大規模 HTTP ライブストリーミング配信における サーバログを用いた視聴遅延の推測手法の提案

二宮 恵 † 長 健二朗 †
†IIJ イノベーションインスティテュート

本研究では、HTTP ライブストリーミングにおける視聴遅延のモデル化を行い、そのモデルに従い Web サーバログからユーザごとの視聴遅延を推測する手法を提案する。提案手法を用いて、夏の甲子園で行なわれた HTTP ライブストリー ミング配信の Web サーバログを用いて視聴遅延の推測を行なった。分析結果から、セグメントの長さとクライアントが視聴開始時においてバッファリングするセグメント数が、視聴遅延の長さを決める大きな要因となることを示した。そして、各視聴ごとの視聴遅延のばらつきはの幅が、視聴遅延の平均を中心にして ±2 セグメントの範囲に収まることを明らかにした。

How Live is Live Streaming over HTTP? Inferring Playback Delay from Server Logs

Megumi Ninomiya† Kenjiro Cho†
†IIJ Innovation Institute Inc.

In this paper, we propose a method to infer the playback delays for each client only from Web server logs, in order to look into playback delays for the entire users. By applying the proposed method to the access logs of a large-scale live event, we have captured the distribution of playback delays. From the results, we have identified the two major factors of playback delays: the selection of segment length, and the startup buffering on the client side. We find that the vast majority of playback delays falls into the range: mean plus-or-minus 2 segment-lengths.

1 はじめに

近年のライブ及びオンデマンドのストリーミングサービスでは、Web サーバを利用した HTTP によるストリーミング配信が一般的になっている。HTTP を用いたストリーミング配信技術として、HLS(HTTP Live Streaming)、HDS(HTTP Dynamic Streaming) や MPEG-DASH などがある。ストリーミングサービスの中でも、大規模なスポーツイベントのライブストリーミングは人気が高く多くの視聴者を集めている。2012 年のロンドンオリンピックにおいては、配信時間と視聴回数の両方でライブストリーミングがオンデマンドを上回ったとの報告が発表されている [1]。

HTTP によるライブストリーミングでは、セグメントファイルによるビデオ配信を行なうために、それまでの RTSP(Real-Time Streaming Protocol) や RTP(Real-Time Transport Protocol) に比べて遅れやばらつきが発生する。実際、HTTP ライブストリーミング配信では、TV 中継と比べて数十秒の遅れがあると指摘されている。数十秒という時間の長さは、スポーツ中継などでは重要なシーンの視聴が丸ごと遅れる場合もあり、ユーザの視聴体験に大きな影響を与える。また、HTTP ストリーミングでは視聴の開始タイミングはクライアント側が決めており、サーバ側では制御できない。そのため、複数のクライアントが同時刻に同じライブストリーミングを再生していても、それぞれのクライアントで再生されている映像の時刻にはばらつきがある。

しかし、実際の HTTP ライブストリーミング配信における視聴者ごとの視聴遅延の程度と、視聴者間の遅延のばらつき具合について、統計的な数値はこれまでほとんど報告されていない。それは、これらの遅延を測定するためには、コンテンツが撮影されてからユーザのプレーヤーで再生されるまでの包括的な計測が必要となり、コンテンツホルダー、サービスプロバイダー、視聴ユーザなど複数のステークスホルダーに跨った計測システムの構築はコストや運用面でのハードルが高いからである。HTTP ライブストリーミングサービスの品質を考える上で視聴遅延は重要な要素の一つであるため、低コストで視聴遅延を推測できるならば、サービス品質を判断する上で有効である。そこで、HTTP ライブストリーミングにおける視聴遅延のモデル化を行い、そのモデルに従い Web サーバログからユーザごとの視聴遅延を推測する手法を提案する。提案手法により、日本の主要なスポーツイベントである、夏の甲

子園で行なわれたライブストリーミング配信のサーバログを用いて視聴遅延の推測を行なった。

結果から HTTP ライブストリーミングにおいて、セグメントの長さとクライアントが視聴開始時においてバッファリングするセグメント数が、視聴遅延の長さを決める大きな要因となることを示した。さらに、各視聴ごとの視聴遅延のばらつきはの幅が、視聴遅延の平均を中心にして ±2 セグメントの範囲に収まることを明らかにした。

2 HTTP ライブストリーミングにおける視聴の遅れ

2.1 ライブストリーミングにおける視聴遅延の要因

HTTP ライブストリーミングにおける視聴遅延の要因として、エンコーダーによる映像のエンコーディング時間、セグメントファイルの生成時間、Web サーバを含む各サーバへのアップロード時間と、クライアント側のダウンロード時間やデコード時間が挙げられる。視聴遅延がクライアントごとに揺らぐ要因として、HTTP ストリーミングにおける視聴開始タイミングが、クライアント側で個々に決められていることが挙げられる。我々は、ここで挙げた要因以外に、HTTP ライブストリーミングにおけるセグメントの長さと、視聴開始時のバッファリングが視聴遅延の要因となることに着目した。

