RT ミドルウェアと Web アプリケーションの融合に関する研究

Research of Cooperation of RT-Middleware and Web Application

○　小島 一浩 (産総研)　正　ジェフ・ビグス (産総研)
正　安藤 慶昭 (産総研)　正　神徳 徹雄 (産総研)

Kazuhiro Kojima (AIST), k.kojima@aist.go.jp,
Geoffrey Biggs (AIST), Noriaki ANDO (AIST), Tetsuo KOTOKU (AIST)

We propose a new graphical interface that can manipulate RT Components on web browser. RT Middleware is implemented by using CORBA protocol, so that web browser can't communicate with CORBA Name Server and RT Components directly by using HTTP. As solution for this problem, we developed new module that can bridge NameServer and web browser. Furthermore, we implemented prototype web application that displays Name Service View on web browser.

Key Words: RT Middleware, Web Application, Graphical User Interface

1.　緒言

本研究では，RT ミドルウェア初心者においてもより簡単に操作ができることを目指し，Web ブラウザから RT コンポーネントを操作できる Web アプリケーションの開発を目的とする．

RT ミドルウエア (RT-Middleware: RTM) は，ロボット機能要素 (RT 機能要素) のソフトウエアモジュールを複数組み合わせてロボットシステム (RT システム) を構築する為のソフトウエアプラットフォーム一般の名称であり，OpenRTM-aist[1] は RT ミドルウエア実装の一つで，独立行政法人　産業技術総合研究所を中心に開発が進められている．RT ミドルウェアでは，RT 機能要素をソフトウェアモジュール化したものを RT コンポーネント (RT-Componet: RTC) と呼び，RTC とデータのやりとりをしたり，通信をしたりする為のポートと呼ばれるインターフェースを備えている．RTC 間のポート接続や，RTC 自体の活性化・不活性化は，eclipse 上で実装された RTSystemEditor (**Fig.**1) で，ドラッグ&ドロップにより直感的かつ容易に実行することが可能となっている．しかし，コンピュータスキルのあまり高くない学生や一般の人にも RT ミドルウェアを体験してもらいたいと考えた場合，RT ミドルウェアや eclipse のインストールがうまくいかず，RT ミドルウェアを体験する前段階において挫折する場合もままある．

一方，Web の世界においては Google の GMail や Google Maps に代表されるように，Web ブラウザ上で動作するスクリプト言語である JavaScript を利用することにより，Web ブラウザ上でデスクトップアプリケーションのように操作可能な Web アプリケーションが開発・提供されている．これらは，ユーザの操作や画面描画のバックグラウンドで，JavaScript における XMLHttpRequest API を用いて Web ブラウザと Web サーバ間で通信を行い，Web サーバから受信した XML データや JSON (JavaScript Object Notation) データをもとに，HTML を DOM (Document Object Model) として操作・更新することにより，サーバの存在を感じさせないシームレスな Web アプリケーションを実現させている．以上の技術は，AJAX[3] (Asynchronous JavaScript + XML) と呼ばれる．初期の AJAX 開発では，開発者が直接 XMLHttpRequest を用いて並列・非同期な JavaScript プログラミングを行うため，開発者の技量がかなり求められた．しかし，現在では一連の処理単位をライブラリとして提供した，prototype.js[4] や jQuery[5] ライブラリが存在するため，以前よりも容易に Web アプリケーションが開発可能となっている．

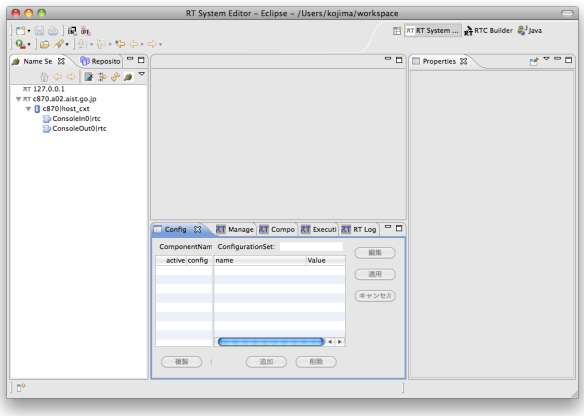


Fig.1 Screenshot of RTSystemEditor.

