3U-06

WebRTCを用いたブラウザ上でのコンテンツ配信の実装と評価

成田 裕司 † 坪川 宏 †
† 東京工科大学　コンピュータサイエンス学部

1　はじめに

近年，Webアプリケーションの開発が盛んになっている．特にJavaScriptは，様々なAPIや来無頼が続々と登場し，アプリケーション開発に活躍される範囲が広がっている．JavaScript APIの1つであるWebRTCを用いることでブラウザ単体によるブラウザ間通信が行えるようになった．

しかし，WebRTC自体は注目されているものの，活用検証についての報告はほとんどない．また，複数ノードからなるネットワークを構築した際の検証もあまり行われていない．

本研究では，WebRTCのDataChannel機能を使用し，P2Pネットワークの構築を行う．構築アルゴリズムにおいては，WebRTC+JavaScriptのみで実装するため，制御用コネクションを使用せず，最小限のコネクション数で行うことが望ましい．そのため，ルーティングテーブルの更新用の通信が不要であり，アルゴリズムがシンプルである「Kademlia」を用いることでP2Pネットワークの構築を実現する．

2　動作環境

本研究では，Google Chrome上で開発を行っている．

Chromeでは，ECMAScript 6や最新のJavaScript APIに対応していることから，本システムのプログラム実装に適していると判断した．

3　システム概要

本システムはJavaScriptとWebRTCおよびPeerJSを用い，Kademliaを実装し，ブラウザ上でP2Pネットワークの構築を行う．また，構築されたP2Pネットワークはコンテンツ配信を目的として使用されることを想定している．ネットワーク概念図を図1を示す．

Kademliaのノード IDのハッシュサイズは160bitとなっているが，PeerJSから受け取る動的なIDが14文字から16文字であり，本システムでは，JavaScriptで実装することや，より大規模な使用の想定はしていないため，想定ノード数がカバーできるということで，本システムのノード IDのハッシュサイズを128bitとした．この値はJavaScriptの処理時間などの関係から最適化が必要である．

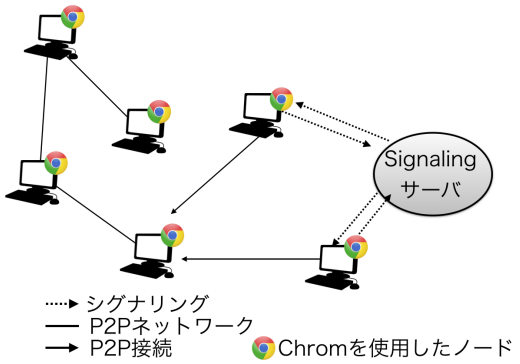

図 1: WebRTC ネットワーク

本システムのKademlia部，通信部，コントローラ部の説明を行う．Kademliaのノード IDはPeerJSから得られるPeerIDをノード IDとして使用している．本来であればPeerIDはKademliaのIPアドレス，ポート番号のタプルに相当するが，PeerJSを使用することで動的な乱数IDを得られるため，初回時のみこのIDを元にノード IDとして，保存し，それ以降は保存されたPeerIDをノード IDとして使用する．

各ノード情報として格納する値はPeerID，NodeID，LifeTimeを各ノード情報としてオブジェクトに格納している．このノード情報をK-bucketリストに格納していく．K-bucketリストにはインデックス情報として自身のノード IDからの距離となるXORの値を持っている．このインデクス情報をまとめたものをルーティングテーブルとして扱っている．ルーティングテーブルの概念図を図2に示す．

Kademliaには PING，FindNode，FindValue，Storeの問い合わせ方法があり本システムでもこの4種類の問い合わせ方法を実装している．また，Kademlia部はフレームワーク化されており，Kademlia部が記述されたオブジェクトを呼び出すことで使用が可能である．

本システムでは，送信するデータ全てオブジェクト型で送受信を行っている．オブジェクト内は問い合わせ方法であるRPCと通信に必要になるPeerID，送信もとである自身ノード IDを必ずオブジェクト内に格納している．それらに追加してKeyとValueをセット

Implementation and Evaluation of CDN on the browser using WebRTC
† Yuji Narita
† Hiroshi Tubokawa
School of Computer Science, Tokyo University of Technology (†)

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.

