M-030

# Web アプリケーションへのマルチデバイス連携フレームワークの適用と評価
# Application and Evaluation of Multi-Device Cooperation Technology to Web Application

徳永 徹郎† 宮崎 泰彦† 井元 麻衣子† 中茂 睦裕† 市井 亮美† 加藤 晃市†
Tetsuro Tokunaga Yasuhiko Miyazaki Maiko Imoto Mutsuhiro Nakashige Ryoumi Ichii
Koichi Kato

## 1. はじめに

ネットワーク対応デバイスの増加に応じて、ネットワークに接続された複数のデバイスを連携させる技術、サービスが広まりつつある。一方、HTML5[1]による Web アプリケーションが注目されている。利用できる API の幅が広がり、これまでより多機能なアプリケーションを作ることができ、対応ブラウザを備えていれば、異なるハードウェアや OS でも動作するといった特徴がある。そのため、HTML5 の Web アプリケーションによりマルチデバイス連携システムを構築することが有望と考えられる。しかし、従来からの Web アプリケーション開発においては複数のデバイスを取り扱うことを想定していないため、デバイス間の連携機能を新たに開発する必要がある。

著者らは、Web アプリケーションによるマルチデバイス連携システムの開発障壁を低くするために、従来の Web アプリケーション開発スキルだけでもマルチデバイス連携システムを構築できるフレームワークを提案している[2]。そのフレームワークは、1 つのブラウザで動作する JavaScript による Web アプリケーションを、中間サーバが複数のブラウザに分割配信し、複数ブラウザ間の通信機能などを中間サーバが提供することでマルチデバイス連携システムを実現するものである。

本稿では、マルチデバイス連携システムの一例として、観光客向けのデジタルサイネージシステムの開発を通して、提案マルチデバイス連携フレームワークを利用した場合と利用しない場合の開発工数とソースコード量を比較する評価実験を行い、実験結果に関して考察する。

## 2. 既存サービスの問題とアプローチ

### 2.1 既存サービスの問題

#### 2.1.1 コンテンツの種類の制限

現在、実現されているマルチデバイス連携の利用法の 1 つとして、PC や HDD レコーダー内の映像や画像、音楽のコンテンツを別の PC や TV で見るなど、デバイスをまたがったコンテンツの視聴がある。これを実現するための技術規格として UPnP[3]が存在する。日本の家電メーカーの多くは、UPnP をベースとした DLNA[4]ガイドラインに従った実装を TV、HDD レコーダー、NAS、ゲーム機などに行っており、デバイスをまたがった視聴を可能としている。また、アップルはアップルのデバイスのコンテンツプレイヤーで再生しているコンテンツを、他のデバイスでストリーミング再生する AirPlay[5]といった機能で、マルチデバイス連携を実現している。Wi-Fi Alliance によって策定された Miracast[6]では、ディスプレイ出力を 1 対 1 で接続した Wi-Fi 経由で無線化し、別デバイスのディスプレイへ出力するという形式でのマルチデバイス連携を実現する。

しかし、これらの仕様や規格だけでは、実現可能なマルチデバイス連携の利用法は限られる。DLNA と AirPlay では、動画、画像、音楽などのメディアコンテンツの制御のみに限定され、Miracast では元のデバイスのティスプレイ出力の拡張あるいは複製のみに限定される。様々なコンテンツを扱うことができ、様々な機能を備えたデバイスが多く存在する昨今のデバイス環境を活かしきれない。

#### 2.1.2 事業者に閉じた連携

デバイスの入力機能を活用した連携利用法として最近登場してきているのが、スマートフォンを操作することで TV、STB や家電の制御を行う利用法[7][8]である。これはスマートフォンがリモコンの役割を担う利用法である。物理的なボタンだけのリモコンに比べて、ソフトウェアで様々な UI を実現できる。任天堂の Wii U では、ディスプレイを備えたゲームコントローラ Wii U GamePad により、これまでにないゲームの楽しみ方が提供されている。

しかし、TV、STB や家電の制御であっても、インタラクティブなゲームであっても、これらのマルチデバイス連携は単独の事業者の中での連携であり、連携可能なデバイスが限られる。利用者にとっては、対応機器の組み合わせをそろえる必要があったり、限られたデバイスだけの限定的なマルチデバイス連携システムとなったりする。また、サービス提供者や開発者にとっては、パソコンやスマートフォンや Web のように誰もがソフトウェアを開発できる環境ではないため、新たなシステムを構築できない。