そこで，本研究では RT コンポーネントを操作する GUI アプリケーションを Web ブラウザ上で提供することにより，RT ミドルウェアと eclipse のインストールを必要とせず，Web ブラウザを立ち上げるだけで，RT コンポーネントの操作が可能な Web アプリケーションの開発を目的とする．提案する Web アプリケーションを用いたシステム全体の構成イメージを **Fig.**2 に示す．提案 Web アプリケーションの提供により，RT コンポーネントを操作する JavaScript コードは Web サーバから提供され，ユーザは Web ブラウザを起動するだけで，遠隔地に存在するロボットの RT コンポーネントを操作し，ロボット自体も操作が可能となる．このため，RT ミドルウェアと eclipse をインストールすることなく，RT ミドルウェアを Web 上で簡単に体験することが可能となる．また，既に RT ミドルウェアを利用している開発者においては，Web サーバをロボット内に組み込むことにより，Web ブラウザを用いてロボットのメインテナンスを簡単に行える利点がある．

本報告では，提案システムの設計を行い，一部機能を試験的に実装する．

2.　システム設計

本章では，提案 Web アプリケーションのシステム設計について述べる．

2.1　CORBA と HTTP

Fig.3 に示すように，RT ミドルウェアは RTC 間の通信に CORBA[8] を使用している．CORBA では，リモートオブジェクトの検索に CORBA NameService を使う．NameService を提供する NameServer への登録・問い合わせは，オブジェクト・リクエスト・ブローカー (Object Request Broker: ORB)

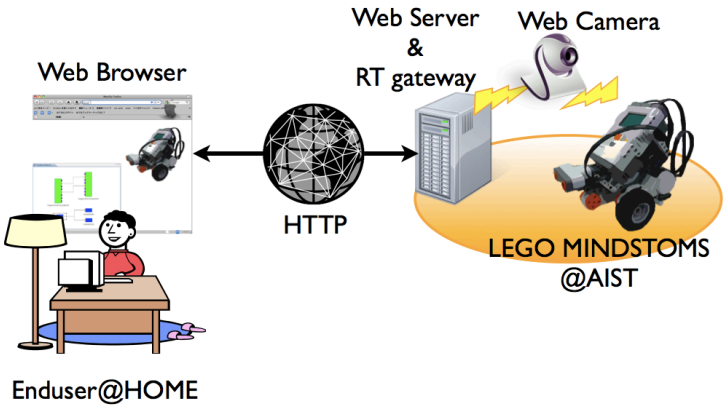


Fig.2 System Image.

を利用する．そのため，NameServer への問い合わせは HTTP で実行することはできない．そこで，新たに `rtctreedump` で HTTP と ORB をブリッジする．

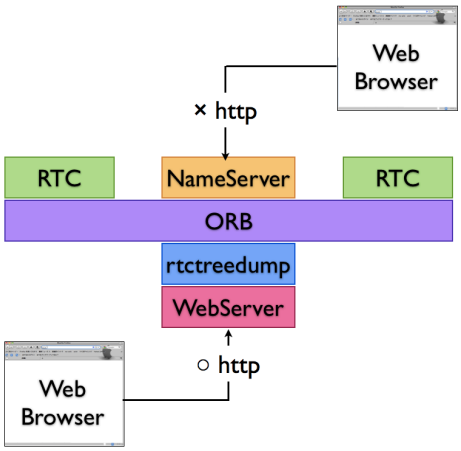


Fig.3 CORBA and HTTP.

2.2 Web アプリケーションフレームワーク

現在の Web アプリケーション開発では，テンプレート，セッション管理，データベースアクセス，O/R マッピングを提供する Web アプリケーションフレームワークを利用するのが一般的であり，PHP，Ruby，Python などの各種言語を対象にしたフレームワークがオープンソースとして提供されている．本研究では，Python 版の RT ミドルウェアが提供され，ツール群も Python で開発されているいることから，Python で開発されている Web アプリケーションフレームワークである Django[6] を採用した．

Django は，MVC (Model View Controller) モデルに非常に似た MVT (Model View Template) モデルを採用している．Model はオブジェクトに対応し，O/R マッピング (Object/Reletiona Mapping) を `model.py` に実装する．また，View はオブジェクトの操作を行いユーザに提示されるデータを作成し，`view.py` で実装を行う．Template 機能では，ユーザにどのようにデータを提示するかを規程する．

