Vol.2013-GN-87 No.6
2013/3/18

中間ブラウザによるマルチデバイス連携技術の提案

中茂睦裕[†1]　宮崎泰彦[†1]　渡邊大喜[†1]　槙優一[†1]　徳永徹郎[†1]

Web 技術を活用したマルチデバイス連携サービスが広まりつつある．しかし，このようなサービスを構築するには，サーバとクライアント両サイドのプログラム開発とそれらの通信を制御するプログラム開発が必要である．一連のサービスを構築するには多様なスキルセットが要求され，開発者への負担が大きい．著者らは，通信制御プログラム無しでマルチデバイス連携サービスを実現するための中間ブラウザを提案する．本稿では，そのアイデアをプロトタイプシステムに実装して動作検証し，サービス開発コストの低減効果を評価したので報告する．

A study of the intermediate browser for cross screen services

NAKASHIGE MUTSUHIRO[†1] MIYAZAKI YASUHIKO[†1]
WATANABE HIROKI[†1] MAKI YUICHI[†1] TOKUNAGA TETSURO[†1]

Cross screen service is spreading with Web-based technology. However, to build such services, it is necessary to develop both server-side and client-side program. In addition, there is a need to develop program for controlling the communication between their devices. To build a set of services, developer required a set of various skills. We propose a middle browser for implementing the cross screen services without the communication control program. In this paper, we tested prototype system implemented with this idea. We also have evaluated the effectiveness of reducing the cost of service development by proposed system.

1. はじめに

ドラフティング中の周辺仕様は数多く残っているものの，W3C による HTML5 仕様の策定作業が完了して勧告候補となった．仕様策定の動きと並行して，Chrome, FireFox, Opera，Safari など主要な Web ブラウザへは HTML5 対応の実装が進んでいる[1]．ブラウザでの HTML5 対応実装が進むと，これまでネイティブアプリでしか実現できなかったようなサービスもブラウザさえあれば利用できるようになると期待されている．例えば，WebRTC[2] による遠隔コラボレーションが挙げられる．特別なアプリケーションのインストール作業をしなくても，PC に接続されたマイクやカメラを使ったビデオチャットが実現できる．

一方で，Web ブラウザを搭載したデバイスとして多種多様なタイプが登場している．旧来からの PC，スマートフォン，タブレットに加えて，TV，音楽プレーヤ，カーナビ，家電などが新たに加わってきた．これらはデバイスメーカや OS が異なる上，搭載する CPU，メモリ，保有機能の差異も大きい．これらのデバイス環境に適応してコンテンツを出し分けるマルチデバイス対応のサービスが普及している．家庭内に限定しても様々なスクリーンデバイスが存在しており，これらのマルチデバイスを連携したサービスも展開が始まっている．例えば，複数のスクリーンを連動させてゲームを楽しむ任天堂の Wii U[3]，手元端末のメディアを大画面で共有する Apple の AirPlay[4], 映像配信サービスの操作を手元デバイスでおこなう NTT ぷららのりもこんプラス[5] などが挙げられる．マルチデバイス対応サービスとマルチデバイス連携サービスの動作イメージを図 1 に示す．

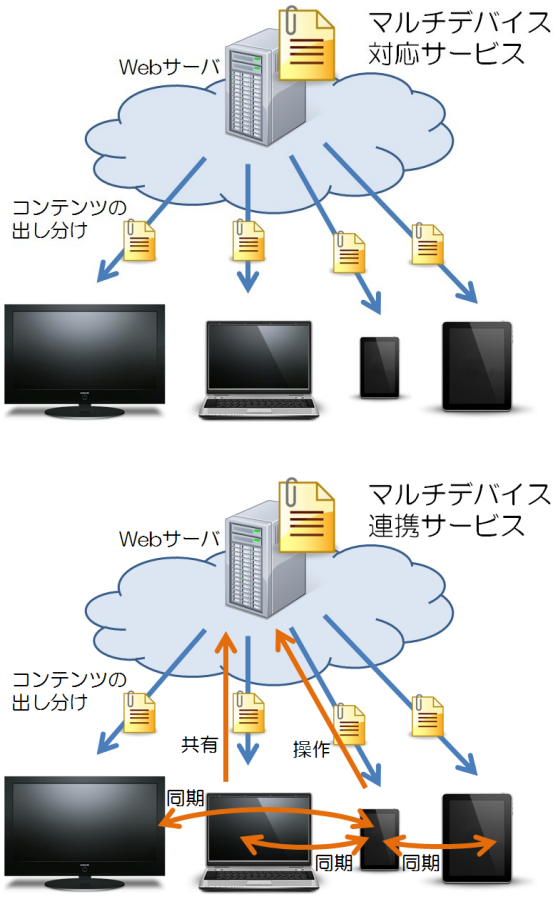

図 1　マルチデバイス連携サービスの動作イメージ
Figure 1　Operation image of cross screen services.