2.2 視聴開始時のバッファリングが視聴遅延に与える影響

ストリーミングの視聴をよりスムーズに行うために、クライアントはバッファリングを行う。視聴開始時のバッファリングが視聴遅延に与える影響は、オンデマンドストリーミングとライブストリーミングで異なる。

プログレッシブダウンロードなどのオンデマンドによる再生では、ビデオの先頭から再生を始めるため、先頭フレームから読み込みを始めてバッファが埋まったら再生を開始する。このため視聴始時のバッファリングは、クライアントの視聴開始待ちの時間の長さに影響を与える。

一方、ライブストリーミングでは、クライアントが再生を開始したタイミングで取得可能かつ最新のフレームの中で、バッファ量分遡ったフレームから読み込みを始め即座に再生を開始する。このため、視聴開始時のバッファリングは、オンデマンドの場合と異なり、クライアントの視聴開始待ちの時間への影響は小さくなるが、バッファ量分の再生時刻の遡りを起こすため、これがライブストリーミングにおける視聴遅延の大きな要因となる。HLS では 10 秒のセグメント長が推奨されており、通常のプレーヤーはトリプルバッファ(one for playing, one for downloading,one for extra) 分のバッファを必要としていると言われている。この場合約 30 秒分の再生開始時刻の遡りを引き起こすとになり、その結果として 30 秒の視聴遅延が発生することになる。

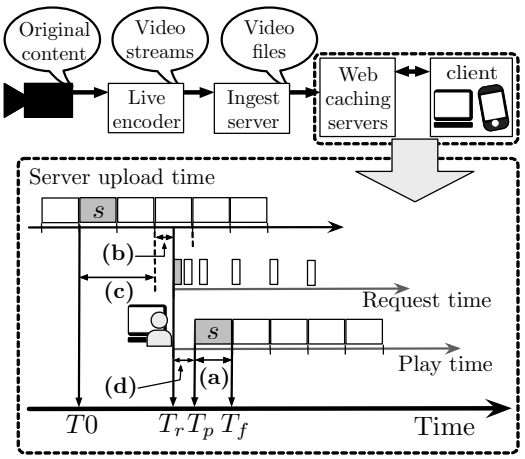

図 1 視聴遅延のモデル化

3 視聴遅延のモデル化

HTTP ライブストリーミングにおいて、オリジナルコンテンツがクライアントで再生されるまでの流れを図 1 に示す。ここではセグメントファイルが Web サーバにアップロードされたあとの視聴遅延について、遅延が起きる要因とその要因ごとの遅延時間の長さに基づいて視聴遅延をモデル化する。

図 1 において、Server upload time は各セグメントファイルが Web サーバ上にアップロードされた時刻である。各セグメントはセグメントファイルの長さ分の間隔でアップロードされる。Request time はクライアントが各セグメントファイルをリクエストした時刻である。Play time はクライアントにおいて各セグメントが再生された時刻である。各セグメントはセグメントファイルの長さ分の間隔で再生が開始される。セグメント S が Web サーバにアップロードされた時刻を T0、クライアントがセグメント S をリクエストした時刻を Tr、セグメント S の再生を開始した時刻を Tp、セグメント S の再生が完了した時刻を Tf とする。

Web サーバにセグメントファイルがアップロードされてから、クライアントでセグメントファイルの再生が完了するまでにかかる時間を、ここで扱う視聴遅延全体の長さとする。

視聴遅延を要因ごとに整理し、各要因ごとの遅延の長さを表す。(a) は、1 セグメント分の再生にかかる時間であ

る。HTTP ストリーミングにおいてビデオストリームがセグメント化されていることによる遅れである。(b) は、クライアントのリクエストタイミングの揺らぎによる遅れであり、この遅れの平均はセグメントの長さの 1/2 となるため、(b) による遅延の長さを 0.5 セグメント分とする。(c) は、視聴開始時のバッファリングにより再生時刻の遡りが起きることによる遅れであり、バッファリングされたセグメント数分のセグメントファイルの長さとなる。(d) は、セグメントファイルのダウンロードとデコード待ちによる遅れである。

セグメントファイルの長さを L、(c) でバッファリングされるセグメント数を N、(d) の待ち時間を w とすると視聴遅延は次のように表せる。式内の定数 1.5 は、要因 (a) と (b) の遅延によるセグメント数を表している。

$$\text{playback delay} = (1.5 + \mathrm{N}) \times \mathrm{L} + \mathrm{w} \qquad (1)$$

