(1) n対n対応:(id_1, id_2)
(2) n対1対応:$(id_1, (id_2, ver))$
(3) 1対1対応:$((id_1, ver_1), (id_2, ver_2))$

この３つの対応関係により関連づけられたコンポーネントのバージョンが一つの同期再生区間に合成される．

(1) バージョンのn対n対応

この関連づけを行われたコンポーネントは，コンポーネント同士を同時再生し，そのコンポーネントのバージョンを順次再生する．

図4(1)は$id1$のコンポーネントと$id2$のコンポーネントを関連づけした場合の動的同期化の例を示している．ここではコンポーネントのバージョンの到着時間及び再生時間を元に，一つの同時再生区間に２つのコンポーネントのバージョンを順次再生するように同期化し，SMILコンテンツを動的に生成している．また，$id2, ver2$及び$id1, ver3$のコンポーネントは前のバージョンが再生中に到着している．この例では，前のバージョンが再生終了した直後に再生を開始するように同期化している．しかし到着時にver1の再生をやめてver2を再生する制御方法も考えられる．この制御方法はアプリケーションに依存する．

この関連づけを行う例として，F1におけるピットの映像とコースのある地点での映像が挙げられる．ピット及びコースのある地点で何かイベントが起こるたびにそれぞれ別々に更新される映像をクライアント側で同時に再生するように同期化することが可能である．

(2) バージョンのn対1対応

この関連づけでは，コンポーネント１とコンポーネント２のあるバージョンを同時再生する．コンポーネント１のバージョンは順次再生される．

図4(2)は，$id1$のコンポーネントと，$id2$のコンポーネントの$verx$のバージョンを関連づけした場合の動的同期化の例を示している．$id2$のコンポーネントの$verx$のバージョンが再生されている間に，$id1$のコンポーネントのバージョンを到着時間を元に順次再生している．

この関連づけを行う例として，F1におけるレース映像とタイムデータが挙げられる．この場合，クライアント側では連続したレース映像と同時に，周回毎に更新されるタイムデータのバージョンを順次再生するように同期化することが可能である．

(3) バージョンの1対1対応

この関連づけでは，コンポーネント１のあるバージョンと，コンポーネント２のあるバージョンを同時再生する．

図4(3)では，$id1$のコンポーネントのバージョン番号と，$id2$のコンポーネントのバージョン番号が同じもの同士を関連づけした場合の動的同期化の例を示している．同じバージョン番号のコンポーネントが到着すると，その２つのコンポーネントを同時再生区間に合成する．

この関連づけを行う例として，F1におけるクラッシュ映像と，その詳細情報が挙げられる．クラッシュの映像$ver1$と，その映像を説明する詳細情報$ver1$を同時に再生し，続いて別地点からの映像$ver2$とその映像を説明する詳細情報$ver2$を同時に再生するような同期化を行うことが可能である．