†1 日本電信電話株式会社 NTT サービスエボリューション研究所
NIPPON TELEGRAPH AND TELEPHONE CORPORATION

現状では，これらのサービスはまだ一部のサービス領域のみが対象となっている上，サービスを享受するためには特定のデバイス環境が要求される．しかし，広く Web の世界では様々なコンテンツ形態が存在している．これらを活用して先に述べたような多種多様なデバイスに対してマルチデバイス対応サービス提供しようとすると，アニメーションを表現するための記述言語の違いや，映像コーデックの違いは，しばしばサービス開発者を悩ませる．また，マルチデバイスを連携するサービスを構築するためにはデバイス間あるいはサーバ・クライアント間の通信制御プログラムを実装する必要があるが,その実装方法も多様である．

先述のとおり，現状のマルチデバイスを連携したサービスは特定のサービス領域に限定されている．その原因として，開発者への負担がある．具体的には，デバイス間の通信の考慮と，多種多様なデバイス環境への対応が挙げられる．本稿では，これら開発者の負担を軽減し，マルチスクリーン連携サービスを創出しやすくする基盤を提案する．具体的には HTML5 仕様の上で動作するシステムを構築し，マルチスクリーン間で自由自在にコンテンツを移動したり，同期制御したりすることができるようにした．同システム上でサービスプロトタイプを開発し，動作検証をおこなったので報告する．

2. 課題と先行事例

先述したように，世の中には，多様なデバイス環境，多様なコンテンツ形態，多様なユースケースが溢れている．このような状況下でどのような課題が顕在化しているか，エンドユーザの視点と開発者の視点でそれぞれ分析する．

(1) エンドユーザ側の課題

まず，エンドユーザが抱える課題としては，インターオペラビリティが挙げられる．具体的には，あるサービスを享受したい場合は，それに合わせてデバイス環境を選択することを強いられている．特定メーカの対応機種が必要な場合も多い．このようなサービスでは，デバイス間での連携方法が独自仕様であるため異なるメーカの機器あるいはソフトウェア間で同期した映像視聴やゲームなどのアプリケーションの利用が困難である．

例えば，異メーカ間の機器の相互接続を容易にするための UPnP 規格[6]や DLNA[7]規格があり，手元のスマートフォンに記録した動画を TV で視聴するなどの連携サービスを受けられる．ただし，これらのガイドラインに沿って設計されたデバイスが必要である．また，それらはマルチキャストアドレスで通信する都合上，同一 LAN 上でしか連携できないなどの課題が残る．

(2) 開発者側の課題

一方，開発者が抱える課題としては，エンドユーザが有する多種多様な環境への対応するための開発コストが挙げられる．例えば，デバイスの OS バージョンなどのプロファイル差異に対応したプログラミングが要求される．開発コストを低減するため 1 ソースでマルチデバイス対応サービスを提供するための様々なアプローチが取られてきた．例えば，デバイスごとのスクリーンサイズの差異を吸収するためのレスポンシブ Web デザイン（RWD）やスケーリング，デバイスごとの保有機能の差異に対応するためのプログレッシブエンハンスメントやモデル・ビュー・コントローラ（MVC）が挙げられる．

(3) マルチデバイス連携サービスの例

本稿では，さらに 1 歩進めたマルチデバイス連携サービスを扱う．先述の UPnP・DLNA では各デバイスがミニマムな Web サーバを有しておりデバイス間の同期・通信が実現されている．他にもいくつかの方法が提案されている．例えば，放送局が検討を進めるハイブリッドキャスト[8]や IPDC（IP データキャスト）[9]は放送受信機（TV・STB）がサーバとなり，放送から受信した制御情報を元に宅内のスクリーンデバイスへ放送波の付属コンテンツを配信する．また，移動体の通信キャリアではスマートフォンをメディアハブとして位置付けて Web サーバを立て，周囲の TV やタブレットと連携する提案をしている[10].

