Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

Webと機器を透過的につなぐ Multimodal Interactionフレームワークの実装

芦村和幸[†1,†3] 小松健作[†2,†3] 一色正男[†1]

World Wide Web Concorcium (W3C) の Multimodal Interaction Working Group (MMI-WG) では, 様々な入出力機器を Web と透過的に連携させるためのフレームワークである MMI 技術の国際標準化に取り組んでいる. 著者らは, W3C MMI-WG での MMI 技術国際標準化活動に参加するとともに, MMI 技術をより簡単に Web アプリケーションから利用することを可能とするために必要な, WebSocket を用いたオープンソフトの開発に取り組んでいる. 本稿では, 上記オープンソフトの実装状況について報告すると共に, 今後の課題と対策案について議論する.

Implementing Multimodal Interaction Framework for Transparent and Smarter Integration of the Web and CE Devices

Kazuyuki Ashimura,[†1,†3] Kensaku Komatsu[†2,†3]
and Masao Isshiki [†1]

The Multimodal Interaction Working Group (MMI-WG) of the World Wide Web Consortium works for global standardization of multimodal interaction technology as a framework for transparent and smarter integration of the Web and various Consumer Electronics devices. We participate in the activity of the WG, and also tackle to implement an open software using WebSocket which make people to author multimodal Web applications much easier. This paper reports the current status of the open software development, and then discusses the issues and possible solutions.

†1 W3C/慶應 (慶應義塾大学 SFC 研究所) - W3C/Keio (Keio Research Institute at SFC)
†2 NTT コミュニケーションズ株式会社 - NTT Communications Coporation
†3 W3C マルチモーダル対話ワーキンググループ - W3C Multimodal Interaction Working Group

1. はじめに

World Wide Web Consortium (W3C)[1] は, Web の可能性を最大限に導き出すことを目的として, Web 技術発明者である Tim Berners-Lee により創設された産業コンソーシアムであり, Web の発展と相互運用性確保に必要な各種プロトコルの開発に取り組んでいる. また, 著者らの先の研究報告[2] で紹介した通り, W3C おける Web 技術仕様化活動の一つである Multimodal Interaction Working Group (MMI-WG) では, 様々な入出力機器を Web と透過的に連携させるためのフレームワークである, W3C Multimodal Architecture (MMI アーキテクチャ)[3)–5)] の標準化が進められている.

本稿では, まず Web 技術の特徴について簡単に触れた上で, 近年ますます高度化する Web アプリケーション開発にあたっての課題について概観する. その上で, 様々な入出力機器を Web と透過的に統合するための枠組である MMI アーキテクチャについて概説するとともに, MMI-WG で取り組む MMI 実験システムの開発について触れる. さらに, 実験システム開発を通して明らかになった課題への対策として HTML5 の WebSocket 機能を MMI アーキテクチャへ組み込むために著者らが取り組んでいる「MMI on WebSocket ライブラリ」について報告し, 今後の課題と対策案について議論する.

2. Web 技術とその特徴

World Wide Web (以下, 単に「Web」)[6] は, インターネット上で提供される, 相互に接続されたハイパーテキスト文書により構成される情報空間, およびそれにもとづくサービスである. Web へのアクセス手法として, 通常「Web ブラウザ」というソフトウェアが用いられ, Hypertext Markup Language (HTML) 等の文書に記述されたテキスト情報, 画像情報, 動画情報等のマルチメディア情報を取り扱うことができる. 「Web ページにアクセスする (もしくは, Web ページを見る)」とは, 何らかの URI (Universal Resource Identifier) を指定することにより, Web 上に存在する Web サーバに蓄積された文書の位置を特定することに始まり, HTTP (Hypertext Transfer Protocol) 等のプロトコルに則って, 当該文書を PC 等のローカルクライアント (Web クライアント) 上に取得した上で, 文書構造を解析し, ディスプレイ画面等に表示することを意味する.