コンポーネント型配信方式の例として，Ｆ１中継の配信が考えられる．サーバ側で配信されるデータは，

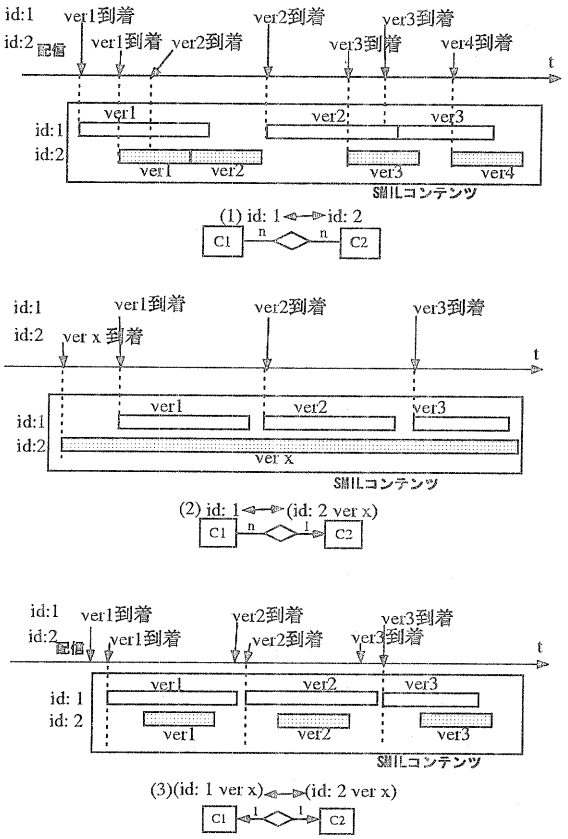


図 4 コンポーネントの動的同期化

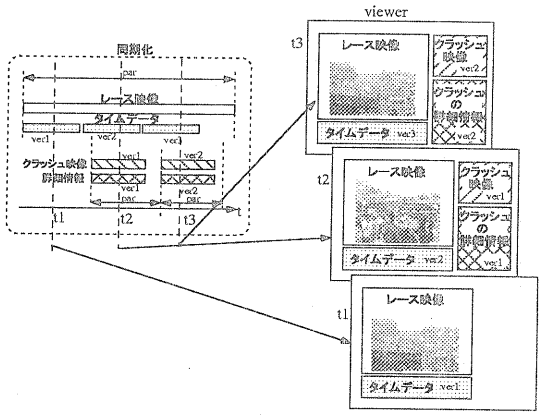


図 5 コンポーネント型配信方式の表示例

レース映像，タイムなどのデータ，クラッシュの映像とその音声解説などである．例えば，レース映像とタイムのデータは上記の (2) で対応づけし，クラッシュの映像と音声解説は上記の (3) で対応づけすると，クライアント側では図 5 のような同期化コンテンツが生成される．また，タイムデータやクラッシュ映像などは頻繁に更新されるため，バージョンが到着する毎に，クライアントは同期化コンテンツを再合成する．

2.3.4 同期化コンテンツの放送型配信の問題点

同期化コンテンツの放送型配信では，複数のコンポーネントを同期化して配信を行う．それぞれのコンポーネントは時間経過に伴い更新される．また，時間の経過とともに情報の価値がなくなることもある．同期化コンテンツの配信では，SMIL ソースや各コンポーネントに有効時間を付加することで，クライアントが価値のない配信情報にアクセスできないようにすることが可能である．

ここで，パッケージ型配信方式における SMIL ソース，及びコンポーネント型配信方式における各コンポーネントに付加された有効時間はサーバ側で変更される．従って，その変更をクライアントが受け取ることができないと，クライアントに蓄積されている情報に矛盾を生じることになる．これはハイパーテキスト配信における問題点と同じ理由から生じる問題点である．

これまでの研究では，映像やその詳細情報を同期化し配信する放送サービスの枠組みとして，ScoopCast[12] が提案されている．ScoopCast は内容記述された映像ストリームをスクリプトを用い，動的に番組構成する放送型サービスである．しかし，配信される映像や内容記述の時間的有効性と，配信情報のバージョン管理については考慮されていない．

従って，同期化コンテンツの配信においても，蓄積されている情報の有効時間に矛盾が生じないようにするための機構が必要である．すなわち，有効時間の一貫性を保つためのバージョン管理を行う機構が必要である．

3. 配信コンテンツの時間的有効性

3.1 放送型ハイパーテキストの時間的有効性

ハイパーテキストの配信システム *Mille-feuille*[4),7),8)] では，リンクは時間依存リンク [4),7)] であり，文書やアンカーに有効時間を付加し，リンクの時間的有効性を管理することができる．すなわち，単純に動的リンクを生成するだけでなく，リンク元とリンク先のアンカー，およびリンク元とリンク先のドキュメントの時間的有効性を検査し，時間的矛盾のないリンクを生成することが可能である [4),7)]．

時間依存リンク [4),7)] は放送型ハイパーテキストのリンク制御を行うリンクモデルである．時間依存リンクはリンク先アンカーとリンク元アンカーの組 a_s, a_d で定義される．

ここで，リンク元及びリンク先のアンカー及びそれ

表1 トランザクション時間と有効時間

	source		destination	
	anchor	doc.	anchor	doc.
valid	$[v_{s_1}, v_{s_2}]$	$[V_{s_1}, V_{s_2}]$	$[v_{d_1}, v_{d_2}]$	$[V_{d_1}, V_{d_2}]$

を含むドキュメントの有効時間を表3.1とすると，時間依存リンクはアンカー及びそれを含むドキュメントがすべて有効である期間のみ存在する．従って，時間依存リンク有効時間 L_{valid} は，リンク先およびリンク元のアンカーとそれを含むドキュメントの有効時間の積として以下のように計算される[4]．ただし，アンカーの有効時間はナビゲーションポイントとして有効な時間，ドキュメントの有効時間はドキュメントにアクセス可能な時間である[4]．

$$L_{valid} = [L_{create}, L_{expire}]$$
$$L_{create} = max(v_{s_1}, V_{s_1}, v_{d_1}, V_{d_1})$$
$$L_{expire} = min(v_{s_2}, V_{s_2}, v_{d_2}, V_{d_2})$$

3.2 同期化コンテンツの時間的有効性

本節では，同期化コンテンツの時間的有効性について述べる．ここで扱う同期化コンテンツはSMILで記述されたコンテンツとする．同期化コンテンツのパッケージ型配信方式では，同期化のシナリオとなるSMILソースに与えられた有効時間が同期化コンテンツの有効時間となる．一方，コンポーネント型配信方式では，同期化コンテンツはいくつかのコンポーネントの集合であり，それらのコンポーネントが再生時に全て有効である時間が同期化コンテンツの有効時間となる．すなわち，同期化コンテンツの有効時間は，同期化コンテンツが再生されてから，コンポーネントが再生されるまでの時間を考慮した有効時間の積で与えられる．従って，前節で述べた時間依存リンクの有効時間の計算方式を拡張することで容易に計算可能である．