これらは，いずれも特定メーカの開発プラットフォームが必要で，サービス実装のために特化されたサーバ環境とデバイス間の通信制御プログラムの個別開発が必要である．つまり，HTML5（HTML5・CSS3・JavaScript）以外の言語を要求され，例えばサーバサイドの言語（Perl・PHP など）による開発が発生する．また，デバイス間の P2P 通信アプリケーションの開発，それらに付随するセッション管理や同期処理の煩雑さも課題となる．汎用のサーバをネットワーク上に設置することも考えられるが，インタラクションが多いアプリケーションを利用する際にクラウド上の WebSocket サーバを利用してデバイス連携することはレスポンス性能の面では不利である．また，個人情報が詰まったスマートフォンで Web サーバを稼動させることにはセキュリティの面でも不安が残る．

(4) 先行研究

ユーザが利用する複数のデバイス環境に応じて Web コンテンツを分割・結合してスムーズな Web 閲覧を実現するアイデアが提案されている[11][12][13]．しかし，これらは静的なページの操作に限定されている．近年では JavaScript を駆使して同一 Web ページ内に留まりながらコンテンツを動的に変化させるサービスが数多く出現しており，先行研究の事例ではこれらのサービスをカバーできない．本稿では，JavaScript による Web ページ内アプリケーションを含むコンテンツも対象にし，マルチデバイス連携サービスを実現できる基盤を扱う．

次章では，課題として挙げたインターオペラビリティと開発コスト低減の観点から，マルチデバイス連携サービスを創出しやすくする汎用の中間ブラウザを提案する．

3. アプローチ

前章で述べたように，マルチデバイス連携サービスを展開する際には，インターオペラビリティと開発コスト低減が課題である．そこで，本稿で提案する中間ブラウザは，HTML5 ベースで動作するプロダクトとする．こうすることで，クライアントデバイスは HTML5 対応ブラウザさえ搭載していればよく，デバイス環境に依存しないサービス基盤を実現できる．中間ブラウザが接続されたデバイスを一元管理するため，各デバイスで Web サーバを稼動させる必要は無い．

具体的には，サービスの記述は HTML5 でおこない，コンテンツ制作は Web の 1 ページだけで済むようにする．各デバイスに提供するコンテンツを別ページに記述して準備するのではなく，1 ページ内にコンテンツを集約する．中間ブラウザは，1 ページに集約されたコンテンツから，各デバイスに適した部分コンテンツを分割して配置する．このように，中間ブラウザはいわばコンテンツ配信サーバとクライアントデバイスとの間に割って入りプロキシサーバのように動作する．プロキシサーバと大きく異なる点は，中間ブラウザがデバイス間の通信を担ってデバイスごとのコンテンツを同期し，サービス連携を実現する点である．そのために，コンテンツに含まれる JavaScript を実行しレンダリングする機能を有する．

中間ブラウザの基本的な動作アイデアを図 2 に示す．中間ブラウザは，Web サーバから配信された HTML5 コンテンツ（HTML5・CSS3・JavaScript）と，配下のクライアントデバイスを一元管理する．つまり，JavaScript は一旦中間ブラウザ上で実行され，その結果が反映されたコンテンツが各デバイスへ配置される．中間ブラウザは，Web の 1 ページに集約された全体コンテンツから，各デバイスへ配置すべき部分コンテンツに分割し，それらを各デバイスへ配置する（①）．マルチデバイス間でサービス連携するためには，分割して配置された部分コンテンツ間の同期を取る必要があるが，各デバイスへ配置された部分コンテンツの状態は中間ブラウザが一元管理して把握しているため，各デバイスの同期を維持したまま，必要に応じてコンテンツを追加配置したり削除したりできる（②）．デバイス連携するための JavaScript のコードは中間ブラウザで自動挿入され，やはり中間ブラウザ上で一元的に実行される．また，部分コンテンツ単位でデバイスを横断して移動したり，一方のデバイスから他方のデバイスへ部分コンテンツを複製したりすることができる（③）．

このように中間ブラウザを実装すれば，コンテンツ制作者にとっては通信制御プログラムの開発無しにマルチデバイス連携サービスを構築できる．しかも，サービスを構築するためには Web 技術による 1 ソースで記述可能であり，開発コストの低減効果が見込まれる．