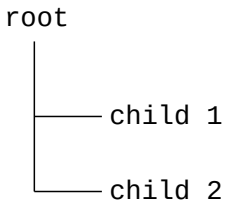
以上より，Web アプリケーション API を提供する関数を `view.py` 内に実装し，これを Web アプリケーション API を指し示す uri とバインドすることにより，API を外部に提供することが可能となる．バインドは，Django の `urls.py` で設定を行う．

3. 実装

本章では，先のシステム構成をもとにシステム提供機能の一部として，NameServer に登録されている RTC を Tree View で表示するネームサービスビューを実装する．

3.1 Tree View の実装：jsTree

Web ブラウザの JavaScript で木構造を操作・表示するために jQuery プラグインである jsTree[7] を用いて実装を行う．jsTree は，木構造を表した HTML，JSON，XML データを読み込み，木構造ファイルビューワのように表示することができる．本研究では，Web サーバからの受信データを軽量化するために，データ形式として JSON を採用する．**Fig.**4 に木構造とこれを jsTree が扱う JSON 形式に変換したデータを示す．JSON 形式では，ノードに子ノードがある場合，さらに再帰的に表すことが可能である．




```
{''data'':''root'',
 ''children'':[''child 1'', ''child 2'']}
```

Fig.4 Tree View and JSON Data Format.

3.2 Django における実装

RT ミドルウェアのツール群である `rtcrtee` は，NameServer に登録されている NameServer 名，コンテキスト，RTC を木構造として内部表現している．また，ターミナル上で RTC を操作できる `rtshell` では，`rtls -R` と実行することにより，先の `rtctree` を再帰的に辿りノード情報を取得可能とする．これらのツール群は Pyhton で実装されている．そこで，本研究では `rtls` コマンドを参考に，`rtctree` を JSON 形式にダンプする `rtctreedump` を新たに作成した．

Django の `view.py` における実装の擬似コードを **Fig.**5 に示す．この `get_tree` を例えば `http://web server/get_tree` とバインドすることにより API として外部に公開する．

```
from import rtctreedump import dump
from import rtctreedump import encode_tree

def get_tree(request):
 dump( tree)
 json =simplejson( tree,
                   default=encode_tree)
 return HttpResponse( json,
           mimetype='application/json')
```

Fig.5 Pseudocode of view.py

4. 実験結果

先の実装を検証するために，**Fig.**6 に示す検証環境を構築した．c870.a02.aist.go.jp では，Python 版 RT ミドルウェアに含まれるサンプルコードの ConsoleIn.py と ConsoleOut.py を実行する．Web サーバは，ラップトップ上の Django に含まれる開発用 Web サーバとし，localhost:8000 でアクセスする．localhost 上の環境変数 RTCTREE_NAMESERVERS は，localhost;c870.a02.aist.go.jp と設定する．

実行結果を **Fig.**7 と **Fig.**8 に示す．**Fig.**7 は Web サーバからの応答 JSON データで，再帰構造のため人間の可読性は良く無いが，プログラムとしては扱い易い構造となっている．また **Fig.**8 に示すように，eclipse 上の RTSystemEditor と同様のネームサービスビューが実現できた．

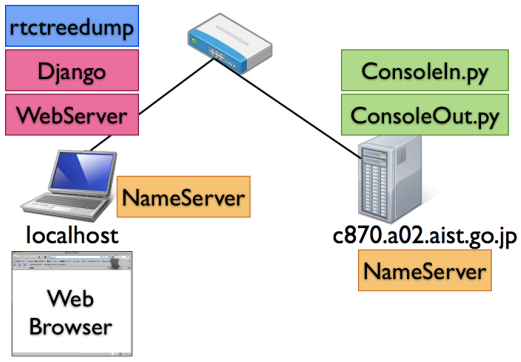


Fig.6 Test Environment.


```
{"state": "open", "icon": "folder",
"data": "/", "children": [{"state":
"open", "icon": "folder", "data":
"c870.a02.aist.go.jp", "children":
[{"state": "open", "icon": "folder",
"data": "c870.host_cxt","children":
[{"icon": "folder", "data":
"ConsoleIn0.rtc", "children":[],
"attr": {}}, {"icon": "folder",
"data": "ConsoleOut0.rtc","children":
[], "attr": {}}], "attr": {}}],
"attr": {}}, {"icon":"folder", "data":
"127.0.0.1", "children": [], "attr":
{}}], "attr": {}}
```


Fig.7 Received JSON Data.