3.2.1 パッケージ型配信方式における同期化コンテンツの有効時間

パッケージ型配信方式では，同期化コンテンツの有効時間は同期化コンテンツ S 自身に与えられた有効時間となる．

3.2.2 コンポーネント型配信方式における同期化コンテンツの有効時間

コンポーネント型配信方式において，同期化コンテンツ S の有効時間は，その同期化コンテンツが含むコンポーネントの有効時間から計算される．時刻 t において S を再生するとき，同期化コンテンツ S の全てのコンポーネント c_i について，c_i が再生される時刻，つまり時刻 $t + d_i$ が c_i の有効時間 $[v_{i_1}, v_{i_2}]$ を含んでいなければならない．言い換えると，時刻 t に c_i が再生可能か否かは，区間 $[v_{i_1} - d_i, v_{i_2} - d_i]$ が時刻 t を含んでいるかどうかによって決まる．従って，S が再生可能となる時刻 V_1 は，S 自身，および S の全てのコンポーネント c_i が時刻 t に再生可能とならなければならない．つまり，

$$V_1 = max\{v_{1_1} - d_1, v_{2_1} - d_2, \cdots v_{n_1} - d_n\}$$

となる．

また，S が再生可能でなくなる時刻 V_2 は，S 自身，および S のいずれかのコンポーネントが再生可能ではなくなる時刻と考えることができる．つまり，

$$V_2 = min\{v_{1_2} - d_1, v_{2_2} - d_2, \cdots v_{n_2} - d_n\}$$

となる．

従って，コンポーネント型配信方式における同期化コンテンツ S の有効時間は，$[V_1, V_2]$ と計算される．

4. 放送型配信におけるバージョン管理

配信されるコンテンツは，サーバ側で時々刻々と変更される可能性がある．サーバ側でコンテンツが更新される毎に，新しいバージョンを配信し，クライアント側ではそのバージョンを蓄積する．このとき，何らかの理由によりクライアントが受信できない可能性がある．例えば，クライアント側が受信状態ではない場合，配送順序が前後する場合などである．従って，本研究では以下の環境を前提としている．

- コンテンツは時々刻々と更新され，配信される．
- 更新情報はすべてサーバに蓄積される．
- 配信された更新情報は各クライアントですべて受信されるとはかぎらないが，受信した情報は各クライアントに蓄積される．

ハイパーテキストの配信においては，クライアントに蓄積されているドキュメントやアンカーの有効時間がサーバ側で変更される可能性があり，この変更を受信できない場合にクライアントに蓄積されているドキュメントやアンカーの有効時間に矛盾を生じる．

また，同期化コンテンツの配信において，パッケージ型配信方式ではクライアント側に蓄積されているSMILソースの有効時間が同期化コンテンツの有効時間となる．一方のコンポーネント型配信方式では，クライアント側に蓄積されているコンポーネントの有効時間と再生開始時間から同期化コンテンツの有効時間が計算される．ここで，SMILソース及びコンポーネントの各バージョンの有効時間は，サーバ側で更新されるため，クライアントがその変更を受信しないと，蓄積しているバージョンの有効時間に矛盾を生じる．

すなわち，ハイパーテキスト配信におけるドキュメ

ント，アンカーの有効時間，および同期化コンテンツにおけるSMILソース及びコンポーネントの有効時間の一貫性を保つための問題点は，クライアントがサーバ側における有効時間の変更を新しいバージョンを受け取ることでしか知ることができないことから生じている．

図6はクライアントが受信しなかったことにより，蓄積しているバージョンの有効時間に矛盾が生じる例を示している．図6において，矩形がドキュメントを表し，矩形の下の[6/1,6/30]などがそのドキュメントの有効時間を表す．図6において，6月1日に「6月の予定」というドキュメントが作成され，クライアントに配信されている．このドキュメントの有効時間は，[6/1,6/30]となっている．サーバ側で6月10日に「6月の予定」というドキュメント内の6月15日の予定が修正されている．この時，修正前のドキュメントは修正されたことにより有効時間が変更時の6月10日までと変更されている．しかし，このときクライアントはシステムを起動していなかったため，バージョン2のドキュメントと有効時間が修正されたバージョン1のドキュメントを受け取ることができない．さらに6月30日に「7月の予定」というバージョン3のドキュメントが作成され配信されると，クライアント側に蓄積されているドキュメントのうち，バージョン1のドキュメントの有効時間が不適切なまま蓄積されることになる．

従って，クライアントはサーバ側で有効時間が変更されたバージョンを必ず受け取らなければならないため，サーバは有効時間が変更されたバージョンを必ずクライアントに配信する必要がある．つまり，サーバはコンテンツ内容，および，有効時間の両者の情報の版管理を行う必要がある．そこで，サーバ側での2分木を用いたバージョン管理方式を提案する．本方式では，線形直列なバージョンモデル☆に議論を限定している．このバージョン管理方式では，内容の変更を左の子節点として追加するとともに，有効時間の変更を新しいバージョンとして右の子節点として追加する．そして，配信時にこのバージョン木の葉節点のノードを配信することにより，つねにクライアント側に有効時間の正しいコンポーネントのバージョンを蓄積することができる．本論文で提案する版管理方式は，サーバ側でコンテンツの更新があれば，新しい版ができるとともに，元のコンテンツの有効時間は短縮されるような状況を想定した管理方式である．ただし，有効時間のみが短縮／延長される場合もあり，その場合も本管理方式は，効果を発揮する．なぜならば，本方式では有効時間の変更をおこなったバージョンは最新バージョンとして起動中のクライアントに配信することで，クライアント側に常に有効時間の正しいバージョンのみを蓄積することができるからである．