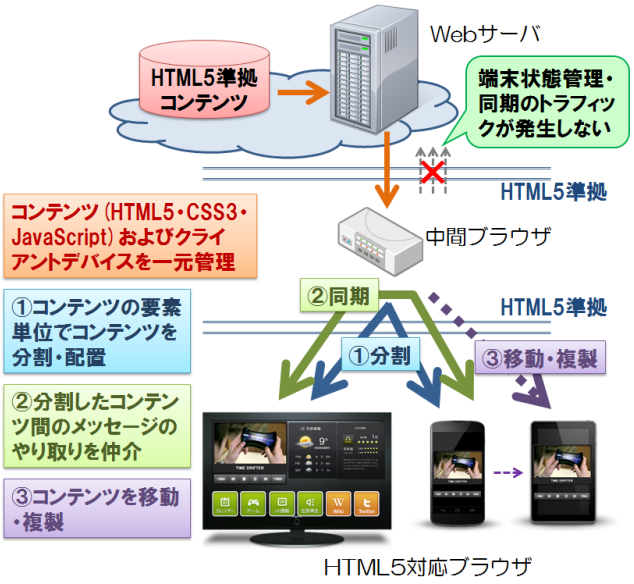

図 2　中間ブラウザの基本的な動作アイデア
Figure 2　The basic operation idea of the intermediate browser.

4. システム実装

前章で提案した実装アイデアを元に中間ブラウザを構築する．今回，中間ブラウザは Node.js 上に構築した．そのシステム構成を図 3 に示す．Node.js とは，JavaScript エンジン V8 がサーバ上でプログラムを実行できるように，ファイルやネットワーク I/O など多くの機能を有するプラットフォームである．軽量で効率よく多くのリクエストを処理するネットワークアプリケーションの構築ができる．

今回，中間ブラウザに用いた PC の OS は CentOS 6.3 であり，Node.js は Ver.0.8.19 を用いた．前章で述べたように，今回構築する中間ブラウザはコンテンツ配信する Web サーバとクライアントデバイスとの間に割って入る．つまり，クライアントデバイスから見るとサーバであるが，Web サーバから見ると唯一のクライアントである．

実装したシステムは大きく分けて下記の 3 つの機能で構成される．

- コンテンツ管理機能

ここでは，Web サーバから受け取った HTML5 コンテンツ（HTML5・CSS3・JavaScript）を理解して一元管理する．JavaScript の実行，DOM ツリーへの変換，コンテンツのレンダリングもここでおこなう．

- 通信機能

WebSocket サーバによって中間ブラウザとクライアントデバイスとの間で通信を担う．部分コンテンツの配置・変更・削除およびユーザ操作などによって変化する各デバイスの状態取得をおこなう．また，各デバイスでのコンテンツ表示に適した CSS を伝達する．

● 同期制御機能

ここでは，複製した部分コンテンツも含めて全ての部分コンテンツの同期を取り，デバイス連携を実現する．

各デバイスへ分割して配置する部分コンテンツの最小単位は，全体コンテンツに記述された DOM の各要素としている．こうすることで，既存の HTML5 コンテンツであっても新たに作り直さずにマルチデバイス連携サービスを提供できる．例えば DIV 要素でグルーピングされている部分コンテンツは，中間ブラウザ配下のデバイス間で移動・複製を自由におこなえる．また，別のデバイスへ分割して配置されていたとしても，部分コンテンツ間の同期も保証される．

映像配信サービスを実装する際を例に図 4 を使って説明する．全ての HTML5 コンテンツは DOM ツリーで表現できるが，デバイスに搭載されたブラウザはこの DOM ツリーに従ってコンテンツをレンダリングしている．このコンテンツの場合は，大きく分けて映像表示部分と再生制御部分の 2 つの DOM ツリーで構成される．

さて，中間ブラウザの配下には TV とスマートフォンが接続されている．各クライアントデバイスには HTML5 ブラウザが起動しており，中間ブラウザと各デバイス間は WebSocket によって通信する．TV には映像表示部分の DOM ツリーが，スマートフォンには再生制御部分の DOM ツリーが，それぞれ分割して配置される．

スマートフォンで再生・停止などのボタンを押して操作があると，その操作の情報は中間ブラウザへ通知される．中間ブラウザは TV の DOM ツリーの状態を変化させ，ボタン操作を反映する．