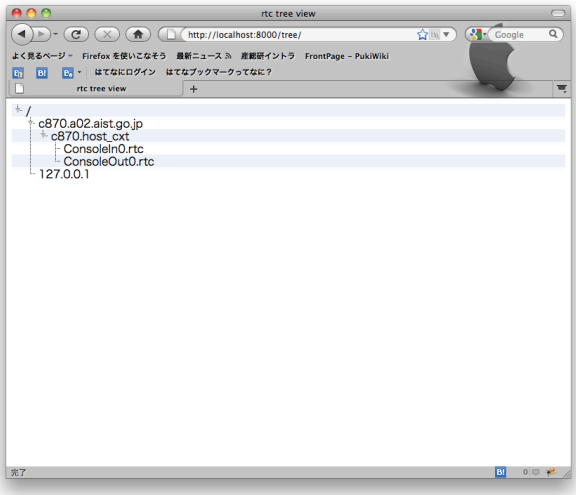


Fig.8 Screenshot of Web Browser.

5. 今後の改良

Web ブラウザと Web サーバ間の接続は，一回のリクエスト毎で切断されてしまう．そのため，NameServer に登録されている RTC の状態が変化しても，Web ブラウザがリロードされない限り View も変化しない．この問題に対しては，jQuery の$ajax 関数を使ってバックグラウンドで定期的にリロードすることにより解決できる．しかし，リロードの周期間隔をどの程度に設定しておけば良いかなどのノウハウが必要になってくる．

そこで本研究では，サーバ・プッシュ技術を利用することを予定している．サーバ・プッシュ技術とは，Web ブラウザと Web サーバ間を常に接続しておき，任意のタイミングでサーバ側から Web ブラウザに情報を Push する技術である．しかし，現在普及している Web ブラウザは，接続を常時維持することはできない．そこで，Web ブラウザがリクエストで接続を確立したら，サーバはその接続を離さずブロック状態を維持し，サーバ側のデータベース等の状態が変化するなど任意のタイミングで情報を提供し，一旦接続を終了する．Web ブラウザは接続の切断と同時に，即座に AJAX によりバックグラウンドでリクエストを発行することにより接続を回復する．以上の手順により，擬似的に常時接続状態とする．このように現在普及しているブラウザに擬似的に Push する技術を Comet[9] と呼ぶ．Comet の通信方法を **Fig.**9 に示す．

Comet は先の説明のように，あくまでも擬似的に Web ブラ

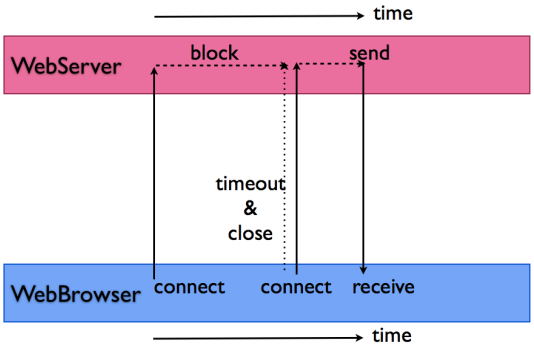


Fig.9 Comet Server Protocol.

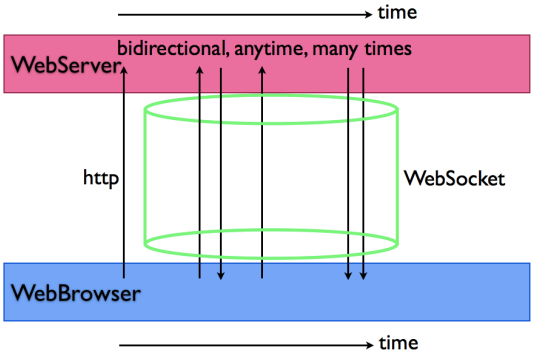


Fig.10 WebSocket Protocol.