本節では，ドキュメント，アンカー，SMILソースおよびコンポーネントの有効時間の一貫性を保ち，クライアントに蓄積されるすべての配信コンテンツの時間的一貫性を保つためのバージョン管理方式について述べる．

4.1 バージョン管理の基本単位

ハイパーテキストの配信，及び同期化コンテンツの配信において，クライアントにおける情報の時間的な一貫性を保つためのバージョン管理の対象となるのは，有効時間を付加されている情報である．すなわち，ハイパーテキスト配信におけるドキュメントとアンカー，及び同期化コンテンツ配信におけるSMILソース及びコンポーネントである．有効時間を付加されたこれらの情報単位をユニットとよび，バージョン管理の基本単位とする．

4.2 サーバ側におけるバージョン管理

サーバ側ではドキュメント，アンカー，SMILソース，及びコンポーネントは，内容を更新するごとに新しいバージョンを作成する．また，更新されたことによって，古いバージョンの有効時間を変更する必要がある場合も存在する．従って，サーバ側でのユニットの更新は，2つの種類にわけて行う必要がある．一つが内容の更新，もう一つが有効時間の更新である．

本方式ではこの二つの更新を，2分木を用いることで制御する．

4.2.1 バージョン木

バージョン木は2分木であり，ノードが各バージョンを示す．また，あるノードの左の子節点はそのノードの内容が更新された次のバージョンであり，右の子節点は有効時間の変更が行われたバージョンである☆☆．

バージョン木Tは以下のように定義される．

$$T = (N, E)$$
$$N = \{c_1, c_2, \cdots, c_n\}$$
$$E \subseteq \{(c, c') | c \in N, c' \in N\}$$
$$c = (id, version, tr, [v_1, v_2]), c \in N$$

Nはバージョン木のノードであり，配信コンテンツ

☆ 線形直列なバージョンモデルとは，バージョンの導出関係が線形であるモデルである．すなわち，バージョンの導出時に枝分かれせず，線形直列に導出されるモデルである．

☆☆ ただし，右の子節点はその親節点のノードのコンテンツの内容も保持している．

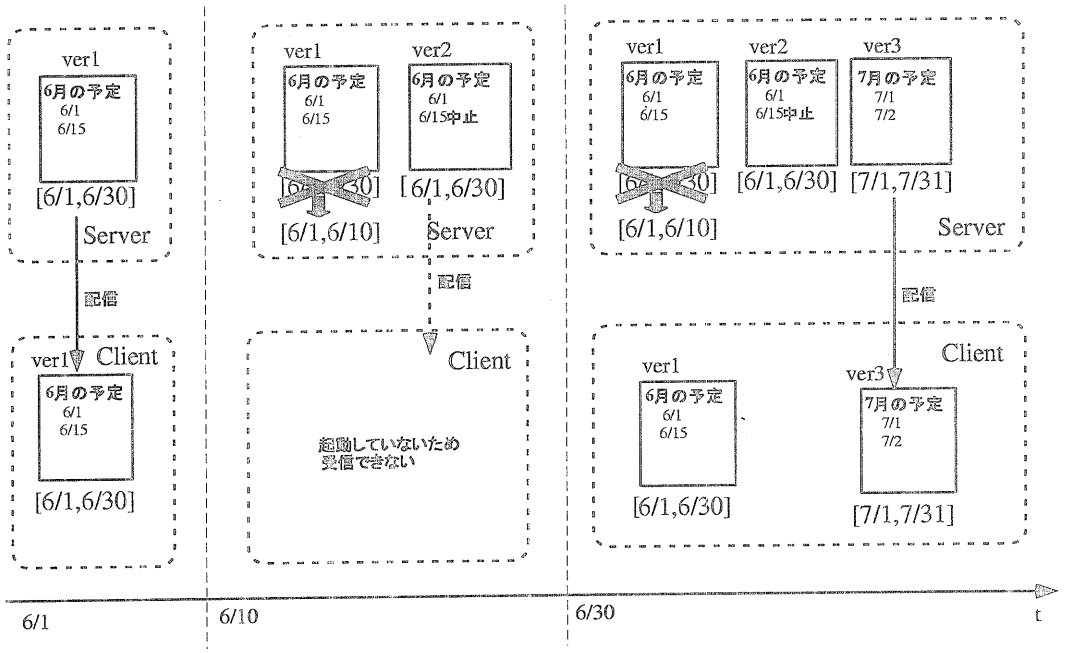


図 6 ハイパーメディア配信の問題点

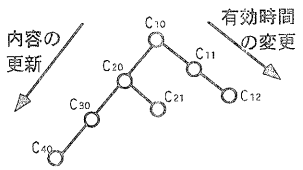


図 7 バージョン木

の更新管理の対象となるユニットの各バージョンの集合を表す.