Web は世界中で共通的に利用されるがゆえに, その相互運用性 (Interoperability), 国際化 (Internationalization), 入出力方法の多様性 (Multi-Modality), および利用者への親和性 (Accessibility) を保証する必要があり, そのために, World Wide Web Consortium (W3C)

Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

等による, 国際的な技術標準化が行われている. なお, Web 技術国際標準化の詳細については, 著者らの既報[2] 等を参照いただきたい.

3. HTML5 による Web のアプリケーション・プラットフォーム化

Web 上の文書記述のために主に利用される形式として, HyperText Markup Language (HTML)[7),8)] が挙げられる. HTML は, ⟨html⟩ というようにタグをブラケットで囲んだ「要素」によって記述され, テキスト, 表, 画像, 音声等の情報を含めることができる. Web ブラウザは, HTML で記述された文書を解析した上で, その内容をディスプレイ上に表示したり, 音声情報の再生を行なうが, 動画情報等, より高度なマルチメディア情報を取り扱うためには, プラグインと呼ばれる拡張を必要としており, 各種プラグイン・ソフトウェアの具体的実装内容が開発元ごとに異なることから, 標準的な取扱いに問題があった. そのため, 様々な機器やモダリティに対応し, より表現力豊かな Web アプリケーションを実現するための標準的な Web アプリケーションのプラットフォームとして HTML5[9),10)] が提案されている. HTML5 は, HTML が元来持つ基本的な GUI 制御機能に加え, よりきめ細かな描画形式の制御, および動的 (Dynamic) かつ Web と利用者との相互的なやりとり (Interactive) に対応したものとなっている. HTML5 で新たに追加された機能の代表例として以下が挙げられる.

- Video & Audio
- Canvas
- Drag & Drop
- WebSocket
- Web Storage
- Web Workers

なお, 上記の各新機能の詳細については, 著者らの既報[2] 等を参照いただきたい. また, 各機能の具体的なデモンストレーション[11),12)] が Web 上で公開されているのでそちらも参照いただきたい.

4. Web アプリケーション開発上の課題

3 章で触れた通り, 近年, HTML5 は単なるマークアップ言語を越え, より表現力豊かな Web アプリケーションを実現するためのプラットフォームとなるべく, 既存の HTML に対して多種多様な機能追加が行なわれており, 動的かつインタラクティブな Web アプリケーションを構築することが可能となりつつあるが, 以下のように, いくつかの技術的課題も指摘されている. なお, 各課題の詳細については, 著者らの既報[2] 等を参照いただきたい.

- アプリケーション記述に関する課題
- 分散アプリケーション統合に関する課題
- 入出力モダリティに関する課題

5. W3C MMI アーキテクチャ

4 章で触れた Web アプリケーション開発上の課題を解決するべく, W3C では, 様々な入出力モダリティを動的に選択できるとともに, 各モダリティがどの機器上に実装されているのかを意識することなく透過的に組み合わせて利用することを可能とするフレームワークである, Multimodal Architecture and Interfaces (MMI Architecture; MMI アーキテクチャ)[3] の標準化に取り組んでいる. 図 1 に MMI アーキテクチャの概念図を示す.

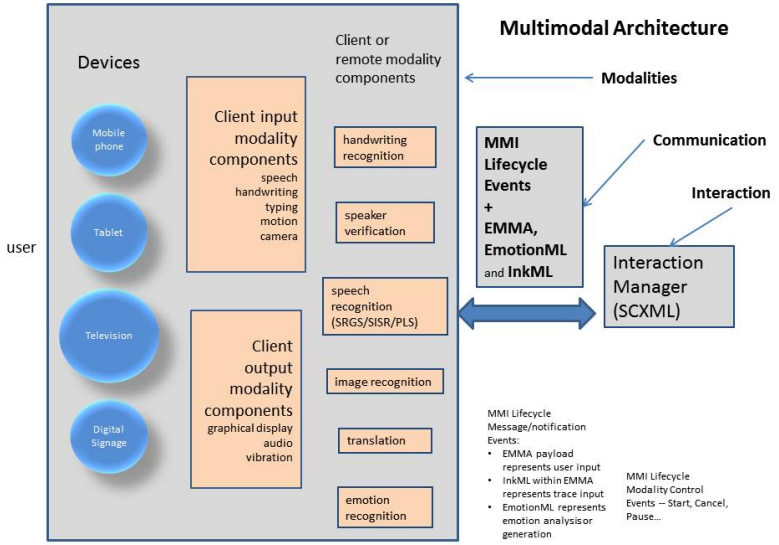

図 1 MMI アーキテクチャ
Fig. 1 MMI Architecture

Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

また図 2 に MMI アーキテクチャの構成要素を示す.