#### 2.1.3 デバイスごとのソフトウェア開発が必要

開発者が、多様なデバイスが連携するシステムを構築しようとすると、多様なデバイスで動作するソフトウェアを開発する必要がある。通常、ハードウェアや OS によってプログラムは異なるため、ソフトウェア開発のためのスキルが多く必要となり、開発コストの増大につながる。

このように既存のサービスでは、現在の多様なデバイス環境を十分に生かしたサービスを利用者が享受できない。また、開発者は自由に開発することができずに、開発コストは増大する。

### 2.2 アプローチ

一方で、HTML5 で制作する Web アプリケーションが広まりつつある。HTML5 仕様の策定が進み、ブラウザへの実装も進み、これまで OS 上のアプリケーションでしかできなかったことが、ブラウザ上の Web アプリケーションでも実現できるようになってきている。また、ネットワーク対応デバイスの高性能化でブラウザを備えたデバイスが増加しており、多くのデバイスで Web アプリケーションが実行できるようになってきている。

これらのことから、Web アプリケーションを開発するこ

†日本電信電話（株）NTT サービスエボリューション研究所, NTT Service Evolution Laboratories, NTT Corporation

とで多くのデバイスで動作するアプリケーションを実現することができる。1 種類の Web アプリケーション開発は、数種類のデバイスに対応する開発と比べて、開発コストを抑えることができる。

そのため著者らは、マルチデバイス連携システムの実装形態として、HTML5 を利用した Web アプリケーションは開発者の負担と開発コストの面から有効であると考えた。

しかし、HTML5 の Web アプリケーションでマルチデバイス連携システムを開発する際には、これまでとは異なり、1 つの Web アプリケーションが対象とするブラウザが、複数存在することに対応する必要がある。複数ブラウザに表示された Web アプリケーションが連携し同期をとりながら動くためには、他ブラウザの状況を把握する必要がある。これを既存技術のみを利用して実現しようとすると、Web アプリケーション開発者は P2P 型あるいはサーバクライアント型の構成で、連携や同期に関する通信制御プログラムをブラウザ側やサーバ側に実装する必要がある。このような通信制御プログラムは、通常の Web アプリケーション開発で用いるプログラムとは異なるため、多くの場合、Web アプリケーション開発者が新たに学習し、スキルを身に着けるか、新たに有スキル者を開発メンバーとする必要がある。これにより開発者の負担増加や、開発コストの増大といった問題が生じる。

この問題に対し、著者らは、従来の Web アプリケーション開発者が新たに通信制御プログラムを書かなくても、Web アプリケーションによるマルチデバイス連携システムを開発できることを要件として、マルチデバイス連携システムを実現するフレームワークを提案している。

次章では、提案しているマルチデバイス連携フレームワークについて述べる。

## 3. 提案フレームワークの概要

著者らは、1 つのブラウザで動作する Web アプリケーションを分割して連携させるという方式でマルチデバイス連携システムを実現するフレームワークを提案している。以下、図 1 に示す動作の流れを通して提案フレームワークを説明する。

Web 標準の下で制作する Web アプリケーションは HTML、CSS、JavaScript によって構成され、これらの各ファイルは Web サーバに保持される。従来は、ブラウザが Web サーバに配置されている各ファイルを取得、解釈し描画、JavaScript を実行する。しかし、本フレームワークにおいては、Web サーバとブラウザの間に中間サーバを配置する。

まず、中間サーバが Web サーバより Web アプリケーションの各ファイルを取得し、中間サーバの内部で HTML から DOM ツリーを構築する。そして、ブラウザは中間サーバの URL を指定して本フレームワーク用のファイルを取得する。このファイルによって、中間サーバとブラウザの間に WebSocket の接続を生成し、これを通して中間サーバからブラウザへコンテンツが配信される。コンテンツはあらかじめ指定された通りに、DOM の部分木単位で HTML が分割され部分コンテンツとして配信する。

また、中間サーバは JavaScript の実行環境を持つ。Web アプリケーションの JavaScript は中間サーバ内で実行される。DOM API の操作があった場合には中間サーバ内の