4 視聴遅延の推測手法

4.1 夏の甲子園のライブストリーミング配信

朝日放送 [2] が取り組んでいる夏の甲子園の Web 展開において、IIJ[3] はライブストリーミング配信を提供している [4]。夏の甲子園は毎年 8 月に開催される日本の高校野球大会であり、入場者数は毎年 80 万人前後を動員する学生スポーツ及び国内アマチュアスポーツ最大の大会である [5]。2014 年 8 月に行われた夏の甲子園におけるライブストリーミング配信では、決勝戦においてピークトラフィック 108Gbps を記録し、2 週間の大会期間の総リクエスト数は約 19 億、ユニーク IP address 数は 130 万となった [6]。表 1 にライブストリーミング配信のアクセス規模の概要をまとめる。

2014 年の夏の甲子園では 38 台の Web サーバ (nginx) を用いて試合中継のライブ配信が行われた。中継映像は朝日放送でエンコードされ RTMP(Real Time Messaging Protocol) で IIJ のインジェストサーバにアップロードされる。インジェストサーバでは PC 向けに HDS(HTTP Dynamic Streaming) とモバイルデバイス向けに HLS(HTTP Live Streaming) の 2 種類のコンテンツを生成した。今回のライブストリーミングでは、PC とモバイルデバイスの両方で Adaptive biterate 用に 450kbps と 750kbps の 2 種類の bitrate によるコンテンツが用意された。PC 向けの HDS コンテンツは、ブラウザを利用して朝日放送の特設 Web サイトから視聴が可能であり、モバイルデバイス向けには iOS と Android 用に専用アプリケーションが提供され、アプリから HLS コンテンツの視聴が可能であった。モバイルデバイスに向けた試合中継のライブ配信は 2014 年が初めての試みであったが、ユニーク IP アドレスのうち 55% はモバイルデバイスからという結果となり、多くのユーザがモバイルデバイスからストリーミングを視聴していたことが分かった [6]。

表 1 2014 年夏の甲子園ライブストリーミング配信におけるアクセス規模の概要

Period of time (days)	# Log entries (billions)	Sent data (TB)	# TCP connections (millions)	# Unique IPs (millions)
14	1.9	531.4	281.0	1.3

4.2 Dataset の用意

視聴遅延を推測するために、大会期間中の全てのサーバログを用いて 2 種類のデータセットを作成した。1 つ目は各ユーザの視聴ごとのリクエスト列で、2 つ目は視聴ごとのリクエストの遅れを測るためのセグメントごとの基準時刻である。

今回のストリーミング配信ではリクエストごとに表 2 の項目がログに記録された。HTTP ライブストリーミングでは、クライアント（ブラウザやスマートフォンの専用アプリケーション) は、プレイリストとセグメントファイルの 2 種類のファイルに対して、HTTP によるダウンロードを繰り返しながら再生を行っている。プレイリストにはその時点でダウンロード可能なセグメントファイルのリストが書かれており、セグメントファイルにはエンコードされたビデオストリームを一定時間ごとに分割して生成された動画データが含まれている。長い動画ファイルの再生と違い、HTTP ライブストリーミングではクライアントは随時更新されるプレイリストに対してリクエストを繰り返し、新たに取得したプレイリストを参照しながら次々にセグメントファイルへのリクエストを行う。

そのため Web サーバのログにはリクエストされたプレイリストとセグメントファイルごとのログが記録される。

使用されたログフォーマットのうち、一般的に利用されている Apache の Common Log Format や Combined Log Format と異なる項目は cache status、response time、connection id および request id である。表 2 の timestamp は、Web サーバにおいてクライアントからのリクエスト処理が完了した時刻を秒単位で示す。cache status は、リクエストされたコンテンツがキャッシュに存在したかどうかを示す。HTTP Request は、HTTP method と URL および HTTP プロトコルを含んでいる。URL にはクライアントのデバイスタイプを識別する文字列が含まれている。リクエストされたコンテンツがセグメントファイルであった場合には、URL はセグメント番号

表 2 Server log format

1	timestamp
2	cache status
3	client IP address
4	client port
5	server IP address
6	HTTP Request
7	response satus code
8	content body size
9	response time
10	connection id
11	request id

を含む。セグメント番号は順番に 1 つずつインクリメントされた番号である。content body size は、転送されたコンテンツサイズを示す。response time は、Web サーバにおいてリクエストの処理にかかった時間をミリ秒単位で示す。connection id は、リクエストされたコンテンツの転送が行われた TCP コネクションの ID を示し、request id は同一 TCP コネクション ID におけるリクエストの順番を示す。ログから各ユーザの視聴ごとのリクエスト列とセグメントごとの基準時刻を求める際に、サーバがクライアントからのリクエストを受け付けた時刻をリクエスト時刻として用いた。リクエスト時刻は、各ログのエントリごとに timestamp から response time を引いた時刻とした。