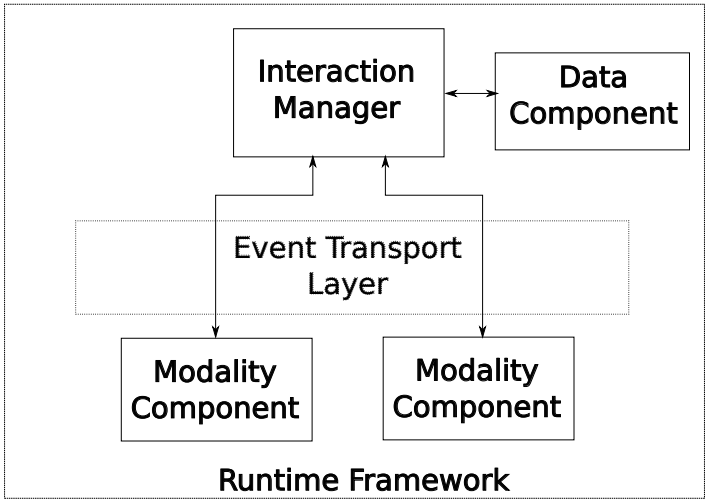

図 2 MMI アーキテクチャの構成要素
Fig. 2 MMI Architecture components

MMI アーキテクチャでは, GUI に加え音声や手書き入力等, 様々なモダリティを組み合わせて利用することが想定されており, モダリティ・コンポーネント (MC; クライアント側機器) とインタラクション・マネージャ (IM; サーバ側コントローラ) という構成要素間で, 相互に情報を交換するための通信プロトコル (イベントおよびデータ形式) が規定されている. ここで, イベント[13] とは, 一連のアプリケーション・ライフサイクル (アプリケーションの起動から終了まで) の中でアプリケーションの処理ロジック自体とは独立して発生し, MC と IM の間でやりとりされる以下のようなメッセージのことを指す.

(1) 処理を開始するための構成要素間のネゴシエーション,
(2) アプリケーション利用中に都度発生する任意のデータ交換 (キーボードが押された, 音声入力があった等の情報, および入力データ)
(3) アプリケーションを終了する際のネゴシエーション

なお, MMI アーキテクチャの詳細については, 関連する下記の情報も参照されたい.

- MMI アーキテクチャ全般: 著者らの既報[2]
- ライフサイクル・イベントの詳細: MMI アーキテクチャ仕様書[3]
- データ形式の詳細: EMMA 仕様書[14]
- マルチモーダルの例: Speak4It[15], Cue-Me[16] SpeechTEK Europe[17]

6. HTTP コネクションによる第 1 バージョン MMI 実験システム

MMI アーキテクチャによるマルチモーダル・アプリケーションを実現するためには, 多様な入力形態への対応, 入出力の同期, ある処理から派生する別処理との通信など, ネットワーク制御に関する高度な知識が必要である. そのため, システム実装のためのガイドラインやオーサリング・ツールの充実している通常の Web アプリケーション構築に比べると難易度が高い. そこで, MMI アーキテクチャの仕様策定に取り組む MMI-WG[18] では相互運用性の検証および実装ガイドラインの作成のためのタスクフォース (Interoperablity Testing Task Force; Interop TF) を設立し, 複数の参加企業で分担しながら, 図 3 に示すような三つの構成要素が協調して動作するアプリケーションを実装し, 問題点の洗い出しを行なった.

