Pull 型サービスにおけるデコーダダウンローダブルシステムの構成及び評価

A study on Architecture of Decoder Downloadable System for pull service

宮澤 敏記 *1　清水 直人 *1　亀山 渉 *2　渡辺 裕 *2　富永 英義 *1*2
Toshinori MIYAZAWA　Naoto SHIMIZU　Wataru KAMEYAMA　Hiroshi WATANABE　Hideyoshi TOMINAGA

*1 早稲田大学 理工学部 電子・情報通信学科　*2 早稲田大学 国際情報通信研究センター

*1 *Dept. of Electronics and Communication Engineering, WASEDA University*
*2 *Global Information and Telecommunication Institute, WASEDA University*

Abstract: In this paper, we propose decoder downloadable system architecture. Currently, applications in telecommunication or broadcast are designed for one single media format. Our proposed system provides an environment that optimum decoders for each media can be downloaded into a media-processing engine by the time it plays the media. Additionally, we evaluate download scheduling employed in our proposed architecture.

1. はじめに

MPEG-2 や MPEG-4 等の標準化により，DCT ベースの画像符号化方式が通信・放送における様々なアプリケーションに用いられることとなった．しかし，DCT による画像符号化方式があらゆるコンテンツに対し最適ではなく，各コンテンツの特性に特化した画像符号化方式 [1] の需要が，今後一層高まって行くと考えられる．一方，システム運用の観点からは，デコーダのバグ修正や最新バージョンへの更新が必要とされており，さらにエフェクト等のメディアに対する追加処理を配信することで，新たなサービス形態が生じる可能性がある．

本研究では，1) 各コンテンツに最適なデコーダ（もしくはエフェクト処理プログラム）を複数のコンテンツを連続再生する過程で動的にダウンロードし，2) ダウンロード済みのデコーダを対象となるコンテンツの再生処理に反映させるためのシステム構成を提案する．加えて，動的なデコーダダウンロードのスケジューリングに関して評価を行ったので報告する．

2. デコーダダウンローダブルシステム構成

2.1 処理の概要

コンテンツを連続再生する制約条件下では，各コンテンツに最適なデコーダを，メディア再生の事前にダウンロードを完了しておく必要がある．デコーダはコンテンツとともに配信するのではなく，コンテンツ情報及びデコーダ情報を抽象化することで (図 1にコンテンツの抽象化情報を示す)，ダウンロード処理を柔軟にスケジューリングすることが可能となる．

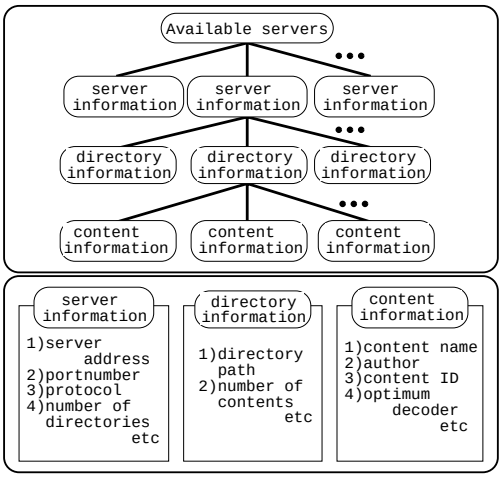

図 1: Content information

クライアントが各コンテンツに最適なデコーダを動的に見つけ，かつダウンロードする処理の流れは次のようになる．1) クライアントシステム内のスケジューラは，適切なタイミングでユーザが指定したコンテンツ情報を解析し，デコーダがクライアント内に存在するか調べる．2) 存在しない場合，コンテンツ情報をサーバに渡す．3) サーバは受信したコンテンツ情報を元に，クライアントに最適なデコーダ情報を検索しクライアントに送信する．4) デコーダ情報には，デコーダファイル名やデコーダ配信サーバアドレスが含まれており，これらの情報を元にクライアントはデコーダ配信サーバからデコーダを受信する．最適なスケジューリングを実現するためには，i) 連続するメディア間の再生待ち時間，ii) ダウンロード済みのデコーダが必要とする総メモリ量の 2 点から検討しなければならない．

