Plugin-free realtime communication

WebRTC : Real-Time Communication for Web

WebRTC is a technology which provides browsers and mobile applications with Real-Time Communication capabilities with the help of its components.

WebRTC is a collection of communication protocols and application programming interfaces (APIs).

WebRTC is a cross-browser and cross-platform technology.

Supported browsers and platforms

WebRTC does a lot of things :

- Get streaming audio, video or other data.
- Get network information such as IP addresses and ports,
  and exchange this with other WebRTC clients (known as *peers*) to enable connection,
  even through NATs and firewalls.
- Coordinate signaling communication to report errors and initiate or close sessions.
- Exchange information about media and client capability, such as resolution and codecs.
- Communicate streaming audio, video or any type of arbitrary data.

The WebRTC APIs

WebRTC performs three main tasks –
- Acquiring audio and video
- Communicating audio and video
- Communicating arbitrary data

which are performed by WebRTC three main APIs –
- getUserMedia
- RTCPeerConnection
- RTCDataChannel

getUserMedia

- capture audio and video
- obtain a media stream with *navigator.getUserMedia*

RTCPeerConnection

- Audio and video communication between peers
- Signaling
- Codec Handling
- Security
- Bandwidth management

RTCDataChannel

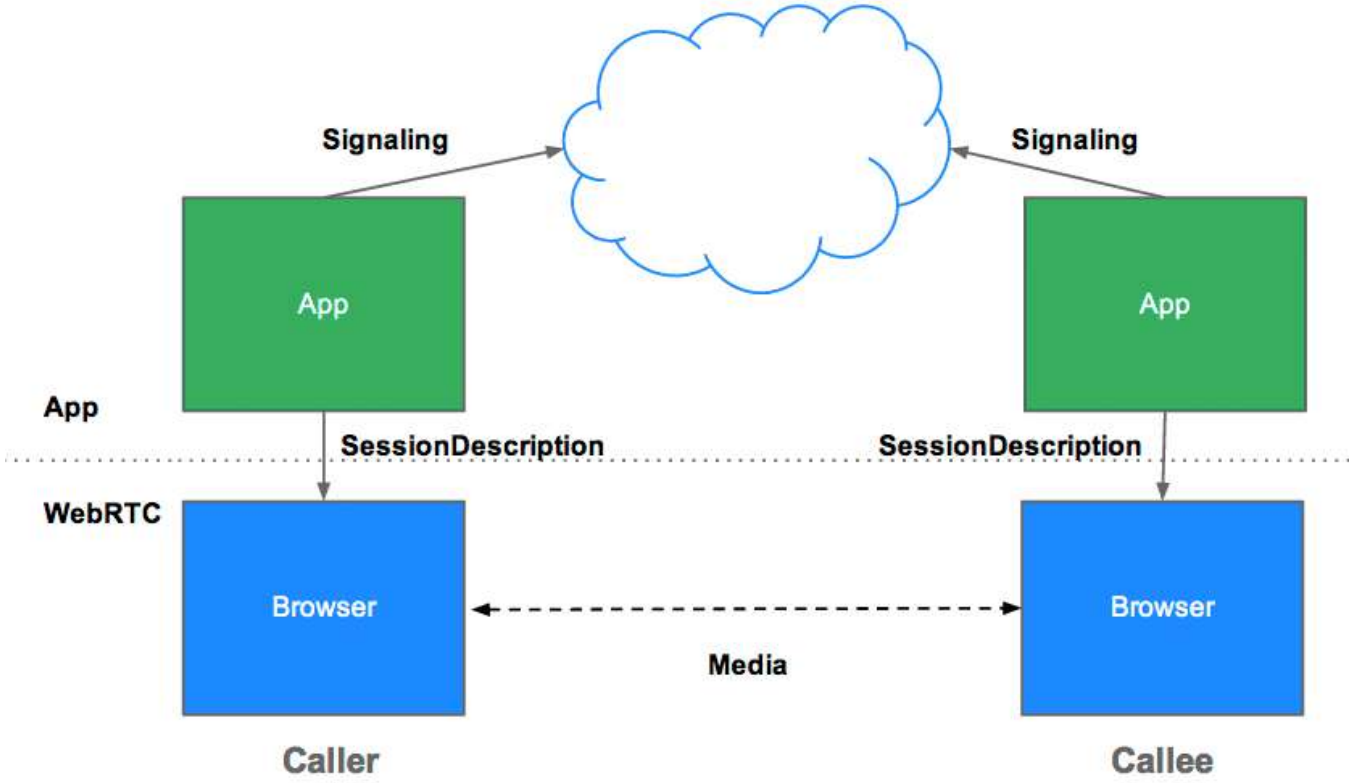
- Bidirectional communication of arbitrary data between peers
- uses RTCPeerConnection