4.2.1 視聴ごとのリクエスト列

各ユーザの視聴ごとのリクエスト列は次のようにして取り出した。まず、サーバログの各リクエストを client IP address とデバイスタイプとで分類した上で、次に、分類したリクエスト群の中からセグメント番号が連続している箇所をリクエスト列として取り出した。取り出したリクエスト列から、視聴として扱うには短すぎるリクエスト列を排除し、残ったリクエスト列の中で視聴の開始を含んでいると考えられるリクエスト列を取り出して、視聴遅延の推測に用いた。ここではリクエスト列の最小リクエスト数を 5 とした。これは約 40 秒分の視聴に当たる。また、取り出したリクエスト列が視聴の開始を含んでいるようにするために、各リクエスト列の前 10 セグメントに対してリクエストのないリクエスト列を扱う。今回のログフォーマットには User-Agent などのクライアントの識別が可能な項目が含まれていないため、NAT やプロキシを介して同じデバイスタイプの複数のクライアントがいる場合、これらのクライアントごとにリクエストを区別することはできない。

抽出した視聴ごとのリクエスト列の概要を表 3 にまとめる。抽出できた視聴ごとのリクエスト列の本数は、PC が 210 万本で、モバイルデバイスが 80 万本であった。この各視聴に含まれるセグメントファイルのリクエスト数は PC が 1.5 億、モバイルデバイスが 0.3 億となり、サーバログ全体に占める割合としては PC で 20%、モバイルデバイスで 9.6% となった。リクエスト数が 5 以上のリクエスト列の本数は PC で 780 万本、モバイルデバイスで 550 万本あったが、視聴開始を含むリクエスト列に限定したために、対象とするリクエスト列の本数は減っている。

表 3 サーバログから抽出した視聴の概要

	PC	Mobile
視聴の開始を含む視聴数 (万)	210	80
セグメントファイルリクエスト数 (億)	1.5	0.3
サーバログに占める割合	20.0%	9.6%
視聴ごとのリクエスト数の平均	76.2	30.3

4.2.2 基準時刻

視聴ごとに取り出したリクエスト列に含まれる各リクエストのリクエスト時刻が、どれくらい遅れているのかを測るために、各セグメントファイルごとの基準時刻のデータセットをサーバログから作成した。各セグメントファイルの Web サーバへのアップロード時刻は記録されていないため、ここでは各セグメントファイルについて、全ての視聴の中で同じセグメントファイルがリクエストされた時刻を全て取り出し、それらのリクエスト時刻の中で一番最初にリクエストされた時刻を、そのセグメントファイルの基準時刻とすることにした。大会期間中の試合が開催されている時間内において、連続するセグメントファイルの基準時刻の差分時間は、PC と Mobile の両方について平均で 8.0 秒となり、今回のセグメントファイルの長さ (8 秒) と一致した。このことから、各セグメントファイルの最初のリクエスト時刻を基準時刻として扱うことができるものとした。

4.3 視聴遅延の推測方法

今回提案する手法では、視聴において再生される各セグメントごとの視聴遅延を推測する。そのために、視聴における各セグメントの再生完了時刻を見積もり、この再生完了時刻とセグメントファイルの基準時刻との差を、各セグメントにおける視聴遅延とする。

各セグメントごとの視聴遅延に含まれる時間成分を図 2 のように、Δreq_delay、$\Delta download$、Δbuf と Δseg_len の 4 つの時間に分け、T0 をあるセグメントの基準時刻、Tr をセグメントのリクエスト時刻、Td をセグメントのダウンロード完了時刻、Tp をセグメントの再生開始時刻、そ

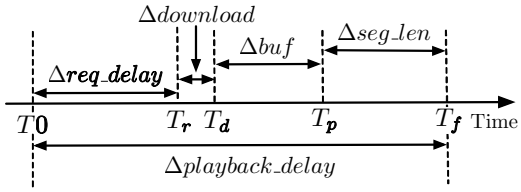

図 2 各セグメントごとの視聴遅延に含まれる時間成分

して Tf をセグメントの再生完了時刻とする。Δreq_delay は、各セグメントのリクエスト時刻における基準時刻からの遅れを表しており、その中には、リクエストタイミングの揺らぎによる遅れと、視聴開始時のバッファリングに起因する再生時刻の遡りによる遅れが含まれている。$\Delta download$ は、クライアントがリクエストを送信してから、セグメントのダウンロードを完了するまでの時間である。Δbuf はこのセグメントの再生が開始される前に残っているバッファの長さ表しており、Δbuf の値はこのセグメントの再生開始前に決まる。Δseg_len は、各セグメントファイルの再生にかかる時間である。今回の配信におけるセグメントファイルの長さは 8 秒であったので、Δseg_len は 8 秒となる。