2.2 設計方針

提案するクライアントシステムは様々な実行環境を想定しているため, Java による実装が望ましい．システムのプロトタイプを構築するに当り, メディア処理を対象とした JMF (Java Media Framework) なるパッケージが Java に実装済みであるので，これを用いることとした．ただし，メディア処理を完全に Java で実装することは実行時間の観点から議論の余地がある．ゆえに JMF 依存の処理を 1 モジュール内でブラックボックス化することで，今後の移植を容易とするように設計した．

2.3 システム構成

本稿で提案するシステム構成を図 2に示し，各処理モジュール及びインターフェースの役割について説明する．

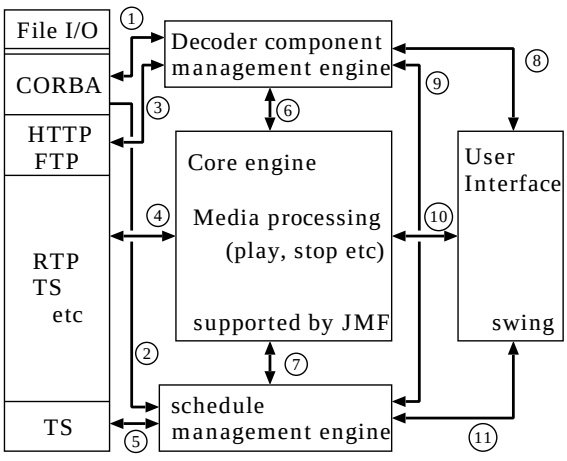

図 2: Downloadable system architecture

2.3.1 基本モジュール

- Core engine
  JMF によるメディア再生処理の実装．各種のトリックモード（早送り，巻き戻し等）も提供される．コーデックはダウンロードされた後，実際に組み込まれる．種々のコーデックを組み込むためには，コーデックが提供するインターフェースを共通化しなければならない．
- Decoder component management engine
  組み込み済みのデコーダの管理，必要デコーダの問い合わせ・ダウンロード及び不要デコーダの廃棄処理の実行．
- Schedule management engine
  メディア再生開始時間，デコーダダウンロード開始時間及びデコーダ廃棄時間等を管理し，対象となるモジュールに処理を指示する．
- User interface
  ユーザ入力に対するインターフェースの実装，再生メディアの表示（本システムでは， Java Swing にて実装）．
- File I/O
  各プロトコルを次の用途に応じて選択する． 1)CORBA：コンテンツ及びデコーダの情報の送受信用， 2)HTTP or FTP：デコーダのバイナリデータダウンロード用， 3)RTP or TS：メディアダウンロード用．

2.3.2 基本インターフェース

- interface 1: デコーダ情報の受信・コンテンツ情報の送信
- interface 2: コンテンツ情報の受信
- interface 3: デコーダ情報の通知・デコーダのバイナリデータダウンロード用
- interface 4: メディアデータ受信用
- interface 5:EPG ダウンロード用
- interface 6: デコーダの確認及び提供
- interface 7: メディア再生・デコーダメモリロード開始指示・状態告知
- interface 8: ユーザによる Plug-in 追加・削除
- interface 9: コンテンツ情報の告知・デコーダダウンロード開始指示・廃棄指示，実装済みデコーダの告知
- interface 10: ユーザ入力 (再生・停止・早送り等)
- interface 11: 視聴可能コンテンツ告知・ユーザによる選択

3. 評価

デコーダダウンロードのスケジューリングを評価するために，1) 蓄積するデコーダ数とメディア再生までの初期遅延時間， 2) 蓄積するデコーダ数と総遅延時間に関して実験を行った． Pull 型サービスにおいて，ユーザは選択したコンテンツ間で恣意的に Jump すると考えられ，ユーザの操作を考慮したコスト換算が必要となる．（実行環境 -CPU:Pentium III 1GHz, RAM:256MB, OS:windows 2000, Java 環境:JDK1.3.1）．

3.1 メディア再生初期遅延の評価

コンテンツ数を 12，全てのコンテンツが 4.25KB のエフェクト処理をダウンロードする必要があると設定し，一番目のメディア再生までの遅延時間を測定した（図 3）．グラフの横軸に蓄積されるデコーダ数を取る．スケジューリング方式は， 1) 既にクライアントにデコーダが存在する場合 (local decoder)， 2) 複数のデコーダを並列にダウンロードする方式 (parallel download)， 3) 複数のデコーダを再生予定順番に従って直列にダウンロードする方式 (serial download) について比較した（対象デコーダダウンロード完了直後，メディア再生プロセス再開）．

