インターネット動画転送における複数動画の同期転送に関する研究

Study on Synchronous Transmission of Multiple Video in Webcasting

趙逸恬† 坂井滋和‡
Yitian Zhao† Shigekazu Sakai‡
† 早稲田大学　基幹理工学研究科
‡ 早稲田大学　国際情報通信研究センター
† School of Fundamental Science and Engineering, Waseda University.
‡ Global Information & Telecommunication Institute, Waseda University.

要旨

本研究の目的は Web 上で複数の動画を同期表示するための最適手法を探し出すことである。まず従来手法の分析によりその問題点について検討した。現在はその大半の手法において、HTTP プロトコルが使用されているため、FPS を一致されることができない。そのため、複数の映像完全に同期することが難しい。そこで、WebSocket プロトコルを用いた 3 つの手法を考案した。ここではそれらについて実験を通じて、その効果を検証した。

キーワード：インターネットコンテンツ配信、WebSocket、画像処理、ビデオ処理、CSS mask-image 属性
Keywords: Webcasting, WebSocket, Image Processing, Video Processing, CSS mask-image Attribute

1. はじめに

インターネットは 2000 年代後半から、YouTube、Facebook、Twitter など様々なサービスが登場し、人々のコミュニケーションを高度化した。2010 年以後はさらに進化を続けて、情報収集においてテレビや新聞に代わって、人々に最も利用されるメディアになった。その中で Web 動画の視聴は大きな割合を占めるようなった [1]。こうした流れの中で、2007 年ニコニコ動画が弾幕コメントシステムを開発し、それまで別々であった映像と文字情報による双方向コミュニケーションを一体化させた[2]。弾幕コメントシステムは、映像に視聴者のコメントに連動させ、リアルタイムで画面に表示する。これにより視聴者は動画を見ると同時に、対応するコメントも見られるため、意見や感想の共有が実現できた。しかし、ユーザーが増えるとともに、弾幕の数も増大し、大量の弾幕は動画の主体映像を遮って、ユーザーの視聴体験を悪くする問題が起きた。2018 年、Bilibili[3]は「智能弾幕」という技術で、弾幕を動画の前景と背景の間に通過させることに成功し、ユーザーの視聴体験を改善した。しかし、技術的に見るとこの技術には明らかな欠点がある。それが背景映像と前景映像のずれの問題である。本研究は Bilibili の「智能弾幕」におけるこの問題を、幾つかの代用手法を提案し、実験を通じて、最適な手法を探し出すことを目標とした。

2. Web コンテンツにおける映像同期

2.1. Bilibili 以外のサービス提供者の処理方法

現在主流となっている動画サイトはほとんど弾幕機能が搭載されているが、その中で、弾幕過剰問題の解決を考案したのは Bilibili だけである。本研究の調査結果によると、Bilibili 以外の弾幕サービス提供者は、弾幕過剰に対して弾幕の透明度を調整することで解決を行っている。しかしそれは矛盾しているように見える。弾幕の透明度を高い値に設置すれば動画の主体は見えるが、逆に弾幕が見えない。弾幕の透明度を半分にすれば弾幕と動画主体両方が曖昧になって、両方とも見にくい。従ってこの方法は弾幕過剰問題の解決策にはならない。

2.2. Bilibili の処理方法

Bilibili は動画を格納する Video 要素の上にある弾幕を格納する div 要素の mask-image 属性に mask-frame を設定することで、動画の主体部分を透過させて前景を作る手法をとっている。前処理として元動

画の frame を画像処理し、前景と背景を分離するための白黒二値画像を生成する。その後、二値画像の前景である主体の部分を透明にし、背景を黒にして mask-frame を生成する。こうすることによって、元動画の frame と mask-frame が下図 1 のように一対一対応になる。