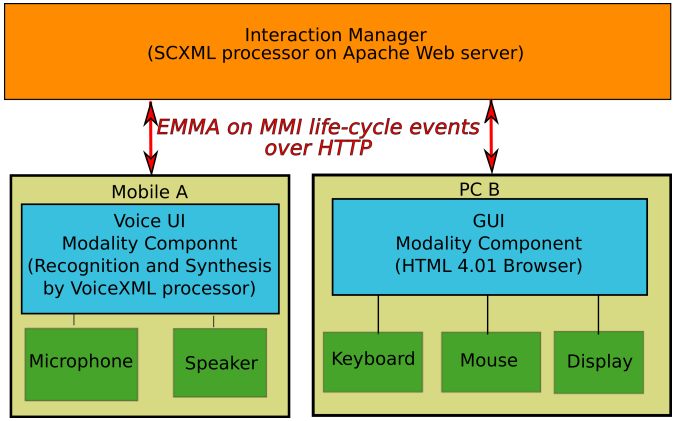

図 3 HTTP コネクションによる MMI 実験システム (第 1 バージョン)
Fig. 3 MMI Test System (Ver. 1) by HTTP Connection

上記システムを構成する三つの要素は以下の通り.

- ネットワーク上のインタラクション・マネージャ
- 携帯電話等 (Mobile A) に搭載された音声ユーザインタフェース・モダリティ
- PC 等 (PC B) に搭載された GUI モダリティ

Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

システム構築時に明らかになった問題点等については, MMI-WG の正式な文書として公開準備中である. なお, 上記第 1 バージョン実験システムの詳細については, 著者らの既報[2] を参照いただきたい.

7. ネットワーク接続における課題と対策: WebSocket の利用

現在, 主に Web 上で利用されているデータ通信プロトコルは HTTP であるが, HTTP は, 元来, HTML ファイルをサーバ側からクライアント側へダウンロードするために開発されたものであり, ファイルのダウンロード処理に特化した仕様となっている. 具体的には, データの送受信順番が厳密に規定されているプロトコルとなっており, クライアントからのデータ要求がサーバーに送信された後でないと, サーバ側からクライアントにデータを返信できない仕組みになっている. この HTTP の仕様制限は, ファイルの (サーバ側からクライアント側への) ダウンロードに関しては全く問題がない. しかしながら, Web が登場してから 20 年が経過する間に, Web の利用方法は多様なものとなっており, 近年ますます双方向的な利用が増加するにしたがって, この仕様制限が大きな問題となってきている.

マルチモーダル Web アプリケーションを実装するにあたって, Ajax 等, 従来の HTTP ベースの Web アプリケーションで用いられていたマッシュアップ手法[19] を利用する場合, サーバ (インタラクション・マネージャ) 側からクライアント (モダリティ・コンポーネント) 側へデータを任意のタイミングでプッシュ配信することが難しい. 例えば, 上記 6 章で触れた第 1 バージョン実験システムにおいても, 基本的にはいわゆる Comet 手法を利用しており, インタラクション・マネージャ側でイベントが発生したかどうかをモダリティ・コンポーネント側から周期的に確認する (ポーリング) 必要がある. そのため, モダリティ・コンポーネントからインタラクション・マネージャへ不必要なイベントを多数送信する必要が生じ, ネットワークおよび機器リソース上の無駄が多い.

一方, 最新の Web 標準仕様である HTML5[9),10)] では, 3 章で触れた通り, HTTP にもとづく通常の Web サーバとの通信とは異なる, 双方向かつ全二重のリアルタイム通信を可能とするメカニズムである WebSocket[20),21)] が規定されている. WebSocket が新たに仕様化された背景として, Web アプリケーションが近年ますますインタラクティブ (相互的; 双方向的) に利用されていることが挙げられる. WebSocket の機能を用いることにより, 一度インタラクション・マネージャとモダリティ・コンポーネント間でコネクションが張られた後は, アプリケーション運用に必要な通信を (毎回ポーリングや TCP ハンドシェイクをすることなく) 全てそのコネクション上で継続して行なうことが可能となる. そのため, アプリケーション記述が簡略化されるとともに, ネットワークおよび機器への負荷が減少し, 実行速度の向上が望まれる. 例えば, 小松の作成した WebSocket のデモサイト[22] によると日本語テキストの分かち書き処理を行なう Web アプリケーションにおいて, 通常の HTTP コネクションでは約 3 秒程度の処理時間がかかるのに対して WebSocket コネクションを利用した場合 100msec 以下の速度で処理が可能である (=約 30 倍の速度向上).