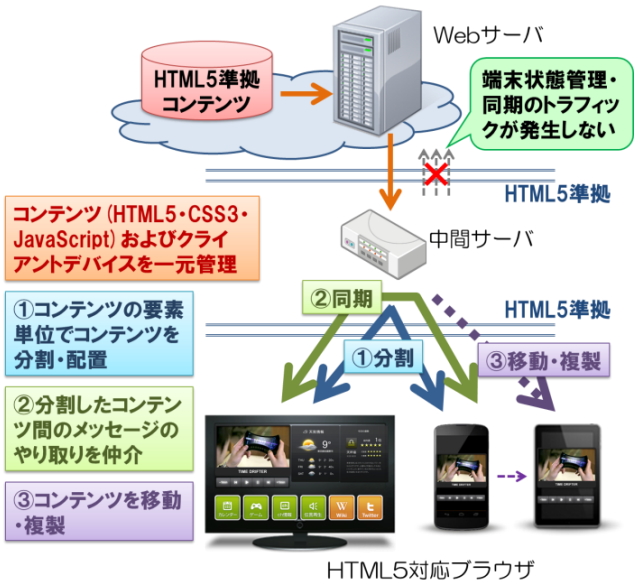


図 1　提案フレームワーク動作の流れ

DOM ツリーに対して操作が行われ、DOM の部分木が配信済みの対応ブラウザにその変更を反映する。

一方、ブラウザでイベントが発生すると、それは中間サーバへ通知される。そのイベントに対応するイベントリスナがある場合には対応する処理が実行される。

部分木は 1 つのブラウザのみに配信されるという制限はないため、複数の異なるブラウザに同一の部分木を送信することができる。これにより、利用者にとっては画面が複製されたように見せることも可能である。

## 4. 評価実験

3 章で述べた提案フレームワークを利用することで、マルチデバイス連携システムを既存技術のみで開発するよりも、開発コストを抑えられるかを検証するために、フレームワーク有無での開発コストとソースコード量の比較検証を行った。被験者として本フレームワークの研究開発に携わっていないソフトウェア開発者 2 人に協力してもらった。

### 4.1 検証用システム

今回の比較検証に用いたシステムは、スマートフォンとの連携機能をもつデジタルサイネージシステムである。マルチデバイス連携サービスの 1 カテゴリとして著者らは大画面と手元の端末の連携サービスがあると考えている。既存の家庭内での TV 操作のように屋外のサイネージ操作サービスが期待できる。

また、検証の際はシステム全体ではなく、マルチデバイス連携に必要なデバイス間の関連付け機能や、同期表示機能、利用者の操作を複数デバイスに反映する機能など、マルチデバイス連携に不可欠な機能の開発を比較対象とした。

このサイネージシステムの主な機能を 4 つ以下に述べる。

(1) コンテンツ再生機能

平常時、デジタルサイネージは、プレイリストに従ってコンテンツをスケジュール通り再生する。

(2) リモコン操作機能

デジタルサイネージ上に二次元バーコードを表示し、ス

マートフォンで読み取ることにより、デジタルサイネージとスマートフォンを関連付ける。関連付けるとスマートフォンに、サイネージと同一のコンテンツが同期した状態で表示され、画面操作によりサイネージ上のスライドショーを前後に移動させることができる。関連付けには二次元バーコードの他に NFC を利用する手段も用意した。

（3）お気に入り機能

関連付けられたスマートフォンで、サイネージに表示されるコンテンツや配信されたコンテンツを随時表示できる。

（4）配信機能

情報配信に同意済みのスマートフォンに時間と場所に応じたコンテンツが配信される。

このデジタルサイネージシステムは、「金沢旅物語スマートインフォメーション」トライアルとして、2013 年 2 月 26 日より 20 日間、JR 西日本金沢駅構内の石川県金沢観光情報センターに設置され、一般の方に利用してもらった。

### 4.2 検証内容

今回、2 つの検証を行った。

（検証 1）

まず初めに、提案フレームワークを利用せずに先述したシステムの設計および実装を行ってもらった。その後、同じ被験者に検証用システムのうち、リモコン操作機能について提案フレームワークを利用して設計および実装を行ってもらった。先に述べたマルチデバイス連携システムに不可欠な機能群が含まれるため、この機能の開発を比較対象とした。

（検証 2）

上記の検証 1 が終わった後に、機能追加を依頼し、提案フレームワークを利用する開発と利用しない開発を実施してもらった。追加した機能は次のような仕様とした。コンテンツが同期して表示されているスマートフォンに表示される「おすすめ」「行きたい」の各ボタンを利用者が押すことで、サイネージの画面に新たに表示されたコンテンツごとの各ボタン押下回数が動的に変化する。