ウザと Web サーバを常時接続状態とみなせるようにする技術であり，その過程は複雑で現在あまり普及はしていない．最近，真に Web ブラウザと Web サーバ間を常時接続にし，サーバ・プッシュを実現する HTML5 API して，WebSocket[10][11] が API として W3C により規格化され IETF によりプロトコルが策定されている．WebSocket はもともと次世代 Web 規格である HTML5 の機能であったが，その後単体機能としてスピンアウトし策定が行われている．WebSocket では，Web ブラウザは始め HTTP で Web サーバに接続要求を出し，その後 WebSocket プロトコルで常時接続状態を実現する．これにより Web サーバは任意のタイミングで情報を Web ブラウザに送信することが可能となる．情報を受信した Web ブラウザは，従来のように JavaScrip で受信データを加工してビューを Web ブラウザに反映させる．WebSocket の通信方法を **Fig.**10 に示す．このように WebSocket の場合，Comet よりも通信方式が単純で Web アプリケーションの開発が単純化できるが，現在普及している Web ブラウザで WebSocket に対応しているブラウザはほとんどなく，次期バージョンの Web ブラウザで対応予定のブラウザがほとんどである．しかし，WebSocket の機能を Flash で実装した `web-socket-js`[12] が OSS で公開されている．そこで本研究では，WebSocket での双方向通信を実現することを予定している．

6. 関連研究

本研究では，Web ブラウザ上で遠隔地の RTC をインターネット経由で操作することを目指している．類似の既存研究として，Virtual Collaboration Arena: VirCA[13] がある．VirCA は，インターネット上の多地点に分散した仮想空間を結び付け、各地点のマニピュレータをはじめとする様々な種類のロボットハードウエアやソフトウエアを仮想空間上で相互作用させるための仮想環境を提供することを目指している．コア技術として，OpenRTM-aist のデータポートの通信を The Internet Communication Engine: Ice[14] で行うように拡張し，

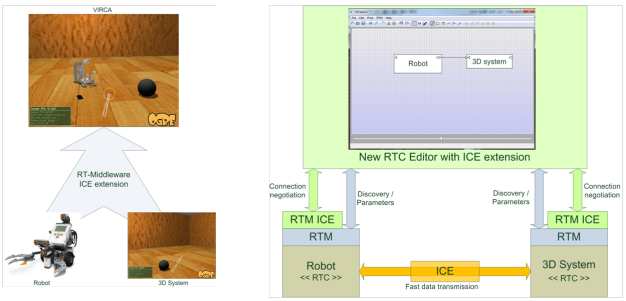


Fig.11 System Image of Virtual Collaboration Arena: VirCA.

インターネット上での高速な通信を実現している．Ice は開発言語として，C++,Java,.NET,Python,PHP,Ruby,Objective-C に対応している．Ice が目指すのは遠隔地間のオブジェクト呼び出しを，インターネット上での Ice 通信で実現することである．これに対し本研究が目指すのは，RTC は一ヶ所に存在し，RTC の操作を行うユーザインターフェースを遠隔地における Web ブラウザで実現することを目指している点が異なる．ただし，Ice は Object-c に対応しているので，iOS ネイティブアプリケーションとしてユーザインターフェース込みで開発することが可能であるが，iOS 限定となる問題がある．これに対し本研究では，Web ブラウザをインターフェースのフロントエンドとして使用することを目指しているため，各種 OS で使用可能という利点がある．

7. 結言

本研究では，RT ミドルウェア初心者においてもより簡単に操作ができることを目指し，Web ブラウザから RT コンポーネントを操作できる Web アプリケーションの開発を目的とした．本報告では提案システムの設計と一部機能の実装例を紹介した．今後，RTSystemEditor の各種機能を Web アプリケーションとして実装していく予定である．

参后献

[1] OpenRTM-aist: http://www.openrtm.org/
[2] Eclipse: http://www.eclipse.org/
[3] J.J.Garrett, “Ajax: A New Approach to Web Applications” (2005) http://www.adaptivepath.com/ideas/essays/archives/000385.php
[4] prototype.js: http://www.prototype.js/
[5] jQuery: http://jquery.com/
[6] Django: http://www.djangoproject.com/
[7] jsTree: http://www.jstree.com/
[8] CORBA: http://www.corba.org/
[9] Comet: http://cometdaily.com/
[10] WebSocket API: http://dev.w3.org/html5/websockets/
[11] WebSocket Protocol: https://datatracker.ietf.org/wg/hybi/
[12] web-socket-js: https://github.com/gimite/web-socket-js/
[13] VirCA: http://virca.hu/
[14] ice: http://www.zeroc.com/