E はバージョン木の枝である. 左の枝が内容の更新を表し, 右の枝が有効時間の変更を示す.

c はバージョン木 T のノードであり, ユニットの各バージョンを示す. ここで, id は識別子, $version$ はバージョン番号, tr はユニットの各バージョンがサーバ側のデータベースに登録された時刻を表すトランザクション時間, $[v_1, v_2]$ はユニットの各バージョンの有効時間である. バージョン番号は, メジャー番号とマイナー番号の 2 つの組で表される. メジャー番号が内容の更新, マイナー番号が有効時間の更新を表す. バージョン番号が (i, j) のノードを c_{ij} と表す.

4.2.2 バージョン木に対する基本操作

バージョン木に対する基本操作は次の 2 つである. 内容の更新を行った際の操作である *version_up* と, 有効時間の変更を行う *propagate* である.

- *version_up(new-version)*
  あるバージョン木の最新のノード, つまり一番左の葉節点の内容に更新した新しいバージョンのノード *new-version* を左の子として追加する操作. *new-version* のバージョン番号は追加されるノードのバージョン番号を $(i, 0)$ とすると $(i+1, 0)$ となる.
- *propagate*$(n, [v_1, v_2])$
  c_{n0} の有効時間を $[v_1, v_2]$ に変更したものを新しいノードとして, c_{n0} から右の子を辿っていき, 右の子がなくなったらそこに新しいノードを追加する操作. 追加するノードのバージョン番号を (i, j) とすると, 追加したノードのバージョン番号は $(i, j+1)$ となる.

この 2 つの操作を用いて, ユニットの追加, 更新, 削除を行う.

4.2.3 ユニットの追加・削除・更新

[I] ユニットの追加

ユニットの追加は, 新たなユニットのバージョン木を作成し, ユニットを配信する.

[II] ユニットの削除

ユニットの削除に関しては, 次の 2 つの場合がある.

(1) ユニット自体の削除

ユニット c 自体を時刻 t_d に削除したときは, す

べてのバージョンを無効にする必要がある．従って処理は以下のようになる．

- c のバージョン木のすべてのバージョン n について，操作 $propagate(n,[v_1,t_d])$ を行う (図 8(a))

(2) ユニットのあるバージョンの削除

ユニットのあるバージョン c_{k0} を時刻 t_d に削除する時は，以下のような処理を行う．

(a) c_{k0} は削除されたために無効にする必要がある．従って $propagate(k,[v_1,t_d])$ を行う．

(b) 節点 c_{k-1} の有効時間の終了時間が *until-changed*☆ の場合は，次の操作を行う．

- c_{k0} が最新のバージョンではない場合，つまり c_{k0} が存在する場合，$propagate(k-1,[v_1,tr_{k+1}])$ を行う (図 8(b))．ただし，tr_{k+1} は $c_{k+1,0}$ のトランザクション時間である．
- c_{k0} が最新のバージョンの場合，つまり左の子節点が存在しない場合，$propagate(k-1,[v_1,$ *until-changed*$])$ を行う (図 8(c))．

図 8 はユニットの削除の 3 つの場合を示している．

(a) はユニット自体の削除を示している．削除により，全てのバージョンを時刻 t_d で無効にするため，c_{12},c_{22},c_{31} を追加している．

(b) は最新でないバージョン c_{20} の削除を示している．まず，c_{22} を追加して時刻 t_d で無効にし，さらに，1 つ前のバージョン (c_{10}) の有効時間を c_{30} のトランザクション時間 T_3 に変更している．

(c) は最新のバージョン c_{20} の削除を示している．まず，c_{21} を追加して時刻 t_d で無効にし，さらに，1 つ前のバージョン (c_{10}) の有効時間を *until-changed* に変更したノードを追加している．

[III] ユニットの更新

ユニットの内容が時刻 t_c に c_{n0} から $c_{n+1,0}$ に更新された場合は，以下の処理を行う (図 9(a))．

(1) 更新するユニットのバージョン木に対し，*version-up*$(c_{n+1,0})$ を行う．ただし，c_{n+1} のトランザクション時間は t_c である．

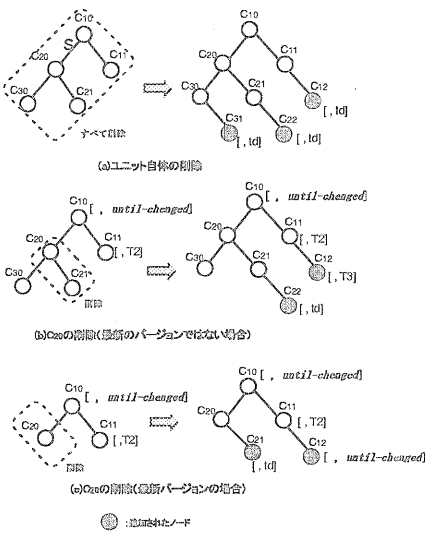


図 8 時刻 t_d におけるユニットの削除

(2) c_n の有効時間が *until-changed* の場合，$propagate(n,[v_1,t_c])$ を行う (図 9(a))．

また，c_{n0} の有効時間を $[v_1',v_2']$ に変更する場合は次の操作を行う (図 9(b))．

(1) $propagate(n,[v_1',v_2'])$ を行い，c_{n0} の有効時間を変更する．

図 9 はユニットの更新の 2 つの場合を示している．

(a) は *until-changed* の場合であり，c_{30} が更新された c_{40} が追加され，また c_{31} を追加し，c_{30} を更新時刻 t_c に無効にしている．