これらの検証 1 、 2 共通でそれぞれの工数の実績値を記録しておいてもらい、作成されたソースコードと合わせて、評価対象とした。

### 4.3 システム構成

図 3 に、提案フレームワークを利用する場合としない場合の、検証用システム全体構成を示す。従来技術のみでの実装の場合、コンテンツサーバの配下にサイネージ端末と利用者のスマートフォンがクライアントとしてつながる形態になる。提案フレームワークを利用した場合は、コンテンツサーバとクライアントとなるブラウザの間に中間サーバが配置される。その他の構成は同一で、コンテンツサーバのサーバサイドプログラムは C#で実装した。利用者のスマートフォンは複数接続できるが、サイネージの操作機能は 1 台だけに割り当てられる。

## 5. 実験結果

### 5.1 検証 1

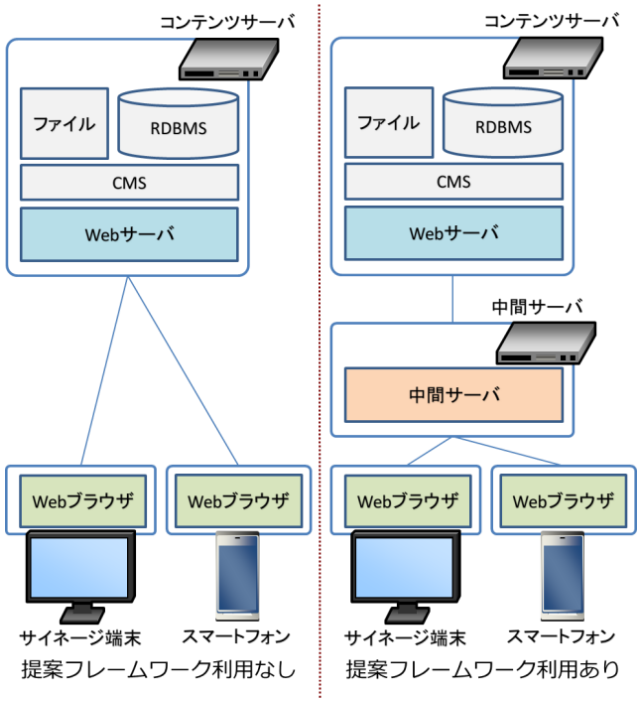


図 3　検証用システムのシステム構成図

表 1 に検証 1 での提案フレームワーク利用有り無しそれぞれにおける工数やソースコードの情報を示す。工数は要した時間、ライン数はプログラムの量、関数およびグローバル変数はプログラムのわかりやすさの指標の一つとして計測した。開発工数は提案フレームワークを利用したことで、86%に抑えられた。同様に JavaScript のライン数は 63%、HTML のライン数は 60%に抑えられた。サーバサイドの C#のプログラムは提案フレームワークを利用した場合、不要であった。

### 5.2 検証 2

表 2 に検証 2 での提案フレームワーク利用有り無しそれぞれにおける工数やソースコードの情報を示す。各数値は機能追加に伴って、新たに追加した部分の値である。

開発工数は検証 1 と同様に、86%に抑えられた。その中でも設計工数が 50%となっている。しかし、ソースコードについての差分は僅かであった。

## 6. 考察

### 6.1 検証 1 について

提案フレームワークでは通信制御プログラムを書かなくても、マルチデバイス連携システムを開発できるようにすることを要件とした。検証結果から、従来はサーバサイドプログラムを必要としているが、提案フレームワークを利用するとサーバ側のプログラムなしでマルチデバイス連携が実現できることがわかる。また、クライアント側の JavaScript のコード量も削減できている。

表 1　リモコン操作機能開発結果比較

| | 開発工数 | | | JavaScript | | | HTML | C# |
|---|---|---|---|---|---|---|---|---|
| | | 設計工数 | 実装工数 | ライン数 | 関数 | グローバル変数 | ライン数 | ライン数 |
| 従来方法 | 7人日 | 2人日 | 5人日 | 481 | 17 | 24 | 100 | 8 |
| 提案フレームワーク | 6人日 | 2.5人日 | 3.5人日 | 303 | 12 | 20 | 60 | 0 |
| 従来比 | 86% | 125% | 70% | 63% | 71% | 83% | 60% | 0% |

表 2　「おすすめ」「行きたい」機能追加開発結果比較

| | 開発工数 | | | JavaScript | | | HTML | C# |
|---|---|---|---|---|---|---|---|---|
| | | 設計工数 | 実装工数 | ライン数 | 関数 | グローバル変数 | ライン数 | ライン数 |
| 従来方法 | 3.5人日 | 1人日 | 2.5人日 | 202 | 8 | 6 | 8 | 296 |
| 提案フレームワーク | 3人日 | 0.5人日 | 2.5人日 | 198 | 7 | 4 | 8 | 274 |
| 従来比 | 86% | 50% | 100% | 98% | 88% | 67% | 100% | 93% |

