WebRTC Server Side Cheat Sheet

If you want to build a WebRTC product that works, you will need to first understand what server side components are necessary. Only then will you be able to pick out your technology stack and architecture to fit your needs.

WebRTC is a rather new technology that needs more than just a “regular” web server and your good ol’ browser.

There are 3 additional server types you will need to deal with and deploy to get that product of yours running well. In this brief cheat sheet, I want to share with you what these servers are, their machine specification and what other developers are doing to successfully build their WebRTC products.

- Tsahi Levent-Levi

There are 4 servers needed to run most WebRTC services:

#1 - Web Server

The server “behind” the URL users place in their browser. Built from every-day web development technologies

- Have one already? Great. See if you can merge it with the signaling server
- Need to decide? Start by picking a singling server and then decide on your web server technology

#2 – Signaling Server

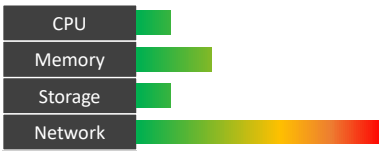
The server handling calls, sessions and rooms. Used to send SDP offer/answer messages between WebRTC devices

- Go for an asynchronous language/framework, preferably based on Node.js
- Check github for existing solutions
- Explore the use of SaaS: Firebase, PubNub, Pusher, …

#3 – STUN/TURN Server

Takes care of making sure media gets connected, even if there are firewalls and NATs along the route

- Don't use public, freely available STUN servers in production
- Deploy COTURN or restund on your own
- Or use a managed service from XirSys, Twilio, Bit6, …



#4 – Media Server

Optional, used when you need to process media in one way or another on the server side

- Most products end up needing media servers
- Their specification and technology stack depends on your exact needs…
- See below for a few examples

SFU – Routing Media

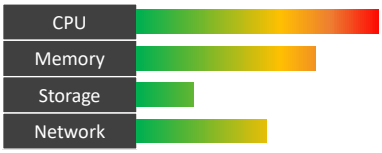
Group video calling where the server routes media between the participants

- The popular choice for multiparty in WebRTC
- Look at Jitsi, Kurento, mediasoup, SwitchRTC and Intel CS for WebRTC

MCU – Mixing Media

Group video calling where the server mixes and combines media inputs into a single stream sent to the participants

- Usually adopted when connecting to external system such as telephony and video streaming
- Look at Kurento, Intel CS for WebRTC and Dialogic PowerMedia XMS



Recording Server

Optional, used when you need to process media in one way or another on the server side

- It depends on how you want to record – the streams received, or post processed for single file playback
- Look at your SFU and MCU solutions to see how recording fits into their architecture

Don't forget CPaaS

Communications Platform as a Service – fully managed (and hopefully with the features on you need)

- Less development effort
- Less ongoing maintenance
- Lower risk