(b) は c_{30} の有効時間の変更を示している．c_{31} を追加することで有効時間の変更を行っている．

さらに，ドキュメントの更新の際，ドキュメント内のアンカーが追加，削除，更新された場合は，そのアンカーに対して，以下の処理を行う．

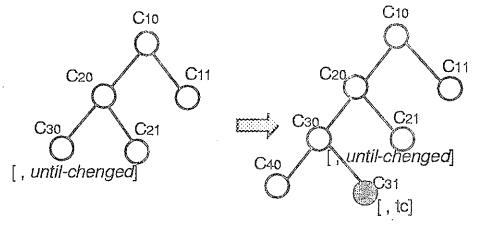
- ドキュメントに新しいアンカーを追加する場合
  新しいアンカーのバージョン木を作成する．
- ドキュメントからアンカー a_n を削除する場合
  削除されたアンカーのバージョン木に対して，4.2.3 のユニット自体の削除と同様の処理を行う．
- ドキュメント内のアンカー a を更新する場合
  a のバージョン木に対して，ユニットの更新と同様の処理を行う．

4.2.4 ユニットの配信

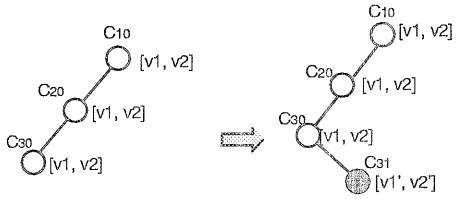
サーバは各クライアントから送信要求を受けた際，クライアントにバージョン木のすべての葉節点のノードを配信する．

クライアントに対するユニットの配信の手順は以下の通りである．

☆ *until-changed* という指定は，次のバージョンが生成されるまで有効であるという，時間変数による指定である．



(a)C_{30}からC_{40}にバージョンアップ



(b)C_{30}の有効時間を変更

図 9 時刻 t_c におけるユニットの更新

(1) クライアントからの送信要求を受ける.

(2) サーバはその時点でのバージョン木のすべての葉節点のノードを配信する.

ここで，葉節点のノードを配信する場合，以前のバージョンの内容の差分や，有効時間の変更だけでなく，コンテンツの内容も送る必要がある．なぜならば，クライアントが内容や有効時間が変更される前のバージョンを受け取っていない可能性があるからである．

また，バージョン木の葉節点のノードのみを配信するため，すべてのバージョンを配信し有効時間の一貫性を保つ方式と比較すると，

- 本方式ではバージョン木の葉節点のノードのみを配信するため，サーバ側におけるすべてのバージョンを配信する必要がない.
- 最悪の場合でもすべてのバージョンを配信する方式と配信コストは同じである.

の2点により，我々の管理方式の方が配信コストがかからないと言える.

4.3 クライアント側におけるバージョン管理

4.3.1 バージョンリスト

クライアント側でのバージョン管理は，各ユニットのバージョンリストを生成することで行う．バージョンリストは線形リストであり，ノードが各バージョンを表す (図 10)．クライアントは起動中に，ユニットのバージョン木の葉のバージョンを1回だけ受信することができる.

ユニットのバージョンリスト l は以下のように定義される.

$$l = [c_1, c_2, \ldots, c_n]$$
$$c = (id, version, tr, [v_1, v_2], ar)$$

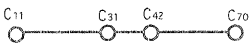


図 10 バージョンリスト

c はユニットの各バージョンである．ここで，id, $version$, トランザクション時間，有効時間はサーバ側のノードと同じものである．ar はそのノードの到着時間である．バージョンリストでは，バージョン番号 (i, j) の i は必ずしも連続しているとは限らない．なぜなら，クライアントがシステムを起動していない間に，受信できないノードがあるからである.

4.3.2 バージョンリストに対する基本操作

クライアント側では，サーバから配信された新しいユニットのバージョンに対して，以下のいずれかの処理を行う．ここで，配信された新しいノードのバージョン番号を (i, j) とする.

