HTML5 を用いた自動車サービス連携アーキテクチャの提案

2009SE111 河合恵里　2009SE181 村松貴介
指導教員：青山 幹雄

1　はじめに

現在，車載システムと外部ネットワークシステムが連携する自動車ネットワークサービスが実用化されており，SMD(Smart Mobile Device)などの端末と自動車がインターネットを通して直接通信を行うことやマルチメディア機能を統合したサービスの展開が期待されている．

本研究では，HTML5 の一部として策定された WebSocket や WebRTC(Real Time Communication)に注目し，プラットフォームに非依存かつリアルタイム性を重視するマルチメディアデータ通信を可能とする車載サービスブローカアーキテクチャを提案する．

2　研究課題

現在車載システムは外部ネットワークシステムと依存関係にあり，A 社の車載システムの場合，A 社のプロバイダのサービスのみ利用可能である[2]．そのため，SMD との連携や他のプロバイダのサービスを利用することが困難である．また，リアルタイム通信やストリーム通信を重要視するマルチメディアデータ通信を用いたサービスを実現することが困難である(図 1)．これらの問題を解決する必要がある．

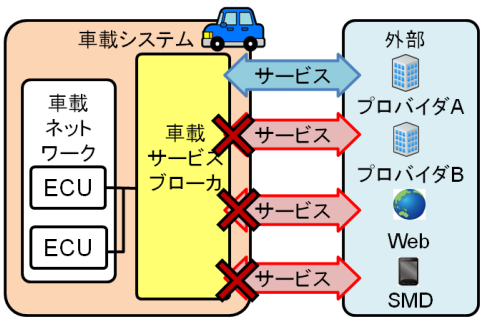

図 1 現在の車外サービス連携

3　関連技術

(1) HTML5

HTML5 は文書構造を定義するだけではなく，様々な要素を包括したアプリケーションレベルのプラットフォームである．

HTML4 と HTML5 が異なる点を以下に示す[3][4]．

(1.1) 新規タグ
(1.2) 動画，音声を HTML 文書内に取り組むことが可能
(1.3) WebSocket や WebRTC の通信プロトコルの追加
(1.4) 端末情報の利用が可能

(2) WebSocket

クライアントとサーバ間で双方向リアルタイム通信を実現する通信プロトコルである[6]．専用プロトコルとして双方向通信を目的とした WS(WebSocket)プロトコルが定義されている．

WebSocket は TCP プロトコルを用いて通信を行うため，時間制約よりも信頼性が重視される文字などをやり取りする通信に適している．

(3) WebRTC

Web ブラウザ上で双方向リアルタイム通信を実現するためのフレームワークである[7]．音声や動画のデータ転送プロトコルとして RTP(Real-time Transport Protocol)プロトコルを使用する．

WebRTC では UDP プロトコルを用いて通信を行うため，時間制約が重視される動画や音声などをやり取りする通信に適している．

(4) Ajax

Ajax(Asynchronous JavaScript and XML)とは，JavaScript の組み込みクラスである XMLHttpRequest を利用した非同期通信を利用し，Web ブラウザ上で既存の枠組みにとらわれないインタフェースを実現するための技術である．

4　アプローチ

本研究では，HTML5 の一部として策定された通信プロトコルである WebSocket や WebRTC を用いることでマルチメディアデータ通信を可能とする．

REST[3]，SOAP，WebSocket，WebRTC の 4 つの通信プロトコルを利用可能とし，プラットフォームに非依存かつマルチメディアデータ通信を可能とする新たな車載サービスブローカのアーキテクチャを提案する．各通信プロトコルの処理をパイプ＆フィルタのアーキテクチャパターンを用いて行うように設計する(図 2)．

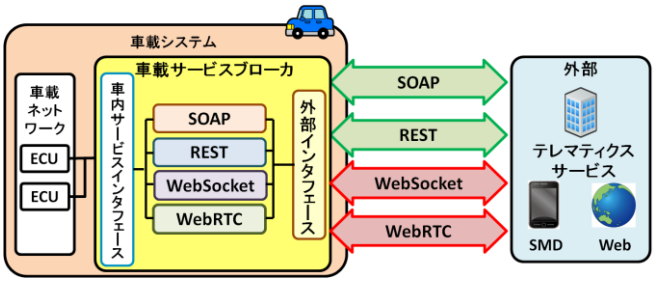

図 2　車外サービス連携モデル

5　提案アーキテクチャ

5.1　車載サービスブローカアーキテクチャ

SOAP, REST, WebSocket, WebRTC の 4 つの通信を可能とする車載サービスブローカのアーキテクチャを提案する.