マルチモーダル Web アプリケーションにおけるネットワーク接続に WebSocket を利用することのメリットを以下にまとめる.

ネットワーク・トラフィックの低減:
マルチモーダル Web アプリケーションでは, 構成要素間で, ライフサイクル・イベントをリアルタイムにやりとりする必要があり, HTTP にもとづくデータ通信のオーバーヘッドは無視出来ない.

使用リソースの低減:
HTTP ベースでサーバ側からクライアント側へデータ送信する手法である Comet では, TCP ソケットのリソースを浪費することとなる. WebSocket 利用によるソケット数の削減は, CPU 使用率の低減の効果も期待できる.

処理速度の向上:
ライフサイクル・イベントの送受信にあたってはリアルタイム性が重要であり, ネットワーク・トラフィックおよび処理負荷の低減により処理速度の向上が期待できる. また, 不要な通信のためのネットワーク待機状態を減らす効果もあるため, 一層の速度向上が期待できる.

8. MMI on WebSocket ライブラリ

マルチモーダル Web アプリケーションにおけるネットワーク接続にあたって HTML5 の WebSocket 機能を利用することを可能とするために, 著者らは, WebSocket を MMI アーキテクチャに組み込むのに必要な JavaScript ライブラリである「MMI on WebSocket ライブラリ」(以下, 「本ライブラリ」) を実装した. 以下に, 本ライブラリ開発にあたっての要求仕様, モジュール構成, 動作確認および負荷試験の結果について報告するとともに, 今後の課題と対策案について議論する. 本ライブラリは, オープンソースのマルチモーダル・ワークベンチとして一般公開し, 世界中の Web アプリケーション開発者がマルチモーダル Web アプリケーションを構築する際の一助とする所存である.

Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

なお, 著者らは情報処理学会 試行標準 /WG4 小委員会[23] と連携した上で標準化の検討作業を進めてきており, 今後も引き続き学会側の活動と協調しつつ検討を進めていきたい.

8.1 ライブラリの基本要求仕様

本ライブラリ設計にあたっての基本要求仕様を以下に列挙する. 本ライブラリでは, モダリティ · コンポーネントおよびインタラクション · マネージャ間のデータ送受信に WebSocket を利用しており, 下記要求仕様項目のうち, 特に第 4 点目 (リアルタイムなイベント配信) において効果が期待される.

(1) ネットワーク上に分散する複数の機器上の様々なモダリティを, ライフサイクル · イベントという統一的なイベント操作の仕組み, および EMMA という標準的なデータ形式により連携させることを可能とする. これを通して, Web アプリケーションのライフサイクル (処理の開始と終了の規定) および処理中のデータ送受信をライフサイクル · イベントにより明確かつ簡潔に規定することを可能とする.

(2) デジタルテレビやビデオ等, 自宅内ネットワーク (基本的にプライベートなネットワーク) 上の機器を, Web アプリケーション (いわゆるクラウド · サービス) と連携させることを可能とする.

(3) 既存のハードウェアおよびソフトウェアに対して, ライフサイクル · イベントの操作機能を付加することにより, モダリティ · コンポーネントとして再利用することを可能とする.

(4) 通常の HTTP ベースのネットワーク · コネクションでは取扱いが困難な, インタラクション · マネージャ側からモダリティ · コンポーネント側へのイベント配信をリアルタイムに実現することを可能とする.

8.2 モジュール構成

本ライブラリは以下の 3 つのモジュールにより構成される. なお, 本ライブラリは MMI アーキテクチャのライフサイクル · イベントを HTML5 ブラウザ側から操作することを目的とするため, JavaScript を用いて実装した. また, ネットワーク機能については, 実装の簡便性および既存の HTTP ベースのネットワーク接続との比較のために, Node.js[24),25)] による socket.io を用いた.

フロントエンド · モジュール:
MMI アーキテクチャで規定された各ライフサイクル · イベントに対応する, JavaScript メソッドを API として提供する.