各セグメントの再生完了時刻を見積もるためには、最初のセグメントの再生開始時刻が必要になる。[7] では、HLS によるライブストリーミングにおいて 80% のクライアントが再生ボタンを押してから、10 秒以内に再生を開始していると報告している。この視聴開始時のセグメントのダウンロードとデコードにかかる時間はクライアントの実装によって異なる。ここでは、Web サーバのログから推測可能な遅延の最小値を求めることとし、最初のセグメントのダウンロードが完了した直後に最初のセグメントの再生を開始するものとして視聴遅延の推測を行なった。

Algorithm 1 では、視聴開始後の 1 番目から N 番目の各セグメントの Δbuf を計算し、それを元にセグメントの視聴遅延 ($\Delta playback\_delay$) を計算している。1 番目のセグメントファイルの再生時にはそれ以前のセグメントによるバッファがないため Δbuf を 0 とする。2 番目以降のセグメントの Δbuf は、前のセグメントの再生完了前に次のリクエストのダウンロードが完了するかどうかで値の設定方法が変わる。再生完了前にダウンロードが完了している場合は Δbuf に前のセグメントの残っているバッファ時間を設定する。一方、再生完了前にダウンロードが完了しなかった場合は、バッファが足りなくなったものとして Δbuf を 0 にリセットする。異なるビットレートのセグメントファイルを取得している場合など、1 つの視聴の中で、同じセグメント番号を持つセグメントファイルを 2 回以上ダウンロードしている場合は、最初のダウンロードの完了時刻を用いて Δbuf の計算を行った。

Algorithm 1 N 番目のセグメントを再生するときの Δbuf と $\Delta playback\_delay$ の計算方法

```
N ⇐ 1
while N ≤ End of request per view do
  if N ≡ 1 then
    Δbuf ⇐ 0
  else
    if Tf(prev_s) < Td(s) then
      Δbuf ⇐ 0
    else
      Δbuf ⇐ Tf(prev_s) − Td(s)
    end if
  end if
  Δplayback_delay
        ⇐ Td(s) + Δbuf + Δseg_len − T0(s)
  N ⇐ N + 1
  prev_s ⇐ s
  s ⇐ s + 1
end while
```

5 サーバログ解析結果

5.1 視聴開始時のバッファリングの動き

視聴開始時のバッファリングの振る舞いを明らかにするために、各視聴ごとのリクエスト列を用いて、先頭から 10 番目までの各リクエストにおけるリクエスト遅延を計算する。リクエスト遅延は、図 2 における Δreq_delay と同じであり、基準時刻とリクエスト時刻との差とする。リクエスト遅延には、リクエストタイミングの揺らぎによる遅れと、視聴開始時のバッファリングに起因する再生時刻の遡りによる遅れが含まれている。

先頭から 10 セグメント目までの PC とモバイルデバイスのリクエスト遅延 (図 3) において、PC もモバイルデバイスも先頭セグメントのリクエスト遅延が最も長く、その遅延の長さが 8 秒以上であることから、1 番目にリクエストされたセグメントファイルは最新のセグメンファイルではなく、それよりも古いセグメントファイルであることが分かる。先頭セグメントのリクエスト遅延は PC で平均 11.4 秒、モバイルデバイスでは 24.0 秒であり、セグメントの長さが 8 秒であったことから、視聴開始時にバッファリングされたセグメント数は PC で 1、モバイルデバイスでは 2〜3 と推測される。

モバイルデバイスでは視聴開始時に PC よりも多くのセグメント数をバッファリングしている。これは、デバイスによる処理能力と利用するネットワーク環境の違いにより、モバイルデバイスではバッファを多く必要とするため

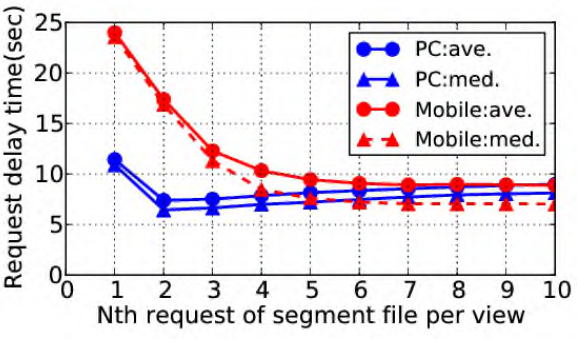

図 3　視聴開始から 10 セグメント目までの各セグメントごとのリクエスト遅延

に、クライアントの実装が異なっているからだと考えられる。また、先頭から 10 セグメント目までのリクエスト間隔は、PC では 1〜2 番目、モバイルデバイスでは 1〜4 番目のリクエスト間隔が短くなっており、視開聴始時に短い時間間隔でバッファリングのためのセグメントファイルのダウンロードが行われていたことを示している。