提案するアーキテクチャは, ルーティング機能, プロトコル変換機能, 各プロトコルの処理機能などが含まれている. また, パイプ&フィルタのアーキテクチャパターンを用いており, (1)外部インタフェース, (2)それぞれのプロトコルの処理を行うパイプ, (3)車内サービスインタフェースから構成される.

提案する車載サービスブローカアーキテクチャの概要を図 3 に示す.

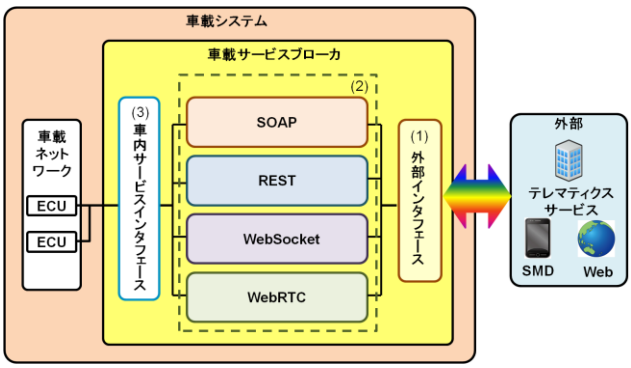

図 3　車載サービスブローカのアーキテクチャ

5.2　外部インタフェース

外部インタフェースは, ルーティング機能を含み, 外部との通信を一元的に管理するインタフェースである.

外部インタフェースの構造を図 4 に示す.

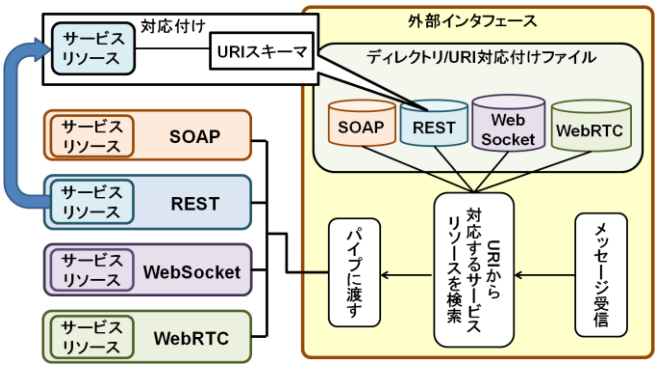

図 4　外部インタフェース

外部インタフェースの中に, リソース/URI 対応付けファイルを設け, 各プロトコルのサービスリソースに URI を対応付ける. 外部から処理の URI が車載システムに送信されるとリソース/URI 対応付けファイルの中から一致する URI を見つけ, その見つけた URI に対応するリソースが格納されている通信プロトコルのパイプに渡し処理が行われる.

5.3　各プロトコルのパイプ

各プロトコルの処理をパイプ＆フィルタアーキテクチャパターンを用いて実現する. このアーキテクチャパターンの利点としてパイプ機能の追加・変更が容易であるということが挙げられる. そのため, 新たな通信プロトコルを用いて車載システムと外部が連携する場合, その通信プロトコルに適したパイプを追加することで通信が可能となる.

パイプとは, 特定のプロトコルの処理を行うコンポーネントである. 外部からのメッセージを処理し, 車内 ECU に送信するメッセージを車内サービスインタフェースに渡す.

実現したパイプ＆フィルタアーキテクチャと WebSocket パイプの処理パターンを以下に示す(図 5, 図 6).