MMI モジュール:
MMI アーキテクチャの構成要素である, モダリティ · コンポーネントおよびインタラクション · マネージャ間でやりとりされる各種データ (EMMA を想定) の生成および妥当性検証を行なう. モダリティ · コンポーネント側モジュールとインタラクション · マネージャ側モジュールが必要.

通信モジュール:
WebSocket による具体的なライフサイクル · イベント送受信を行なう. モダリティ · コンポーネント側モジュールとインタラクション · マネージャ側モジュールが必要.

各モジュールは, インタラクション · マネージャ側およびモダリティ · コンポーネント側のそれぞれにおいて, 図 4 に示す形で実装される.

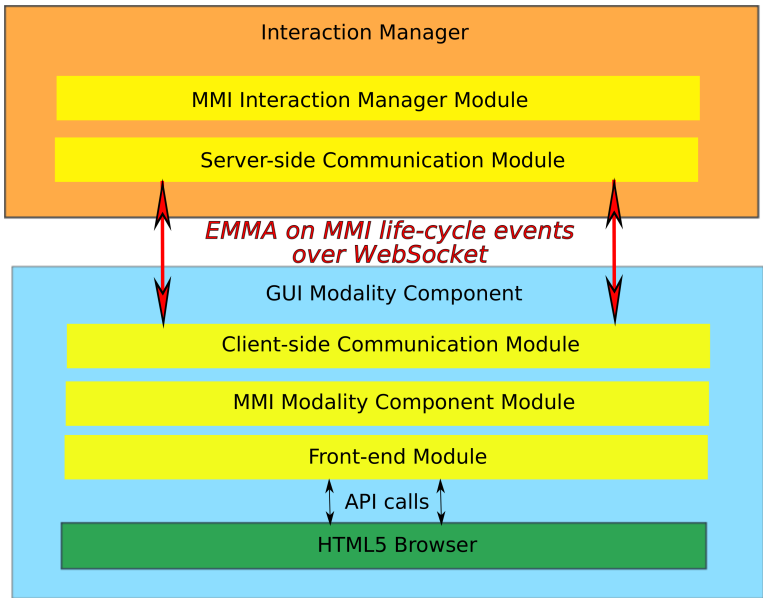

図 4 MMI on WebSocket ライブラリの構成
Fig. 4 MMI on WebSocket Library

Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

9. 動作確認および負荷試験

本ライブラリの動作を確認するべく, まず各種 Web ブラウザを用いて, MMI アーキテクチャ仕様書[3] に含まれる例に則って, アプリケーション開始から終了までの一連のライフサイクル・イベントが動作することを確認した.

さらに, Web ブラウザに相当するコマンドライン・インタフェース=ベースの HTML クライアントツールを作成した上でモダリティ・コンポーネントに組み込み, 複数 (最大 100 件) モダリティ・コンポーネントから同時に単一のインタラクション・マネージャへアクセスする負荷をかけ, インタラクション・マネージャ側の CPU 使用率およびメモリ消費量を測定した. 図 5 に結果を示す.