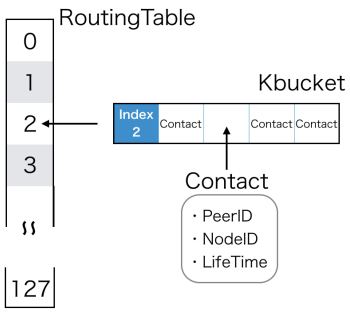

図 2: ルーティングテーブル

することにより，ノードの検索や Value の検索を行う．

コントローラ部では，接続処理でユーザからの PeerID や Key の入力が必要になる．また，FileAPI を使用し，データの送信を行うため，データ取得部分などユーザ側からの操作をトリガーとし，値の取得する部分を扱う．また，受け取った値を Kademilia 部や通信部に値を渡す部分である．

4 接続

PeerJS では，接続するために他の PeerID を使用し，Signaling サーバを経由した後にブラウザ同士の接続を行う．そこで，個々のブラウザに与えられるオリジナルな ID をなんらかの方法で入手しなければ，本システムのネットワークに接続することができない．

5 システム実装

JavaScript と WebRTC および JavaScript API のみでシステムの実装を行った．本システムの Kamdemlia は PeerJS を使用し，接続や通信部分，IndexedDB を使用し，データベース部分を実装した．ルーティングテーブル各機能，FindeNode は JavaScript ではスレッド化ができないため，主に Lookup 処理については所持ノード数が大きくなるにつれて処理時間がかかってしまう．そのため，一定間隔で処理を行い続けるようになっている．

実装後，システムを検証を行った．Kademlia の基本的な機能である 4 つ問い合わせとルーティングテーブル部分はすべて行えたことを確認した．また，FindeValue と Store など送受信，書き込みが同時にも行えることも確認した．

6 WebRTC 評価検証

WebRTC の評価検証を実装したシステムで行った．検証用に IndexedDB から SessionStorage に書き換えるなどの変更を加えている．また，音声通信の実装も行い，音声の通信の検証ができるよう Web Audio API を実装した．

WebRTC および，PeerJS は，UDP を使用しているが，Chrome にて実装を行っているため，SCTP をサポートしている．また，PeerJS を使用しているため，Chunk 機能が既に実装されている．

検証については，本システムのコネクションからデータチャンネルの確立までの時間とデータ通信の完了時間の計測を行った．コネクション検証では PeerJS で提供されている Peer サーバを使用した．接続時間は 0.5 秒から 2 秒程度となった．この差は連続で計測を行った場合，接続間隔が短くなると接続完了の時間が短くなるためである．次にデータ通信時の遅延検証を行った．7.5MB の JPEG 画像を STORE，FindValue を行う実験グローバル間，ローカル間で行った．グローバル間が平均 6.35 秒であった．ローカル間では，平均 3.02 秒となった．音声データ検証では MP3 ファイルの送信および再生を行った．画像送信検証でも起きたが，10MB 以上のデータの通信を行おうとする場合，コネクションが切断されてしまう不具合が起きた．そこで，あらかじめ，個別に自身で Chunk を 512KB に分割することで対処した．これは音声データを送信した場合，Chunk サイズを小さくしすぎると音飛びや順序，ピッチが安定せず，もっとも安定して音声が流れたサイズが 512KB であった．

7 JavaScript 負荷検証

Lookup 処理の処理時間を計測した．本システムでは他の処理ができるようにしているため，処理が一定に行っていない．setTimeout 関数などを使用し，遅延実行処理を行い，この状況の回避を行っている．

計測した結果，Lookup 処理中の接続時間込みで 1〜30 台までの接続時に行われる Lookup 処理の計測を行った．計測の結果，ノード数の乗数分の処理時間がかかることがわかった．

8 まとめ

今回，WebRTC を用いた P2P ネットワークの構築検証を行った．データ通信の接続時間，遅延の検証，JavaScript での Kademila の処理速度の計測などからネットワーク部分よりも JavaScript の処理時間が問題になることがわかった．今後は，アルゴリズムの最適化やユーザ操作の妨げにならない工夫が必要である．

参考文献

[1] P. MAYMOUNKOVKademlia: A Peer-to-peer Information System Based on the XOR MetricPeer-to-Peer SystemsProc. the 1st International Workshop on Peer-to-Peer SystemsIPTPS'0253-65Mar. 2002

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.