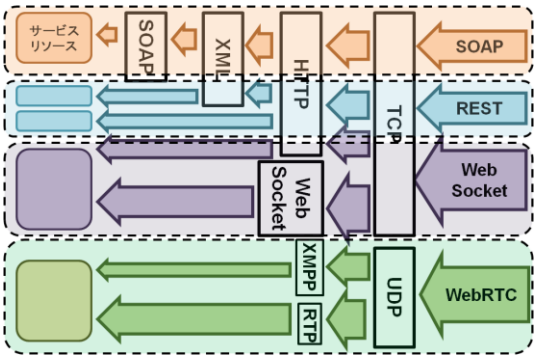

図 5　パイプ&フィルタ

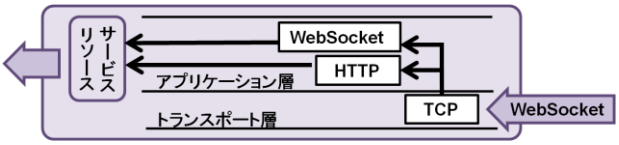

図 6　WebSocket のパイプ

5.4　車内サービスインタフェース

車内サービスインタフェースでは, パイプからのメッセージを車載ネットワークのプロトコルに変換, 各 ECU へルーティングをする.

車内サービスインタフェースの構造を図 7 に示す.

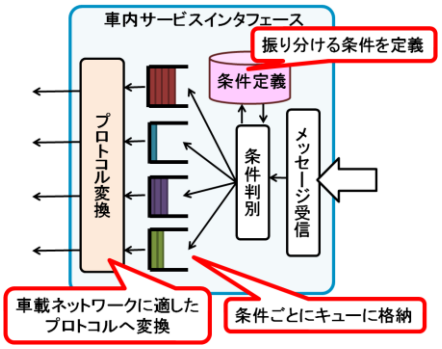

図 7　車内サービスインタフェース

車内サービスインタフェースの処理の流れを以下に示す.

(1)　それぞれのパイプからメッセージを受信する.
(2)　そのメッセージを条件判別する.
(3)　判別したメッセージを条件ごとにキューに格納する.
(4)　各キューから命令を取り出し, プロトコル変換を行い車内ネットワークへ送信する.

6　提案アーキテクチャのプロトタイプ

6.1　利用シナリオ

プロトタイプの利用シナリオとして，車が盗難された際など車の位置情報や車内の状況を把握することができる「自動車探索サービス」を選択する．このサービスのリソースは自動車内にあり，外部から自動車へのアクセスによりサービスが起動される．サービスの利用申請時には REST を用い，サービス起動後は WebSocket を用いてストリーム通信をする．

6.2　実装

プロトタイプでは GPS の代わりに時刻を，車載カメラ映像の代わりに画像ファイルを用いる．提案アーキテクチャに基づく車載システムのプロトタイプの構成を図 8 に示す．

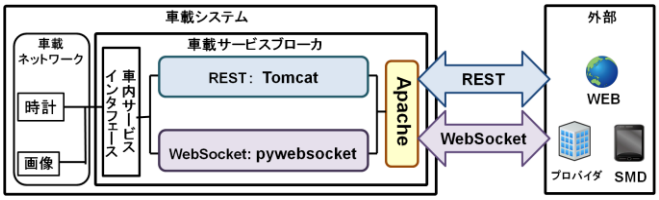

図 8 プロトタイプの構成

(1) Apache(Web サーバ)

外部インタフェースとして使用する．TCP と HTTP のフィルタの機能を持つ．

(2) Tomcat

REST 通信を実現する技術として用いる．開発言語は Java である．REST サービスのリソースを配置する．

(3) pywebsocket

WebSocket 通信を実現する技術として用いる．開発言語は python である．WebSocket のフィルタ機能を持つ．WebSocket サービスのリソースを配置する．

(4) 車内サービスインタフェース

プロトタイプは python で実装した．Tomcat と pywebsocket からのメッセージを車載ネットワークへ受け渡す．同様に，車載ネットワークからのメッセージを Tomcat と pywebsocket に受け渡す．