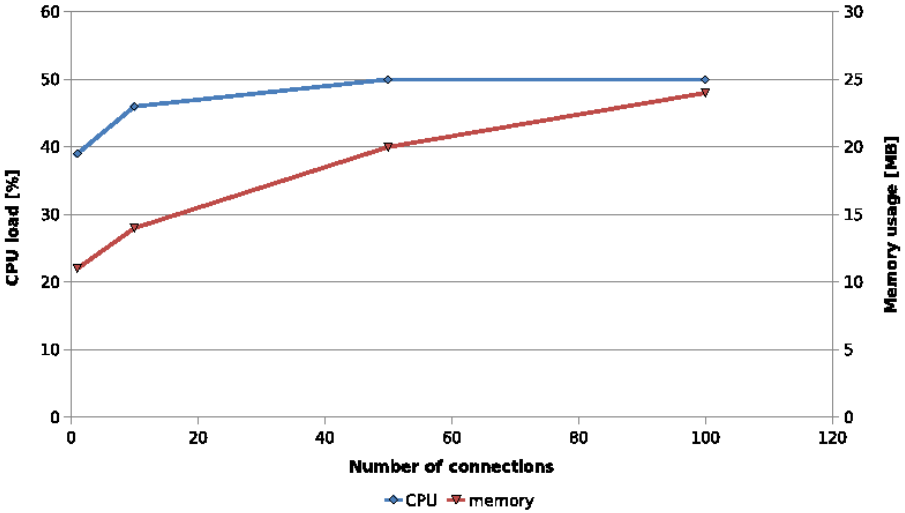

図 5 負荷試験結果
Fig. 5 Load test results

図 5 に示す通り, 100 件のモダリティ・コンポーネントから同時に単一のインタラクション・マネージャへアクセスした場合においても CPU 使用率は 50%程度にとどまっている. また, 50 件同時アクセスと 100 件同時アクセスにおいて差が見られない (飽和状態) ことから, より多くの同時アクセスにも耐えうるものと予想される.

一方, メモリ使用量については, 100 件同時アクセスにおいても 25MB 程度しか使用していないことから相当数の同時アクセスに耐えうると考えられる. なお, 単一アクセス (=1 件) と 10 件同時アクセスの差, 10 件同時アクセスと 50 件同時アクセスの差, 50 件同時アクセスと 100 件同時アクセスの差が徐々に小さくなっていっていることから, ある程度の同時アクセス数 (例えば 1000 件程度) でメモリ使用量が飽和状態になり, より多くの同時アクセスにも耐えうる可能性が示唆される.

今後は, 6 章で触れた第 1 バージョン MMI 実験システムとの性能比較を行なうべく, 本ライブラリを利用した第 2 バージョンの MMI 実験システムを構築した上で, ネットワーク上に分散する各種機器および Web サービスを透過的かつ統合的に制御できることを実証したい. なお, 想定される第 2 バージョン実験システムの概念図を図 6 に示す. 第 2 バージョン実験システムの詳細については, 著者らの既報[2] も参照いただきたい.

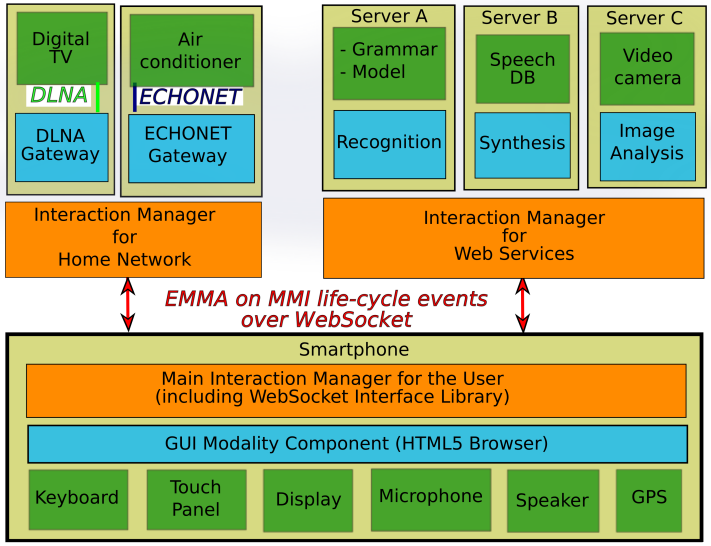

図 6 WebSocket による第 2 バージョン MMI 実験システム
Fig. 6 MMI Test System (Ver. 2) by WebSocket

Vol.2012-GN-82 No.22
Vol.2012-CDS-3 No.22
2012/1/20

10. おわりに

本稿では, まず Web 技術の特徴について簡単に触れた上で, 近年ますます高度化する Web アプリケーション開発にあたっての課題について概観した. またさらに, 様々な入出力機器を Web と透過的に統合するための枠組である MMI アーキテクチャについて概説するとともに, MMI-WG で取り組む MMI 実験システムの開発について触れた. その上で, 実験システム開発を通して明らかになった課題への対策として HTML5 の WebSocket 機能を MMI アーキテクチャへ組み込むために著者らが取り組んでいる「MMI on WebSocket ライブラリ」について報告するとともに, 今後の課題と対策案について議論した. 今後は, 上記ライブラリを利用した MMI 実験システムを構築した上で, ネットワーク上に分散する各種機器および Web サービスを透過的かつ統合的に制御できることを実証したい.