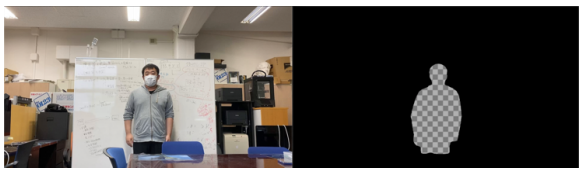

図 1　元動画の frame とそれと対応する mask-frame

動画が再生される時、端末側は動画の時間軸データを獲得し、それをサーバー側に送る。そしてサーバー側は時間軸に合う mask-frame を端末に返す。端末はそれを弾幕 div の mask-image に設定する。1 つの div にとって、mask-image の深さは常に一番手前なので、弾幕は mask-frame の透明区域を遮らない。また mask-frame の透明区域は、元動画の主体と対応しているため、ユーザーの視点からは弾幕が動画主体の後ろを通過するように見える。この手法では HTTP プロトコルの GET メソッドを用いてデータのやり取りを行っている。HTTP プロトコルは Request と Response が必ず 1 組になるため、サーバー側は全ての Request に対して一々処理しなければならない[4]。それはサーバーとクライアント間の遅延を大きくする原因となる。この手法の mask-frame は最大 20fps しか到達することができないが、それは業界標準の 30fps に比べて少ない。加えて回線状態が悪い場合は端末側が GET のメソッドに対する発信の優先度が、Receive 側より高く設定されているため、Request を全部出したにもかかわらず、Response は 1 つも受け取られていない状況が発生する。それが原因で動画の同期において大幅な遅延や機能停止が発生する。この問題に対して、Bilibili は一定の時間内で標準の回線状態に達していないと「智能弾幕」を自動的にオフにするという解決策を出したが、それは解決策とは言えない。

2.3. 本研究の提案手法

通信プロトコルを HTTP から WebSocket に変更する方法はこの問題の解決に有効である。WebSocket では HTTP のようにデータを 1 ブロックことに転送する方法とは異なり、端末とサーバーの間の接続が 1 度確立されたら、通信専用のトンネルが作られる。その後はトンネル内でデータのやり取りを行うため、簡単に双方向通信が実現できる[5]。本研究では、プロトコル変更のほかに複数の手法を開発し、全部で合計 3 方式 4 手法の検討を行った。それぞれ frame-video(FV)方式による時間軸同期法、frame-frame(FF)方式による同期転送法、video-video(VV)方式による前景背景分離法、frame-frame(FF)方式による α チャンネル法と呼ぶことにする。実験によってその効果を検証し、初期化時間、同期率、遅延率を総合的に評価し、最適な手法について検討を行う。

3. 研究内容

3.1. アプローチ

以下に本研究で行った 3 方式について、具体的な説明を記す。

FV 方式による時間軸同期法の原理は、従来手法と同じである。Bilibili の手法に比べて異なる点は、通信プロトコルを WebSocket 変更したことのみである。この手法はインフラ層においてできる限り HTTP のモラールをまねするため従来手法の延長線にある改良である。

FF 方式-同期転送法の原理は元動画とそれと対応する mask-frame を同時転送することによって、同期を行う。サーバー側で前処理を行なって元動画の frame と mask-frame を 1 つの base64 文字列に圧縮する。端末で再生される時にサーバー側に「start」の Request を送信する。サーバー側は開始信号を受け取ったら Timer を起動し、毎秒 30frame で文字列を端末に送る。端末はそれを合図に文字列を解読し、元

動画の frame を div に表示し mask-frame を弾幕 div の mask-image 属性に設定する。

VV 方式-前景背景分離法では、元動画の前景と背景を分離しそれぞれを単独の動画にして同期を行う。前処理を行なって frame 毎に動画の主体部分と他の部分を分離する。その後、主体の部分を透明背景がつく動画に圧縮し、他の部分を主体部分が透明の動画に圧縮する。端末はこの２つの動画を位置とサイズが完全に一致する div に格納し、この２つの動画のメディアコントロールもリンクする。次にもう１つ同じ div を前２つの div 間に挟み込む。これによって、再生ボタンを押す時に、この２つの動画も同時に再生される。

今回の実験では、上記 3 手法のほかに FF 方式-αチャンネル法を加えた。この方法は、RGB の色彩情報に、αデータを加えた 4 チャンネルの画像データを転送し、端末側で画像合成を行う。前処理として、サーバー側は元動画 frame の RGB チャンネルと mask-frame のデータを数列化し、できた MatVector を端末に送る。端末は Opencv.js 画像処理ライブラリが必要となるが、これは事前に配置しておく必要がある。その後、受け取った数列データを解読し、その中の RGB を Opencv の merge 機能で元動画 frame に復元し、αチャンネルデータを直接画像にし、mask-frame として使用する。