被験者からは、次のようなコメントがあった。「デバイス間でのメソッド・イベントの対応付け等を考えなくてよいのが直観的でわかりやすかった。」「従来法では、値をやりとりするのにも、メソッド・イベントで渡す必要があるが、提案法では直接変数に代入するだけでよいのがわかりやすかった。」従来、複数ブラウザ間の通信が必要な部分が、変数に値を代入するだけで実現可能なことが評価された。これは、Web アプリケーション開発者にとって大幅に敷居を下げることにつながると考える。

また、通信制御機能を提案フレームワークに任せることで、エラー処理の記述が省略できることも次のように評価された。「従来法では、WebSocket を利用している部分で通信エラーの際の挙動も記述していたが、提案法では記述を省略できたのが簡便でよかった。」

一方で、設計工数だけは従来方法よりも増加している。この点について次のようなコメントがあった。「設計時には、従来方法で利用した機能やライブラリの対応状況の調査に時間を要した。」また、次のようなコメントもあった。「従来法で利用していたライブラリで提案フレームワークで利用できないものがあったため、その部分の変更に工数が多くかかった。」提案フレームワークがあらゆる JavaScript のコードに対応できる状態ではないため調査が必要となり、従来法の実装で利用している JavaScript ライブラリで未対応のものが発見されたため、同様の機能を持つライブラリを探すなど、異なる作業をする必要があったことが設計工数増加の要因と考えられる。

### 6.2 検証 2 について

ソースコードの差分が僅かであったことについて、次のようなコメントがあった。「一度加算した値をサーバ側で保持する必要があり API を追加したため、両方法ともで時間を要した。サーバ側で保持しない状態ならば提案法は簡単に実装ができた。」この機能追加では、通信制御以外の処理が多く必要となり、追加部分のほとんどが通信制御以外の処理を記述したコードであった。そのため提案フレームワーク利用の効果があまり出なかったと考えられる。

しかし、設計工程では違いがあったというコメントがあった。「従来法の場合、どんな API を追加する必要があるのか考えるのに時間を要したが、提案法ではその点をあまり考えなくてよいため、設計が容易だった。」ここでの API はサーバサイドの Web API である。提案フレームワークを利用しない場合、通信でやりとりする内容をあらかじめすべて洗い出して API 設計をする必要があるが、提案フレームワークを利用する場合はその設計が不要となるため、時間を短縮することができたと考えられる。

## 7. まとめ

本稿では、著者らが提案している通信制御プログラムなしに Web アプリケーションによるマルチデバイス連携システムを実現するフレームワークをデジタルサイネージとスマートフォンの連携システムの開発に利用した場合と利用しない場合の、開発に要した工数とコードの比較を行った。開発工数の短期化とコード量の減少ならびにコードの単純化が見られた。

今後は、提案フレームワークが対応できる JavaScript ライブラリを増加させ、Web アプリケーション開発者が従来通りの開発でマルチデバイス連携システムを構築できるようにフレームワークの充実化を図る。

### 参考文献

[1] W3C, HTML5, http://www.w3.org/TR/html5/, Dec-2012
[2] 中茂 睦裕, 宮崎 泰彦, 渡邊 大喜, 槙 優一, 徳永 徹郎, “中間ブラウザによるマルチデバイス連携技術の提案”, 研究報告グループウェアとネットワークサービス（GN）, 2013-GN-87(6), 1-6 (2013).
[3] UPnP Forum, UPnP Specifications, http://upnp.org/sdcps-and-certification/standards/, 参照 Jun-2013
[4] Digital Living Network Alliance, Consumer Home , Digital Living Network Alliance, http://jp.dlna.org/, 参照 Jun-2013
[5] Apple, AirPlay, http://www.apple.com/jp/airplay/, 参照 June-2013
[6] Wi-Fi Alliance, Wi-Fi CERTIFIED Miracast™, http://www.wi-fi.org/wi-fi-certified-miracast%E2%84%A2, 参照 Jun-2013
[7] NTT ぷらら, ひかりＴＶ りもこんプラス, http://www.hikaritv.net/point/remoconplus/, 参照 Jun-2013
[8] （株）東芝, レグザ Apps コネクト | REGZA : 東芝, http://www.toshiba.co.jp/regza/apps/index_j.htm, 参照 Jun-2013