(5) 時計

プロトタイプは python で実装した．現在時刻取得リクエストを受け取り，値を返す．

(6) 画像

プロトタイプは python で実装した．画像情報取得リクエストを受け取り，画像を返す．

プロトタイプの実行環境を表 1 に示す．

表 1 実行環境

	車載システム	外部システム
OS	Windows Vista SP2	Windows XP SP3
CPU	Intel^{R} Core^{TM}2 Duo 2.10GHz	Intel^{R}Pentium^{R}4 3.20GHz
メモリ	4.00GB	1.00GB

6.3　プロトタイプの結果と振る舞い

プロトタイプを実装し，WebSocket によるストリーム通信の実現を確認した．

実装したプロトタイプの自動車探索サービスのシーケンス図を図 9 に示す．

WebSocket はクライアントとサーバでコネクションを形成し，任意のタイミングでメッセージを送信することが可能である．そのため，ブラウザから車載システムへ現在時刻や画像の取得リクエストを送信しなくても車載システム側の情報の変更に応じて，情報が送信される．

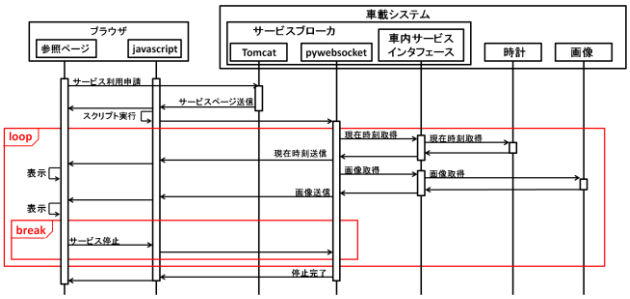

図 9 REST，WebSocket を用いた場合のシーケンス図

6.4　Ajax との比較

(1) WebSocket と Ajax の相違点

WebSocket と同様のリアルタイム通信技術である Ajax と比較を行う．

WebSocket と Ajax の主な相違点を表 2 に示す．

表 2 WebSocket と Ajax の相違点

	WebSocket	Ajax
使用プロトコル	WebSocket プロトコル	HTTP
サーバ push	可能	不可能
メッセージヘッダ	数バイト	数十~数百バイト

(1.1) 使用プロトコル

Ajax は HTTP を用いて通信をする．

WebSocket は専用の WebSocket プロトコルを用いて通信をする。.

(1.2) サーバ push

Ajax は HTTP でのリクエスト/レスポンス型通信をしており，サーバはクライアントからリクエストが来ないかぎりメッセージの送信が不可能である．

WebSocket はサーバとコネクションを形成するため，クライアントとサーバは任意のタイミングでメッセージを送信することが可能である．そのため，サーバからクライアントに対してストリーム通信を行うことが可能である(図 10)．

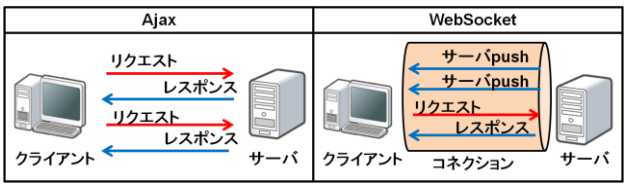

図 10 通信方式の違い

(1.3) メッセージヘッダ

Ajax は，メッセージには HTTP ヘッダが含まれる．HTTP ヘッダは数十から数百バイトを要する．

WebSocket はコネクション形成後 WebSocket プロトコルによるデータフレーム通信をしており，ペイロード以外のデータ量は数バイトである．Ajax に比べて通信のオーバーヘッドでが小さい．