3.2. 実験

本実験における設定条件としては、回線速度と CPU 性能の 2 点がある。回線速度は端末とサーバーの間通信の実際速度を意味する。例えば回線速度 1Mbps の場合、端末とサーバーの間の最大アップロード/ダウンロード速度は最大毎秒 1Mb を意味する。動画視聴の際、ユーザーは常に良好なネット環境にいるわけではないので、回線速度は動画転送にとって重要な評価基準だと思われる。本実験では回線速度を 0.5Mbps、1Mbps、2Mbps、5Mbps、10Mbps で設定した。また、CPU 性能は動画に対する端末側の処理能力。動画生成のためには CPU によるレンダリング処理が必要なため、CPU 性能によってレンダリング結果に差が出ると推測される。本実験では CPU のクロックを制限することによって、異なる CPU 性能を実現した。周波数制限をかけない条件のほかに周波数を 4 分の 1 と 6 分の 1 とする条件を設定した。そして回線速度の設定と CPU クロックの制限は Google Chrome 開発者ツールの Performance 機能を利用した。

本実験では、HD 解像度(1920x1080)の 8.40 秒(T_0)分（30fps）の動画を使用した。そして測定結果としては、初期化時間（T_i）、動画 1 プレイ完了時間（T_{v1}）、動画 2 プレイ完了時間（T_{v2}）3 つを設定した。測定精度を保つため、測定は全てツールによる自動測定を使用し、測定回数は 10 回で、最終結果は 10 回測定の平均値で算出した。そして本実験において、結果を評価するための指標としては、同期率(S)と遅延率(L)を使用した。同期率と遅延率の計算方法を下記の式(1)、(2)に示す。

$$S = \frac{Min(T_{v1}, T_{v2})}{Max(T_{v1}, T_{v2})} \tag{1}$$

$$L = 1 - \frac{T_0}{Max(T_{v1}, T_{v2})} \tag{2}$$

4. 実験結果とまとめ

次ページ図 2 は CPU が正常の状態で各手法における初期化時間と回線速度の関係を示している。αチャンネル法以外の手法の初期化時間は少なく、回線速度にもあまり影響を受けないことが示されている。また回線速度の上昇とともに、αチャンネル法の初期化時間も減少する傾向が見られる。ただし減少の量は回線速度の増加に比例するのではなく、変化率が徐々に下がっている。したがって CPU 性能と初期化時間の関係において FV と VV 方式の初期化時間は、CPU 性能に影響を受けないことがわかる。