Signaling

WebRTC uses RTCPeerConnection to communicate streaming data between peers,
but also needs a mechanism to coordinate communication and to send control messages,
a process known as signaling

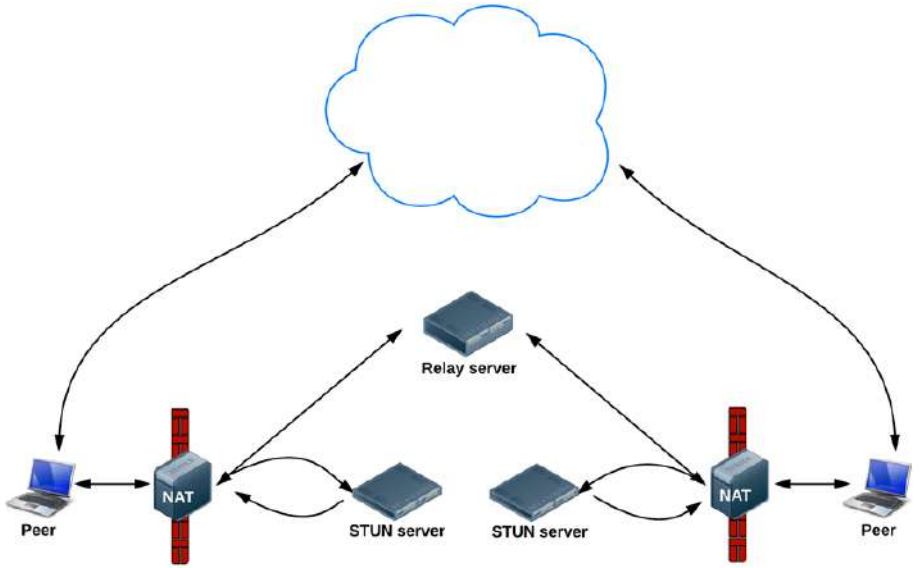
Signaling is used to exchange three types of information :
- Session control messages: to initialize or close communication and report errors
- Network configuration: to the outside world, what's my computer's IP address and port
- Media capabilities: what codecs and resolutions can be handled by my browser
  and the browser it wants to communicate with

Signaling


Signaling
Signaling
App
App
App
SessionDescription
SessionDescription
WebRTC
Browser
Browser
Media
Caller
Callee

Signaling

WebRTC is designed to work peer-to-peer, so users can connect by the most direct route possible.
However, WebRTC is built to cope with real-world networking :
client applications need to traverse NAT gateways and firewalls,
and peer to peer networking needs fallbacks in case direct connection fails.
As part of this process, the WebRTC APIs use STUN servers to get the IP address of your computer,
and TURN servers to function as relay servers in case peer-to-peer communication fails.



STUN Servers

Session Traversal Utilities for NAT (STUN) :
simple protocol for discovering the server-reflexive address.
– Client: Where do you see me at?
– Server: I see you at 206.123.31.67:55123.

A STUN server is only used during the connection setup and once that session has been established, media will flow directly between clients.

TURN Servers

If a STUN server cannot establish the connection,
TURN (Traversal Using Relays around NAT) comes into picture.

TURN Server allocates some of its resources to the client.
The client then uses the TURN server for the media flow.
It acts as a relay between different peers.
A TURN server provides a relayed address to the clients which is then used by the clients for communication.

Unlike STUN, TURN remains in between the peers throughout the session.

ICE : Interactive Connectivity Establishment

- A framework for connecting peers
- A combination of STUN and TURN technology
- Tries to find the best path for each call
- First priority is STUN, if STUN fails, it turns to TURN

JavaScript Frameworks

There are many JavaScript frameworks available which implement WebRTC and provide easy to use APIs to build WebRTC applications

- simpleWebRTC (https://simplewebrtc.com/)
- PeerJS (http://peerjs.com/)
- easyRTC (https://easyrtc.com/)

My Apps

- https://hxiicwrtc2.appspot.com (simpleWebRTC)
- https://hxiicwrtc.appspot.com (PeerJS)

Thank You

- Hitesh Kumar