

# Ruby によるプッシュ型通信のための WebSocket ライブラリの開発

古谷 崇

松原 俊一

Martin J. Dürst

青山学院大学理工学部情報テクノロジー学科

## 1 はじめに

近年、ネットワークの通信速度が増加の一途をたっている。しかし、チャットなどリアルタイム性が求められるプログラムに関しては、サーバの情報をクライアントへすぐに反映させることが求められる。Ajax を使用すればページの一部を再読み込みはできるが、短い間隔で HTTP 接続をしているに過ぎない。そこでプッシュ通信型の接続方式が重要である。現在、プッシュ通信には Comet という技術がよく使われている。しかしこの技術は HTTP のサーバからのレスポンスを遅延させて使用するものである。レスポンスを遅延させて使用する方法是 HTTP 本来の使用方法ではない。そこで提案されたのが WebSocket プロトコルである。このプロトコルでは、リアルタイムで更新が反映される。WebSocket はサーバ、クライアント間で双方向プッシュ型通信を行い、リアルタイムな更新を実現するプロトコルである。

本研究では WebSocket を利用したサーバを開発するための Ruby 用ライブラリを作成した。本ライブラリは WebSocket の低水準の機能を実装した部分とアプリケーション開発のための部分からなる。

## 2 WebSocket

### 2.1 WebSocket プロトコル

WebSocket [1] は Internet Engineering Task Force (IETF) によって議論されているプロトコルである。WebSocket はサーバとクライアント間でコネクションをむすび双方向のプッシュ型通信を行うためのプロトコルである。

### 2.2 WebSocket API

WebSocket プロトコルを Web ブラウザで使用するための API [2] が提供されている。この API によりブラウザ上で WebSocket を利用することが可能になる。

### 2.3 対応状況

現在、Websocket は Google Chrome や Safari などの Web ブラウザで実装されている。

しかし、プロトコルの変更があり最新の WebSocket プロトコルに対応していないので使用する際には特に注意が必要である。また Firefox と Opera ではセキュリティ上の問題が解決するまで機能を無効化した。

### 2.4 他のプッシュ型通信に対する優越性

複雑性が低く、プッシュ型、双方向、低いレイテンシでの通信ができることが WebSocket の特徴である。この中でもプッシュ型通信は Comet などを実装されていたが、Comet は図 1 の上部のとおり HTTP のレスポンスを遅延させる使用方法をしている。一方、WebSocket では図 1 の下部のとおりハンドシェイクを行いコネクションが確立すると、以後はフレームを作成し小さなヘッダ情報を付加することで通信を行うことができる。これにより余分なサーバ資源、特に HTTP 接続を使用せずに通信を行うことができる。

## 3 ライブラリの構成

本研究のライブラリは 2 層構造になっており基礎的な部分と応用部分に別れている。基礎ライブラリを独立させ WebSocket プロトコルの変更に対応できる。サーバとクライアントに共通する処理は基礎ライブラリに集約した。独自の WebSocket のソケットプログラムを作成したい場合には基礎ライブラリを使い、サーバを作成することができる。応用ライブラリではチャットサーバなどの実用的な実装をした。機能別にライブラリを作成し利用できるようにした。

### Development of a WebSocket Library for Push Type Communication in Ruby

Takashi Furuya, Shun-ichi Matsubara and Martin J. Dürst  
Department of Integrated Information Technology, College of  
Science and Engineering, Aoyama Gakuin University  
5-10-1 Fuchinobe, Chuo-ku, Sagami-hara-shi, Kanagawa 252-  
5258, Japan  
furuya@sw.it.aoyama.ac.jp,  
{matsubara, duerst}@it.aoyama.ac.jp

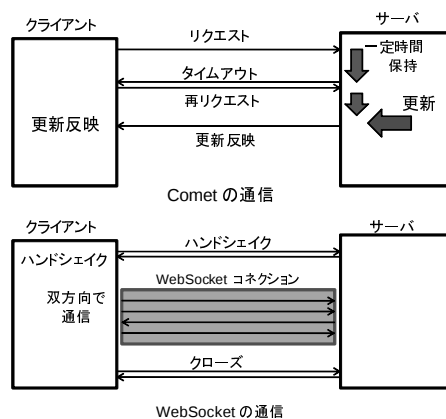


図 1: Comet と WebSocket の比較

#### 4 基礎ライブラリ

基礎ライブラリでは基本的な WebSocket の通信を制御している。基礎ライブラリでサーバとクライアント両者に対応している。

WebSocket の通信を開始するためにはハンドシェイクと呼ばれる通信が必要になる。この通信には key のやりとりが必要になる。key のやり取りが複雑なため基礎ライブラリで処理している。ハンドシェイクを行う際に記述するプログラムをとて少なくすることができる。

またデータを送信する際に必要なヘッダ情報の付加が必要になるが、その処理もこのライブラリ内で実装する。これにより送信するデータを指定すれば送信することが可能である。

#### 5 応用ライブラリ

応用ライブラリはチャットの実装などに用いられる。処理の違いにより複数のライブラリを作成し、用途により使い分けられる。

エコーサーバは主にサーバやクライアントのテストに用いる使用を想定している。エコーサーバはクライアントからの接続要求を受け入れ、クライアントから送られてきた文字列をそのまま送り返す。クライアント同士の通信はできない仕様になっており、サーバとクライアントの一対一で通信を行う。このライブラリを使用すると、サーバ URL、ポート番号を指定するだけでサーバを起動させることができる。

チャットサーバライブラリはチャットサーバを作成す

るためのものである。クライアントからの書込みがあるとすべてのクライアントへ向けてデータが送信される。新規のクライアントが接続されると過去 10 件の履歴を送信する。このライブラリもサーバ URL、ポート番号を指定するだけで動作する。しかし基本的なチャット機能であるのでチャットルームなどの実装はユーザ自身が自由に実装する。このライブラリを使用することで WebSocket を簡単に利用できる。

また、Ruby によってテスト用クライアントを作成した。現段階でクライアント側になる Web ブラウザはいくつか存在するが最新の WebSocket プロトコルに対応しているものはない。そこでサーバをテストする際にクライアントの用意が必要になるので、クライアントも Ruby で実装する。このクライアントではコマンドライン上でのテストを前提とする。サーバ側と同じ基礎ライブラリを使用し通信を行う。入力された文字列をサーバへ送信し、サーバから送られてきたデータを画面に表示させる。テストをするために必要な機能は実装しており、デバッグモードを使用し通信の詳細を表示させることもできる。

#### 6 おわりに

Web サービスを運用する上で今後必要になってくるのは、更新をいち早くクライアントへ伝える手段である。通信速度の増加に対応して通信方法も変えていかなければならない。チャットプログラムによる HTTP 接続を少なくし、サーバの負荷を抑え、通信を円滑にすることが必要である。WebSocket プロトコルの仕様は未だに決定しておらず、脆弱性も見つかっており、日々改良されている。それでも WebSocket プロトコルが完成すれば有用な技術になる。このライブラリによって WebSocket による通信の利点を知ってもらい WebSocket が普及すれば幸いである。

#### 参考文献

- [1] Ian Fette. draft-ietf-hybi-thewebsocketprotocol-04. <http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-04>, 2011.
- [2] Ian Hickson. The Websockets API, W3C Editor's Draft. <http://www.w3.org/TR/websockets/>, 2010.