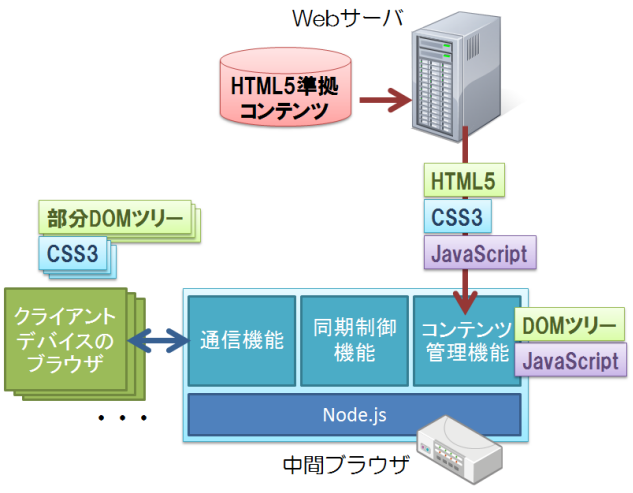

図 3　中間ブラウザのシステム構成

Figure 3　System configuration of the intermediate browser.

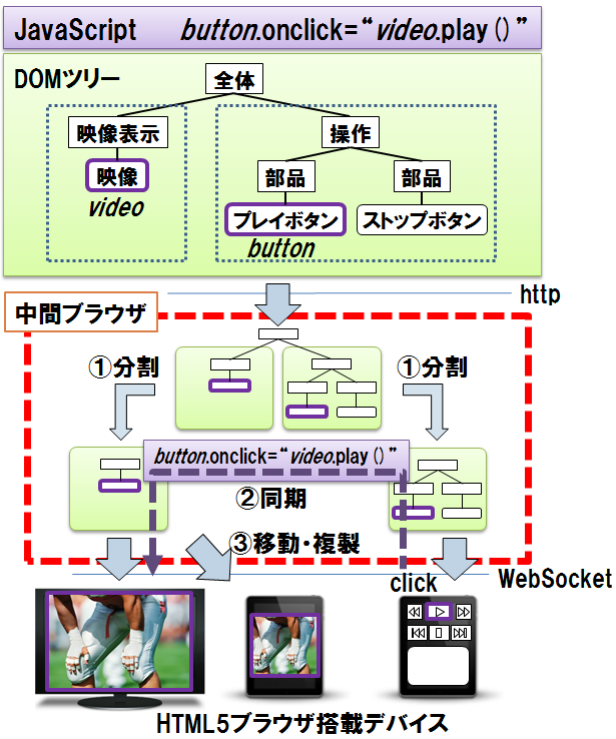

図 4　映像配信サービスを例にした挙動説明

Figure 4　Operation example with a video delivery service.

本技術で特徴的なのは，このようなデバイス間の連携に際してコンテンツ制作者が特別なコードを用意しなくて良い点である．今回の例では，スマートフォンに表示された再生・停止ボタンの操作に合わせて，TV の映像コンテンツが連動して再生・停止するが，用意するコンテンツにはデバイスを跨がった通信を意識することは無い．つまり，＜button.onclick="video.play()"＞のように単純にボタン操作（再生・停止）をするとビデオが変化（再生・停止）するような JavaScript を記述しておきさえすれば良い．実際にデバイスが同期するためのコードは中間ブラウザ内で自動的に生成され，実行される．

スマートフォンのボタン操作（再生・停止）による状態変化は中間ブラウザの通信機能が WebSocket により取得し，同じく WebSocket を利用して TV に表示された映像コンテンツの DOM 要素の状態を変更（再生・停止）する．このように，中間ブラウザを介することで，1 ページの Web コンテンツを準備するだけでマルチデバイス連携サービスを実現できる

5. 評価と考察

前章で構築した中間ブラウザを用いて，実際にマルチデバイス連携サービスのプロトタイプを実装してその動作検証をおこなった．1 つは文字チャットアプリケーション，もう 1 つは写真共有アプリケーションである．

表 1　中間ブラウザ有無による開発コード量比較
Table 1　Comparison of development code with and without the intermediate browser.

	サーバサイド	クライアントサイド		合計
アプリケーション	JS	HTML	JS	
文字チャット(従来サーバ)*1	36	25	27	88
文字チャット(中間ブラウザ)	0	18	28	46
写真撮影・共有(従来サーバ)*2	35	23	59	117
写真撮影・共有(中間ブラウザ)	0	24	14	38

*1: Socket.IOを利用　　単位:ライン
*2: WebRTCを利用

いずれのアプリケーションも設計どおりに動作し，中間ブラウザの配下でマルチデバイス間の同期も正常であった．開発したアプリケーション 2 つは，いずれも動作検証のための単純なものであるため非常に短いコードではあるが，本システムを利用した場合と利用しない場合での開発コストの低減効果を評価した．中間ブラウザの有無による開発カード両を比較した結果を表 1 に示す．

