WebRTC調査

WebRTC概要

WebRTCとは

- Web Real-Time Communicationの略称
- Webブラウザ、モバイルアプリでリアルタイムP2P通信を実現
    - データ、オーディオ、ビデオ etc.
- pureなjsだけで実現可能
    - 使える機能等はブラウザ依存あり
    - adapter.jsを用いた一般化も検討必要
- WebRTCの主な流れとしては
    i. シグナリング
        - WebRTCを行うための必要な情報 (IP、メディアの種類等)を共有
            - = SDPによって規定されたsession description

    ii. P2P通信

SDP (Session Description Protocol)とは

- RFC4566で規定
- session descritionに関するプロトコル
  - session descriptionはWebRTC通信の設定
    - P2P通信するために事前に両者が知っていなければならない情報
  - 送信するメディアの種類、フォーマット、プロトコル、IP、ポート etc.

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

シグナリングサービスとは

- WebRTCコネクションを作成するためにsession descriptionを共有する方法の総称
  - emailでも、伝書鳩でも共有できればなんでもいい
- シグナリングを行う上で難しいのがピア同士のグローバルIPの取得
  - いわゆるNAT超え

NAT超えとシグナリング

STUN(Session Traversal Utilities for NAT)+シグナリング

- シナリオ
  - 登場人物はピア1とピア2とSTUNサーバ
  i. STUNサーバは、ピア1のNATによって割り振られるグローバルip+portをピア1に返す
  ii. STUNサーバは、ピア2のNATによって割り振られるグローバルip+portをピア2に返す
  iii. シグナリング
  iv. P2Pを実現
- NATの種類によってはブロックされる
  - 知らないグローバルipからのアクセスをブロックするタイプ

TURN (Traversal Using Relays around NAT)

- シナリオ
  - 登場人物はピア1とピア2とTURNサーバ
  i. ピア1はクレデンシャルを用いてTURNサーバで認証
  ii. TURNがピア1とピア2のパケットを仲介
    - レスポンスの送信元がTURNサーバ (既知のグローバルip)となる
    - 知らないグローバルipからのアクセスをブロックするタイプを通過
- FWの設定次第で弾かれる可能性がある
  - 特定のUDPポートを使うため
- 最も確実にP2Pを行いたいのであればTURN over TCP
  - 特定のTCPポートを使う (80番を使える)
- TURNサーバの負荷が大きい＋帯域を食う

ICE (Interactive Connectivitty Establishment)

- STUNとTURNのどちらかを用いるか、プロトコル(UDP or TCP)をどうするかを決定するための手法
- MDNによると以下を試して、ピアを接続するために最低遅延の手法を探す (ほんまか？)
    i. 直接UDP接続(STUNサーバを使用してグローバルip取得+シグナリング)
    ii. HTTPポート経由の直接TCP接続
    iii. HTTPSポート経由の直接TCP接続
    iv. TURNサーバ経由の間接接続

jsの実装フロー ~シグナリング編~ (MDN参考)

1. 送信者が `MediaDevices.getUserMedia` でメディアタイプ設定
2. 送信者が `RTCPeerConnection` の作成+ `RTCPeerConnection.addTrack()` 呼び出し
3. 送信者が `RTCPeerConnection.createOffer()` を呼び出し
4. 送信者が `RTCPeerConnection.setLocalDescription()` で自身のsession description記述
5. signaling+受信者が受け取ったsession descriptionを `RTCPeerConnection.setRemoteDescription()`
6. 受信者は `RTCPeerConnection.addTrack()` の情報をもとにメディアをattach
7. 受信者は `RTCPeerConnection.createAnswer()`
8. 受信者は `RTCPeerConnection.setLocalDescription()`
9. 受信者がanswer+送信者はanswerをもとに `RTCPeerConnection.setRemoteDescription()`

jsの実装　コメント

- 要はお互いのlocal, remote descriptionさえ設定できればP2Pできるようになるんじゃないのと思うが確証はない
    - そんな実装例が出てこない
- local descriptionが、事前に決めておけるのかイマイチ不明
    - 実装例をそれほど眺めていない

WebRTCのP2P通信

WebRTCのP2P通信

- jsの実装上の分類
    - プロトコルが違うかはしらん

1. MediaStream
    - 動画や音声
2. DataChannel
    - データ

WebRTCのdata channel

- example

```
var pc = new RTCPeerConnection();
var dc = pc.createDataChannel("my channel");

dc.onmessage = function (event) {
  console.log("received: " + event.data);
};

dc.onopen = function () {
  console.log("datachannel open");
};

dc.onclose = function () {
  console.log("datachannel close");
};
```