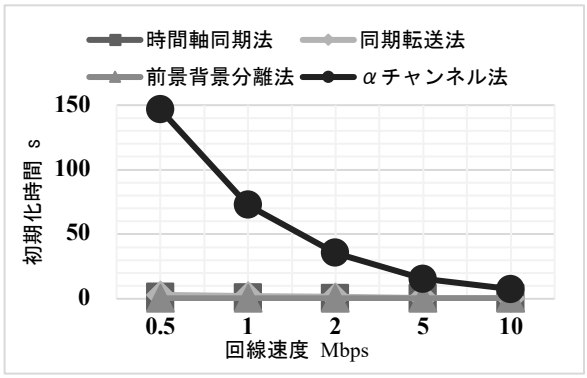

図 2　各手法初期化時間と回線速度の関係

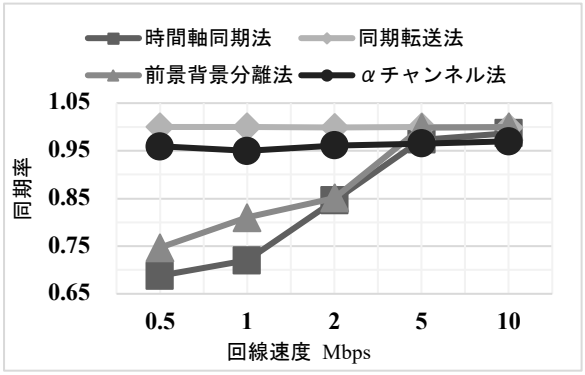

図 3　各手法同期率と回線速度の関係

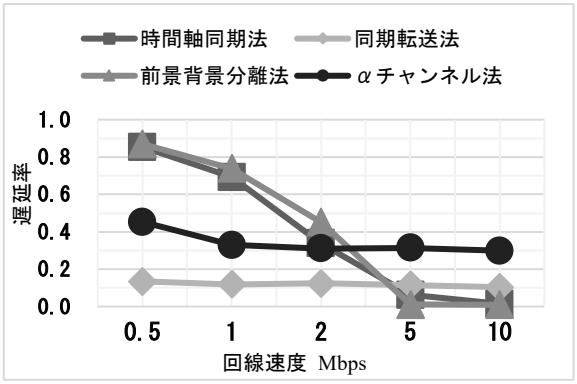

図 4　各手法遅延率と回線速度の関係

左図 3 は CPU が正常の状態で各手法における同期率と回線速度の関係を示している。図からは、FF 方式が回線速度の影響を受けず、常に高い同期率を維持していることがわかる。同じ FF 方式である同期転送法はαチャンネル法より高い同期率を持っており、ほぼ完全同期している。一方、FV 方式と VV 方式の同期率は回線速度の影響を受けやすく、回線速度の上昇とともに増加する。5Mbps を超えたあたりからは、前景背景分離法はほぼ完全同期になる。CPU 性能と同期率の関係は、全ての手法に対して異なる CPU 性能の条件下で同期率の変化は極めて少なく、CPU 性能は手法の同期率に対して影響がないと考えられる。

左図 4 は CPU が正常の状態で各手法における遅延率と回線速度の関係を示している。図に示されたように、FF 方式の遅延率は同期率の場合と同じく回線速度に影響を受けにくいが、同期率と違って理想的な値を保つことができない。たとえ比較的低い同期転送でも 10%以上の遅延が常に発生している。αチャンネル法ではさらに悪く、30%以上の遅延が発生している。それに比べ FV 方式と VV 方式の遅延率は回線速度に影響されやすく、0.5Mbps の時は 80%を超える遅延が発生するが、回線速度の上昇とともに遅延が改善され、10Mbps のとき無遅延の状態になることが示された。CPU 性能と遅延率の関係は、FV と VV 方式の遅延率は CPU 性能に影響を受けず、どの CPU 性能条件においても遅延率はほぼ同じであり、逆に FF 方式の遅延率は CPU の性能に大きな影響を受けることがわかった。

以上の実験結果により、FF 方式は同期率において優位性が見られるが、遅延率が高い欠点を持っている。逆に FV と VV 方式は低い回線速度において、指標が要求に満足できないが、回線速度の上昇とともに、同期率の上昇と同時に遅延率が減少し、最終的にほぼ完全同期かつ無遅延の状態に達することができる。αチャンネル法についてまだ多くの問題が存在しているため、現段階では実用化は難しい。このように、現時点では、どうの方式にも利点と欠点があるため、最適な方法として、1 つを選択することは難しい。利用目的やネットの利用環境によって 3 方式の中から相応しい手法を選択することが求められる。

参考文献

[1] Rena F., “55 video marketing statistics for 2020”, “https://biteable.com/blog/video-marketing-statistics/”, visited on Nov.10th, 2020.
[2] ニコニコ動画, “【6 周年】ニコニコ建国記念日ちょっと昔話” “http://blog.nicovideo.jp/niconews/ni036699.html” , 2020 年 11 月 10 日閲覧.
[3] Bilibili 動画, “弹幕阳光计划第十弹 蒙版听说过吗，弹幕黑科技了解一下？”, “https://www.bilibili.com/read/cv534194” , 2020 年 11 月 10 日閲覧.
[4] W3C, “HTTP - Hypertext Transfer Protocol”, “https://www.w3.org/Protocols/”, visited on Nov.10th, 2020.
[5] W3C, “The WebSocket API - W3C Candidate Recommendation 20 September 2012”, https://www.w3.org/TR/websockets/, visited on Nov.10th, 2020.