文字チャットアプリケーションでは開発コードを約 52%に圧縮できた．このアプリケーションは，入力した文字が各クライアントデバイス間で共有できるものである．本システムを利用しない場合は，サーバ・クライアント間の通信には Socket.IO を用いている．同じく，写真共有アプリケーションでは開発コードを約 32%に圧縮できた．こちらは，スマートフォンのカメラで撮影した写真を別のデバイスで共有して表示できるものである．本システムを利用しない場合は，サーバ・クライアント間の通信には WebRTC を用いている．

今回の評価では，サンプルが 2 種類しかないため，他のアプリケーション例でも追試が必要ではあるが，いずれのアプリケーションも通信制御プログラムの開発が不要となり，サーバサイドの開発コードがゼロである．提案する中間ブラウザを利用することで開発コードの削減に寄与しており，開発コストの低減効果があると言える．

もっとも，コードの削減量だけで効果を評価することは不十分である．これまで，マルチデバイス連携サービスを実装するには，図 5 の上図に示すように様々なスキルセットを保有する人材が必要であった．しかし，本技術を利用すれば，図 5 の下図のように HTML5 に関する知識セットさえあればマルチデバイス連携サービスを実装できる．クリエータのみによるサービス開発も可能であり，人員リソースの低減効果も期待できる．

同時に，マルチデバイス連携サービスを開発する際の工程も変えてしまう可能性がある．これまでは，サーバサイドとクライアントサイド双方の開発が必要で，これらを連結する設計およびテストが必要であった．サーバサイドの開発が無くなることで，これらの工程も不要となり，コンテンツ制作に専念できるようになる．

図 5　マルチデバイス連携サービス構築に要求されるスキルセット
Figure 5　Skill set required to build cross screen service.

6. まとめ

マルチデバイス連携サービスを創出しやすくすることを目的に，汎用の中間ブラウザを提案した．通信制御プログラムなどサーバサイドの開発無しでマルチデバイス連携サービスを実現できる，HTML5 ベースで稼動する新規技術を開発した．デバイス環境に依存しない基盤であるため，HTML5 対応のブラウザさえあればマルチデバイス連携サービスを享受できる．

エンドユーザにとってはインターオペラビリティの面でメリットがある．また，Web の 1 ページのコンテンツだけを準備すればサービス提供できるため，開発するコード量の低減効果がある．サービス開発者にとっては開発コストを低減できるメリットがある．

今後，より規模の大きな開発での評価を実施する予定である．また，具体的なユースケースを定めた上で中間ブラウザの機能拡充と簡便な操作 UI の検討を進める。併せて，

新たなマルチデバイス連携サービスの創出を目指して，コンテンツクリエータと積極的にコラボレーションしていきたいと考えている．

参考文献

1) 石井晋司, 渡部智樹, 井原雅行, 小林透: 次世代のコンテンツ流通にかかわる W3C における標準化動向, NTT 技術ジャーナル 2013.1, pp.56-59 (2013).
2) WebRTC: Web Real-Time Communication
http://www.webrtc.org/
3) 任天堂, Wii U:
http://www.nintendo.co.jp/wiiu/
4) Apple, AirPlay:
https://www.apple.com/jp/airplay/
5) NTT ぷらら, ひかり TV りもこんプラス:
http://www.hikaritv.net/point/remoconplus/
6) UPnP, Universal Plug & Play:
http://www.upnp.org/
7) DLNA, Digital Living Network Alliance:
http://www.dlna.org/
8) NHK, Hybridcast:
http://www.nhk.or.jp/strl/hybridcast/HCsummary.pdf
9) IPDC, Internet Protocol Data Cast:
http://www.ipdcforum.org/
10) NTT ドコモ, Smart TV dstick:
http://www.nttdocomo.co.jp/product/2013_spring_feature/dsh/dstick/
11) 前川卓也, 原隆浩, 西尾章治郎: 複数のモバイルユーザのための Web ページ分割を用いた協調 Web ブラウジングシステム, 情報処理学会研究報告. ITS, [高度交通システム] 2004(114), pp.1-7 (2004)
12) Jacek Chmielewski, Krzysztof Walczak: Application architectures for smart multi-device applications, Multi-Device '12 Proceedings of the Workshop on Multi-device App Middleware Article No. 5 (2012).
13) Bin Cheng: Virtual browser for enabling multi-device web applications, Multi-Device '12 Proceedings of the Workshop on Multi-device App Middleware Article No. 3 (2012).