PC から視聴する場合は通常のブラウザを利用しているため、バッファリングの挙動は今回のライブストリーミングに特化したものではなく、一般的な挙動であると考えられる。一方、モバイルデバイスは専用アプリケーションからの視聴を提供しているため、アプリケーションの設定、実装によってバッファリングの挙動は、今回のストリーミングに特有の可能性がある。今回の解析で得られたバッファリングの挙動が一般的な動きかどうかを判断するためには、今後異なる設定および実装による HTTP ライブストリーミングの解析を行う必要がある。

また、今回提案したモデルおよび解析では、セグメント数によりバッファ分量を定めたが、プレーヤーの実装によってはバッファ分量を再生可能時間やデータサイズによって定義していることも考えられる。その様な場合、セグメント長の長さやサイズによってバッファリングされるセグメントファイル数が変わると考えられるが、これらのバッファ分量についても今後解析が必要である。

5.2　PC とモバイルの視聴遅延の分布

提案手法によりサーバログを用いた視聴遅延の推測から得られた、PC とモバイルデバイスの視聴遅延の平均は、それぞれ 22.7 秒と 32.4 秒であった。これは式 1 で求められる視聴遅延とほぼ一致する。視聴開始時のバッファリングセグメント数 N を PC で 1、モバイルデバイスで 2.5 とし、セグメントの長さ L を 8 秒とし、w をサーバログから得られた平均レスポンスタイムを用い、PC は 0.3 秒、モバイルデバイスは 0.6 秒として計算すると、モデルから

表 4　PC とモバイルデバイスの視聴遅延の推測値とモデル値

	PC	Mobile
提案手法で推測された視聴遅延の平均 (sec)	22.7	32.4
モデルから求めた視聴遅延 (sec)	20.3	32.6
平均レスポンスタイム (sec)	0.3	0.6

表 5　PC とモバイルデバイスの視聴遅延の分布の幅

	PC	Mobile
平均から ±1 セグメント内の割合	88.0%	85.4%
平均から ±2 セグメント内の割合	99.3%	98.4%

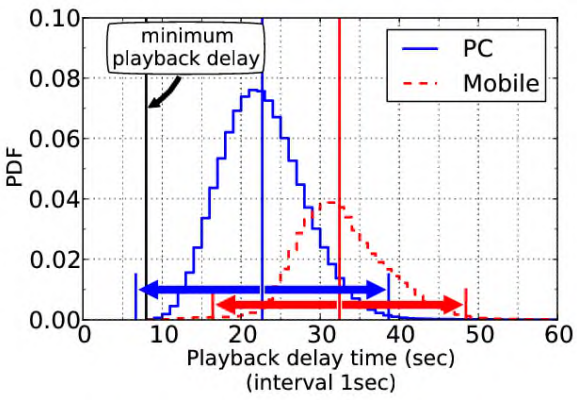

図 4　PC とモバイルデバイスの視聴遅延の分布

求められる視聴遅延の長さは、PC が 20.3 秒、モバイルデバイスが 32.6 秒となる (表 4)。この結果は、視聴開始時のバッファリングによる再生時刻の遡りによる遅れと、1 セグメント分の再生時間による遅れが、視聴全体における視聴遅延の大きな要因となることを示している。

視聴遅延の分布は、PC もモバイルデバイスも平均を中心とした左右対称の正規分布に近い形状をしており、PC でもモバイルデバイスでも、ほぼ全ての分布が視聴遅延の平均から ±2 セグメントの範囲に入ることが明らかとなった (表 5)。

提案手法を用いて計算した PC とモバイルデバイスの視聴遅延の分布を図 4 に示す。それぞれの平均から ±2 セグメントの範囲を矢印で表す。この分布において、視聴遅延の最小値は 1 セグメント分の再生時間で、ここでは 8 秒である。視聴遅延が最小値となるのは、最新のセグメントファイルが Web サーバにアップロードされた直後に、クライアントがそのセグメントファイルをダウンロードし、バッファリングのセグメントを持たずに再生を開始した場合と考えられる。