- 新しいバージョンリストを作成
  受け取ったノードに該当するバージョンリストが存在しなかった場合，新しいバージョンリストを作成し，ノードをそのリストに追加する.
- バージョンリストの最後尾にノードを追加
  バージョンリストのバージョン番号 (k, l) の k に対して，$i > k$ が成り立つ場合，そのノードをバージョンリストの最後尾に追加する.
- バージョンリストにノードを挿入
  現在のバージョンリストのどのノードのバージョン番号 (k, l) の k に対して，$i \neq k$ かつ $i < k$ である場合は，そのノードをバージョンリストの適切な位置に挿入する.
- バージョンリストのノードを交換
  現在のバージョンリストのノード (バージョン番号 (k, l)) に対して，$i = k$ となるノードがすでに存在する場合は，すでにあるノードと配信されたノードを交換する．交換したノードは，有効時間が正しくないノードなので，破棄する.

図 11 はクライアント側における動作を示している.

(1) サーバ側で c_{10} が作成され，クライアント側がそのノードを受信すると，クライアントはバージョンリストを作成する.

(2) 次に，c_{10} の内容が更新されて c_{20} が作成されるとともに，c_{10} の有効時間が変更された c_{11} が追加される．しかし，このときクライアントはシステムを起動していないため，受信することができない.

(3) c_{20} の内容が更新され，c_{30} が作成される．この

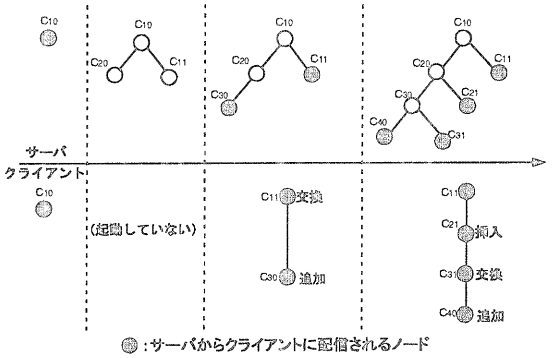


図 11 クライアントにおける処理

ときクライアントがシステムを起動し，サーバ側に送信要求をおくると，サーバはそのときのバージョン木の葉のノードである c_{30} と c_{11} を配信する．クライアントは c_{30} と c_{11} を受け取ると，c_{30} をバージョンリストに追加し，c_{11} を c_{10} と交換する．

(4) サーバ側で c_{30} の内容が更新され，c_{40} が作成されるとともに，c_{20} 及び c_{30} の有効時間を変更した c_{21} 及び c_{31} が作成され，クライアントがそれを受信すると，c_{21} をバージョンリストに挿入し，c_{31} をバージョンリストの c_{30} と交換を行い，さらに新しいバージョンである c_{40} をバージョンリストに追加する．

4.3.3 クライアントにおけるコンテンツの一貫性

クライアント側では，システムを起動していないと受信できない．しかし，サーバ側において有効時間が変更されたノードは，クライアントがシステムを起動したときに，必ずクライアント側に配信される．なぜならば，サーバは，すべての葉節点のノードを配信するためである．クライアント側では，そのノードと配信済みのノードを交換することによって，有効時間が正しいノードのみを蓄積する．従って，受信できないノードはあるが，蓄積されるノードの有効時間は正しい．

ハイパーテキストの配信においては，蓄積されているドキュメント，及びアンカーのすべてのバージョンの有効時間が正しければ，情報の一貫性が保たれる．なぜならば，ドキュメントの有効時間をチェックすることにより，その時刻に価値のあるドキュメントに対してのみのアクセスを許すからである．また，表示したドキュメントの中に含まれるリンクのリンク元とリンク先のドキュメントとアンカーの有効時間をチェックすることにより，関連に矛盾が生じるリンクをのぞき，その時刻に有効なリンクのみを表示する．本バージョン管理方式をドキュメントとアンカーに対して適用することで，クライアントに蓄積されているドキュメントとアンカーのすべてのバージョンの有効時間に矛盾が生じないので，常に正しい情報をユーザに提供することができる．

同期化コンテンツの配信においては，クライアント側に蓄積されている各コンポーネントの各バージョンの有効時間が正しいと，そのコンポーネントのバージョンから成る同期化コンテンツの有効時間も正しい．同期化コンテンツの有効時間は，それぞれのコンポーネントがもっている有効時間や再生時間，再生開始時間が正しくなければいけないが，再生時間は，各バージョンに固有で，変化しない値である．再生開始時間は，SMIL ソースにより求められる値なので，再生時にクライアント側で計算可能である．また，コンポーネントの各バージョンの有効時間は，ここまで述べてきたバージョン管理方式により，常に一貫性が保たれている．よって，同期化コンテンツの有効時間の一貫性も保たれることになる．

従って，ハイパーテキストの配信，同期化コンテンツの配信のどちらも，本バージョン管理方式を用いることで，クライアント側に蓄積されている情報の時間的一貫性が保たれる．

5. おわりに

本論文では，ハイパーメディアの放送型情報配信と，配信情報の時間的一貫性を保つためのバージョン管理方式について述べた．ハイパーメディアの放送型配信の対象となる情報として，ハイパーリンク構造を有するハイパーテキスト，および複数のマルチメディアコンテンツの再生時間の同期を指定した同期化コンテンツである．特に，同期化コンテンツの配信方式として，サーバ側でコンポーネントをあらかじめ同期化し，パッケージ化して配信するパッケージ型配信方式，およびサーバ側でコンポーネントのみを配信し，クライアント側で動的に同期化するコンポーネント型配信方式の２つの方式を提案し，様々な情報の配信に対して適用可能なことを示した．

また，今回提案したバージョン管理方式により，クライアントがシステムを起動していない期間に受信できなかった情報があったとしても，情報の有効時間に関する一貫性が保たれることを示した．

今後の課題として，配信時間や到着時間の導入と，それによる一貫性の精密な制御があげられる．また，クライアント側で配信されるコンポーネントの内容に基づいた動的な同期化コンテンツの生成機能，及び，

情報フィルタリング機能の検討が必要である.