(2) サービスへの比較

Ajax と WebSocket を車載カメラの映像を取得するサービスに適用した場合の振る舞いをシーケンス図を用いて比較する(図 11，図 12)．

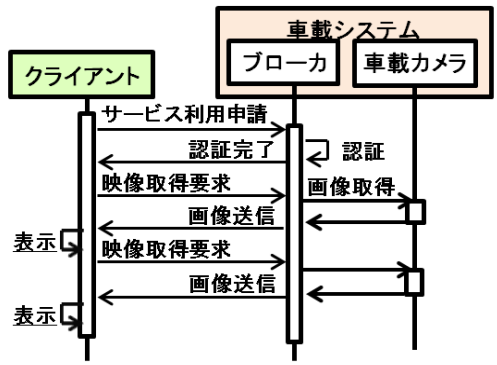

図 11 Ajax を用いた場合

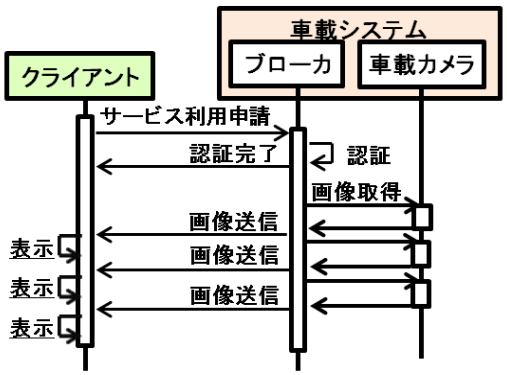

図 12 WebSocket を用いた場合

Ajax は，HTTP でのリクエスト/レスポンス型で通信を行っており，クライアントからリクエストがない限り映像を送信することができない．

WebSocket では，クライアントからリクエストが送信されなくても連続的に映像を送信することが可能である．

7 評価・考察

(1) サードパーティサービスの利用

現在の車載システムは，プラットフォームに依存しており独自のプロトコルにより通信を行っている．提案アーキテクチャでは，REST，SOAP，WebSocket，WebRTC といった標準プロトコルを用いることでサードパーティサービスの利用を可能にした．

(2) WebSocket と Ajax の比較

WebSocket と Ajax の比較より，WebSocket のマルチメディア通信に対しての優位性を示した．

(3) マルチメディアとアプリケーションサービスの融合

提案されている車載サービスブローカアーキテクチャでは，リアルタイムにマルチメディアとアプリケーションサービスを融合したサービスの提案がされていない．本稿では，HTML5 の一部として策定された WebSocket，WebRTC を用いることによりマルチメディアとアプリケーションサービスの融合を実現した．

8 今後の課題

今後の課題として，以下の 3 点が挙げられる．

(1) セキュリティの考慮

提案したアーキテクチャでは，セキュリティの考慮がされていない．外部ネットワークシステムと連携する場合，セキュリティを考慮する必要がある．

(2) 応答時間の計測

実装したプロトタイプの応答時間を計測し評価する必要がある．

(3) SOAP，WebRTC を用いたプロトタイプの実装

本稿のプロトタイプでは，WebSocket と REST を用いている．しかし，提案アーキテクチャでは，WebSocket，REST，SOAP，WebRTC の 4 つの通信に適さなければならないので，SOAP，WebRTC を用いたプロトタイプを作成し，評価をする必要がある．

9 まとめ

本稿では，プラットフォームに依存せず，WebSocket や WebRTC を用いてマルチメディアとアプリケーションサービスを融合したサービスを可能とする新たな車載サービスブローカのアーキテクチャを提案した．提案アーキテクチャでは，REST，SOAP，WebSocket，WebRTC の 4 つの通信を可能とする．

REST と WebSocket を用いたプロトタイプを開発し，Ajax を用いたときの場合と比較することにより，提案アーキテクチャの評価を行った．

参考文献

[1] M. Emmelmann and C. C. Kellum，Vehicular Networking，WILEY，2010.

[2] 濱千代 正弥，片桐 雅仁，自動車ネットワークサービスの連携アーキテクチャ 南山大学 2010 年度卒業論文，2011.

[3] 小松 健作，徹底解説 HTML5 マークアップガイドブック 最終草案対応版，秀和システム，2011.

[4] 中道 理ほか，HTML5 でつながる，日経エレクトロニクス，No. 1086，2012 年 7 月 9 日号，pp. 25~49.

[5] L. Ricbardson and S.Ruby，RESTful Web サービス，オライリー・ジャパン，2008.

[6] 田村 和也ほか，自動車技術，vol.65，No.2，自動車技術会，2011 年 2 月 1 日発行，pp. 40~45.

[7] W3C Working，http://www.w3.org/TR/webrtc.