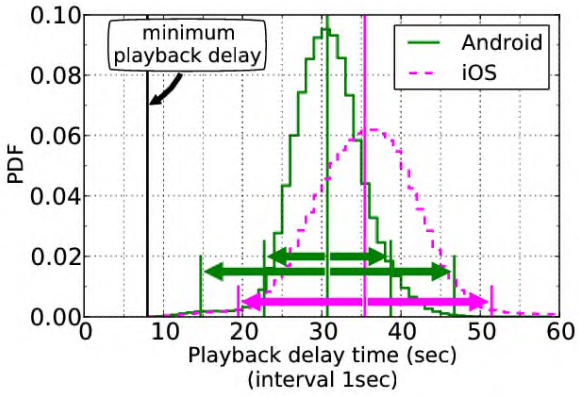

図 5　iOS と Android の視聴遅延の分布

5.3　Android と iOS の視聴遅延の分布

今回のライブストリーミング配信では Android と iOS の 2 種類のモバイルデバイス向けに専用アプリケーションが提供されていたが、配信設定の都合上それぞれのデバイスごとにプレイリスト上のセグメントの数が異なっていた。Android 向けのプレイリストの長さは常に 3 セグメント分となっており、一方の iOS のプレイリストにはその日に生成されたセグメントが追加されていき、最終的に 1 日分のセグメントがプレイリストに含まれていた。そのため、Android では視聴開始時に最大 3 セグメントしかバッファリングできないが、iOS ではバッファリングするセグメント数に制限はなくクライアントの実装に合わせたバッファリングが可能であった。

提案手法によりサーバログを用いた視聴遅延の推測から得られた Android の視聴遅延の平均は 30.7 秒となり、iOS の平均は 35.5 となった。デバイス別に分けた場合でも、推測された視聴遅延の平均は、モデルから求めた視聴遅延とほぼ一致している (表 6)。

Android と iOS との違いを比べると、プレイリストのセグメント数に制限のあった Android の方が制限のなかった iOS よりも聴遅延の長さが 4.8 秒短くなっている。また、Android と iOS の先頭セグメントのリクエスト遅延から視聴開始時にバッファリングするセグメント数を推測すると、それぞれ 2 セグメントと 3 セグメントとなり (表 6)、制限のあった Android の方が視聴開始時にバッファリングされるセグメント数が少ないという結果になった。さらに、視聴遅延は iOS よりも Android の方が狭い範囲に分布しており、視聴遅延のばらつきも Android の方が少ないことが明らかになった。Android の場合、視聴遅延の平均から ±1 セグメントの範囲に 92.6% が入っており、iOS では視聴遅延の平均から ±1 セグメントの範囲に入っているのは 80.1% であった (表 7)。Android と iOS の視聴遅延の分布を (図 5) に示す。

表 6　Android と iOS の視聴遅延の推測値とモデル値

	Android	iOS
提案手法で推測された 視聴遅延の平均 (sec)	30.7	35.5
モデルから求めた 視聴遅延 (sec)	28.7	36.6
先頭セグメントの リクエスト遅延 (sec)	21.4	26.2
バッファリングセグメント数	2	3
平均レスポンスタイム (sec)	0.7	0.6

表 7　Android と iOS の視聴遅延の分布の幅

	Android	iOS
平均から ±1 セグメント内の割合	92.6%	80.1%
平均から ±2 セグメント内の割合	99.3%	97.7%

PC とモバイルデバイスで示された結果と同様に、Android と iOS も視聴遅延の平均から ±2 セグメントの範囲にほぼ全ての分布が入ることが示された。Android の平均から ±1 と ±2 セグメントの範囲と、iOS の平均から ±2 セグメントの範囲を矢印で表す。

今回のライブストリーミング配信では、視聴遅延とは無関係にプレイリストのセグメント数が決められていたが、この結果から、サーバ側でプレイリストのセグメント数を少なくすることで、視聴遅延のばらつきを小さくし、視聴遅延を短く出来る可能性が示された。

6　関連研究

これまでにスポーツイベントやコマーシャルサイトにおいて、HTTP ライブストリーミングの遅延に関する研究がいくつか行われている [8, 7, 9]。

サーバサイドの研究として Kupka らは、Smooth Streaming を用いたスポーツ中継チャンネルのライブストリーミングサービスにおけるサーバログを分析し、同じセグメントファイルへのリクエストは 3 秒から 10 秒の内にその 90% が収まると報告している [8]。3 秒から 10 秒という時間は今回のサーバログから得られた結果に比べるととても短いが、Smooth Streaming によるストリーミングでは、セグメントファイルの長さは 2 秒に固定されており、バッファリングされるセグメント数を今回と同様に 1 から 3 セグメントとした場合に、報告されている秒数は、提案したモデルから得られる視聴遅延の範囲とほぼ一致する。