謝辞 筆者の一部（田中）は本研究において，一部，文部省科学研究費重点領域研究「高度データベース」（領域番号 275（08244103））の援助を受けています．また，本研究の一部は，日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「マルチメディア・コンテンツの高次処理の研究」（プロジェクト番号 JSPS-RFTF97P00501）によっています．ここに記して謝意を表すものとします．

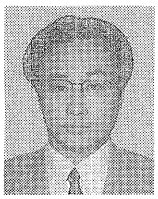
参 考 文 献

1) 角谷和俊:「放送型データベース：放送型情報配信のためのデータ配信モデルとシステム」, ACM SIGMOD 日本支部第 12 回大会講演論文集, pp. 105–150 (1999).
2) PointCast: PointCast Network, *http://www.pointcast.com*.
3) Lemay, L.: *Official Marimba Guide to Castanet*, Sams.net(松田晃一ほか訳：Marimba オフィシャルガイド Castanet, トッパン, 1997) (1997).
4) 角谷和俊, 野田玲子, 田中克己: 放送型ハイパーメディアのための時間依存リンク機構, 電子情報通信学会論文誌 D-I, Vol. J82-D-I, No. 1, pp. 291–302 (1999).
5) SMIL (Synchronized Multimedia Integration Language), *http://www.w3.mag.keio.ac.jp/AudioVideo/*.
6) Hoschka, P.: An Introduction to the Synchronized Multimedia Integration Language, *IEEE Multimedia*, Vol.5, No.4, pp. pp84–88 (1998). http://www.computer.org/multimedia/mu1998/u4toc.htm.
7) 野田玲子, 角谷和俊, 田中克己: 時間依存リンクを用いた情報配信システム Mille-feuille の設計, 電子情報通信学会第 9 回データ工学ワークショップ (DEWS'98) (1998).
8) Sumiya, K., Noda, R. and Tanaka, K.: Hypermedia Broadcasting with Temporal Links, *Proc. of 9th International Conference on Database and Expert Systems Applications (DEXA'98)*, pp. 176–185 (1998).
9) Halasz, R. and Schwartz, M.: The dexter hypertext reference model, *Communications of the ACM*, Vol. vol.37, No. 2, pp. 30–39 (1995).
10) Hardman, L., Bulterman, D. and Rossum, G.: The amsterdam hypermedia model: Adding time and context to the dexter model, *Communications of the ACM*, Vol. vol.37, No. 2, pp. 50–62 (1995).
11) RealSystem G2: Real Networks, *http://www.real.com/*.
12) 川口知昭, 土居明弘, 角谷和俊, 田中克己: ScoopCast: ライブ映像ストリームによる動的番組編集システム, アドバンスト・データベース・シンポジウム'98(ADBS'98), 情報処理学会, pp. 167–174 (1998).

(平成 1999 年 6 月 27 日受付)
(平成 1999 年 9 月 27 日採録)

(担当編集委員 **有川 正俊**)

野田 玲子（学生会員）
1998 神戸大・工・情報知能工学卒，現在，同大学大学院自然科学研究科修士課程在学中．データベース，ハイパーメディアに興味を持つ．



角谷 和俊（正会員）
1986 神戸大・工・計測工学卒，1988 同大学大学院工学研究科修士修了．同年松下電器産業株式会社入社．ソフトウェア開発環境，マルチメディアデータベースの研究開発に従事．1998 神戸大学大学院自然科学研究科 (情報メディア科学専攻) 博士後期課程修了．1999 年 神戸大学都市安全研究センター都市情報システム研究分野 (工学部情報知能工学科) 講師．博士 (工学)．システム制御情報学会，IEEE Computer Society，ACM 各会員．

田中 克己（正会員）
1974 京大・工・情報工学卒，1976 同大学大学院修士修了．1979 神戸大学教養部助手，1986 同大学工学部助教授．1994 同大学工学部教授 (情報知能工学専攻)．1995 同大学大学院自然科学研究科 (現在，情報メディア科学専攻) 専任教授，現在に至る．工学博士．主にデータベースの研究に従事．96 年度より通信・放送機構「次世代デジタル映像通信の研究開発」の研究統括責任者，文部省科研費重点領域研究「分散発展型データベースシステム技術の研究」の研究代表者，神戸マルチメディアインターネット協議会会長，人工知能学会，IEEE Computer Society，ACM 等各会員．