参考文献

1) W3C: W3C Top Page, available from ⟨http://www.w3.org/⟩ .
2) 芦村和幸, 一色正男, 中田潤也, 中島博敬, 小松健作: Web と機器を透過的につなぐ Multimodal Interaction フレームワークの構築, 2011-CDS-2-17 (2011).
3) Barnett, J., Dahl, D. A., Kliche, I., Tumuluri, R., Yudkowsky, M., Bodell, M., et al.: Multimodal Architecture and Interfaces (MMI Architecture), available from ⟨http://www.w3.org/TR/mmi-arch/⟩ (2011).
4) Maes, S. H., & Saraswat, V.: Multimodal Interaction Requirements, available from ⟨http://www.w3.org/TR/mmi-reqs/⟩ (2003).
5) Grifoni, P., et al.: *Multimodal Human Computer Interaction and Pervasive Services*, Premier Reference Source, IGI Global, available from ⟨http://www.igi-global.com/book/multimodal-human-computer-interaction-pervasive/787⟩ (2009).
6) Wikipedia: World Wide Web, available from ⟨http://en.wikipedia.org/wiki/World_Wide_Web⟩ .
7) Wikipedia: HTML, available from ⟨http://en.wikipedia.org/wiki/HTML⟩ .
8) W3C: HTML 4.01 Specification, available from ⟨http://www.w3.org/TR/html401/⟩ .
9) Wikipedia: HTML5, available from ⟨http://en.wikipedia.org/wiki/HTML5⟩ .
10) Hickson, I. & Hyatt, D.: HTML 5 A vocabulary and associated APIs for HTML and XHTML, available from ⟨http://www.w3.org/TR/html5/⟩, (2011).
11) Google: HTML5 ROCS, available from ⟨http://www.html5rocks.com⟩ .
12) Mozilla: Audio Data API, available from ⟨https://wiki.mozilla.org/Audio_Data_API⟩ .
13) Wikipedia: Event-driven programming, available from ⟨http://en.wikipedia.org/wiki/Event-driven_programming⟩ .
14) Johnston, M., Baggia, P., Burnett, D., Carter, J., Dahl, D. A., McCobb, G., et al.: EMMA: Extensible MultiModal Annotation markup language, available from ⟨http://www.w3.org/TR/emma/⟩ (2009).
15) AT&T: Speak4it 2.4 screencast, available from ⟨http://www.youtube.com/watch?v=mURHxA7sCmc⟩ (2011).
16) Openstream: Openstream's Cue-me Multimodal Browser available from ⟨http://www.youtube.com/watch?v=Orja2ZqEOKY&feature=related⟩ (2008).
17) SpeechTEK Europe: Multimodal Challenge, available from ⟨http://www.speechtek.com/europe2011/MultiModalChallenge.aspx⟩ (2011).
18) W3C: Multimodal Interaction Working Group, available from ⟨http://www.w3.org/2002/mmi/⟩ .
19) Wikipedia: Mashup, available from ⟨http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)⟩ .
20) Wikipedia: WebSocket, available from ⟨http://en.wikipedia.org/wiki/WebSocket⟩ .
21) Hickson, I. : The WebSocket API, available from ⟨http://www.w3.org/TR/websockets/⟩, (2011).
22) 小松健作: Wakachi demo (WebSocket による日本語テキスト分かち書きデモ), 入手先⟨http://www19072u.sakura.ne.jp/⟩ .
23) 学会試行標準 IPSJ-TS 0012:2010: マルチモーダル対話のための記述言語 Part1 要求仕様, 入手先⟨http://www.itscj.ipsj.or.jp/ipsj-ts/ts0012/toc.htm⟩ (2010).
24) Wikipedia: Node.js, available from ⟨http://en.wikipedia.org/wiki/Node.js⟩ .
25) Joyent, Inc.: Node.js official site, available from ⟨http://nodejs.org/⟩ .