また、ここで示されている時間はリクエスト全体を対象として、リクエストされた URL に含まれるセグメント

ファイルの生成時刻とクライアントのダウンロード時刻との差を表しており、分析対象となったリクエストが、視聴のどのタイミングでリクエストされたのかとうい点については考慮されていない。我々は、提案の中でユーザの各視聴ごとにリクエスト列を扱うことで、視聴開始時にクライアントがバッファリングのために短い時間間隔でリクエストを行っていることを明らかにし、このリクエストパターンを踏まえて、視聴開始時のバッファリングと、1 セグメント分の再生完了時間による視聴遅延への影響について考慮した上で、視聴時間の推測モデルの提案を行った。

HLS を用いた mobile 向けの Internet TV サービスにおいて Li らは、クライアントサイドデータからほとんどのユーザでは視聴開始ボタンを押してから再生が開始するまでの時間が 10 秒以内であったと報告している [7]。ここで指摘された遅延はクライアントにおけるセグメントファイルのダウンロードとそのデコードおよびレンダリングにかかるる時間であり、我々が今回推測した視聴遅延と補完し合うと、視聴遅延はさらに長くなることが示唆される。我々の提案手法が正しく視聴遅延を推測しているかどうかの評価を行うためには、クライアント側での視聴遅延の計測を行うことが必要であり、今後の課題であると考えている。

7 まとめと今後の課題

HTTP ライブストリーミング配信における視聴遅延をモデル化し、モデルに基づいて、Web サーバのログのみを用いて計算できる視聴遅延の最小値を推測する手法を提案した。提案手法を用いて、大規模な HTTP ライブストリーミング配信イベントにおける Web サーバのログから視聴遅延の推測を行った。

提案手法によりサーバログから推測された視聴遅延と、モデルから計算された視聴遅延がほぼ同じになることを示した。このことから HTTP ライブストリーミングにおいて視聴開始時にバッファリングされるセグメント数とセグメントの長さから視聴遅延の推測が可能なことが示された。そして、HTTP ライブストリーミングにおいて、セグメントの長さと、クライアントの視聴開始時にバッファリングされるセグメント数が、視聴遅延の長さを決める大きな要因となることを示した。また、各視聴ごとの視聴遅延のばらつきの幅は、視聴遅延の平均を中心にして ± 2 セグメントの範囲に収まることを明らかにした。さらに、プレイリストの長さを短くすることで、視聴遅延を短くし、そのばらつきを小さくできる可能性を示した。

本研究では、Web サーバのログを対象として視聴遅延の推測を行ったが、推測手法の正しさを評価するためには、サーバログだけでなくクライアント側やエンコーダ側での計測を行う必要がある。今後サーバ側以外での視聴遅延の計測を行い、提案手法との比較を行う予定である。また、今回は甲子園のログのみを対象として視聴遅延の推測を行ったが、設定環境の異なる HTTP ライブストリーミングの Web サーバのログに対して、同様の手法が適応できるのかという点についても今後の課題と考えている。

謝辞

本研究において貴重なデータの収集および公開にご協力を頂いた朝日放送株式会社の赤藤倫久様に御礼申し上げます。データ収集にご協力を頂いた IIJ プロダクト本部の岡庭大輔様、山本文治様に感謝致します。

参考文献

[1] C. O ’Riordan. The story of the digital olympics: streams, browsers, most watched, four screens. Technical report, BBC Internet Blog, 2012.

[2] ABC. Asahi broadcasting corporation. **http://http://www.asahi.co.jp**

[3] IIJ. Internet initiative japan inc. **http://www.iij.ad.jp**

[4] 山本文治. 最新ストリーミング技術. Technical Repot Vol.25, Internet Infrastructure Review(IIR), November 2014.

[5] Wikipedia. Japanese high school baseball championship. **http://en.wikipedia.org/wiki/**

[6] 二宮恵. 2014 年夏の甲子園ストリーミング配信アクセスログ解析結果報告. Technical report, IIJ, May 2015.

[7] Y. Li, Y. Zhang, and R. Yuan. Measurement and analysis of a large scale commercial mobile internet tv system. In Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC ’11, pages 209224, New York, NY, USA, 2011. ACM.

[8] T. Kupka, C. Griwodz, P. Halvorsen, D. Johansen, and T. Hovden. Analysis of a real-world http segment streaming case. In Proceedings of the 11th European Conference on Interactive TV and Video, EuroITV ’13, pages 7584, New York, NY, USA, 2013. ACM.

[9] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min. Inside the bird’s nest: Measurements of large-scale live vod from the 2008 olympics. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, IMC ’09, pages 442455, New York, NY, USA, 2009. ACM.