結果から複数のデコーダを一括してダウンロードする際，必要なデコーダを優先的にダウンロードする直列方式の方が並列方式よりも再生までの遅延が小さいことが判明した．また直列方式であっても，ダウンロードするデコーダ数が増えるにつれて遅延時間が増加するのは， ftp セッション処理がメディア再生処理のパフォーマンス効率を低下させるためと考えられる．

3.2 総再生遅延コスト評価

ダウンロードに伴う総遅延コストを $T_d(n)$，前節で求めた初期遅延を $d_s(n)$， 1 デコーダダウンロードに伴う遅延を d_d，コンテンツ変更が有った際，新たなデコーダを必要とする確率を $p(k)$，恣意的にユーザがコンテンツ変更する確率を $q(k)$，総コンテンツ数を N，デコーダを必要とするコンテンツ数を M，蓄積するデコーダ数を n とおけば， $T_d(n)$ は次式で表すことができる（ただし， $p(k) = \frac{M-n-k+1}{N-k}$ である）．

$$T_d(n) = d_s(n) + d_d \sum_{k=1}^{M-n} p(k)q(k)$$

q(k) はある分布に従うものと想定されるが，本評価では $q(k)$ の値を 1， 0.5， 0.25， 0 とし，またその他の値は前節に従った． T_d と n の関係を図 4に示す．結果よりユーザによるコンテンツ変更が多い (少ない) 程，蓄積デコーダ数が多い（少ない）方が総遅延が少ないことが分かる．一方，ユーザによる変更の確率が 0.25 である場合，デコーダ蓄積数が 8 で総遅延時間が最小となる．

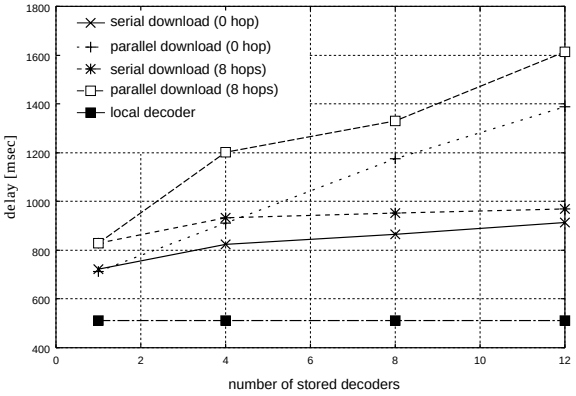

図 3: 蓄積デコーダ数とメディア再生初期遅延の関係

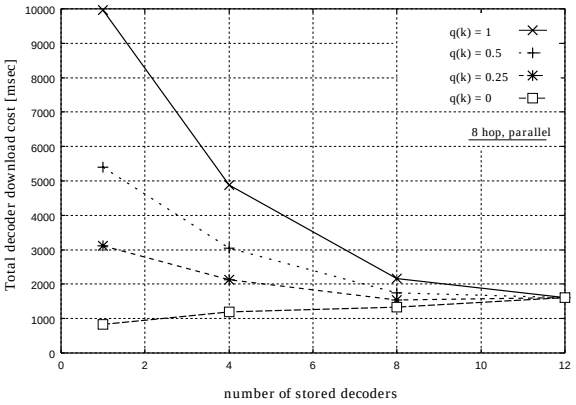

図 4: 蓄積デコーダ数と総遅延コストの関係

4. まとめ

本稿では，デコーダダウンローダブルアーキテクチャ構成を提案するとともに，デコーダダウンロードのためのスケジューリングに関して評価を行った．提案アーキテクチャにより，コンテンツに最適なデコーダを動的にダウンロードし，メディア再生することが可能となった．またスケジューリング評価により，蓄積デコーダ数と総遅延の関係を導くことができた．

参考文献

[1] 宮澤 敏記，亀山 渉，渡辺 裕，阪谷 徹，富永 英義：”アニメーション画像符号化における輪郭線抽出と近似方式の検討”，情処研報， AVM32-12,pp.65-70(2001)

〒 169-0072 東京都新宿区大久保 3-4-1 55 号館 N06-02
早稲田大学理工学部電子・情報通信学科 富永研究室
TEL 03(5286)3385 EXT.3419 FAX 03(5286)3111
e-mail:miyazawa@tom.comm.waseda.ac.jp