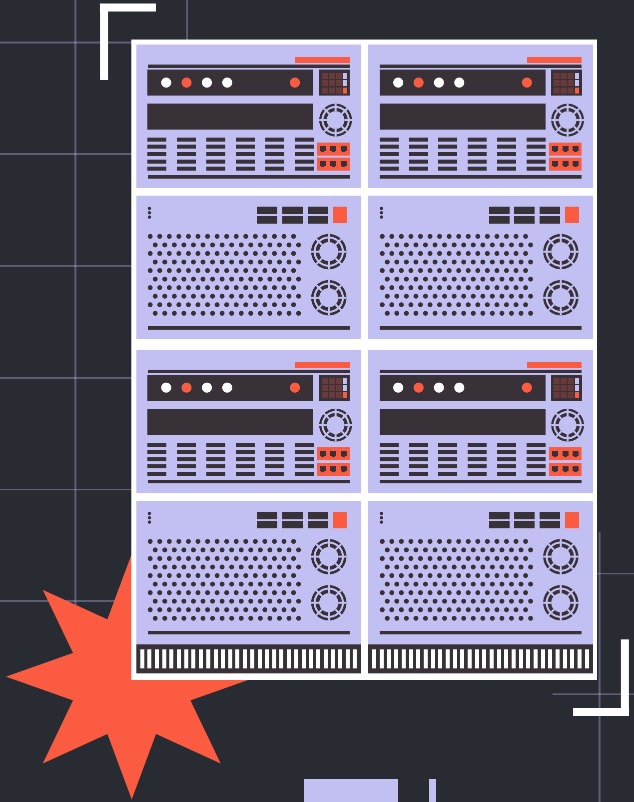
Web Programming course - fall 2024



WebRTC
And real-time communication

Neda Zarei

Instructor: Dr. Saeedi

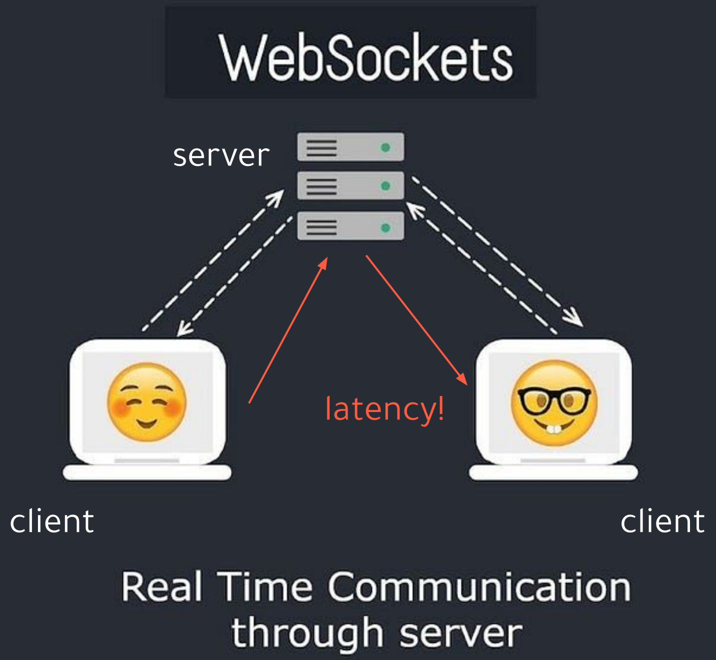
CONTENTS

01

What is WebRTC?

Web Real Time Communication = WebRTC

Web sockets



WebSocket connections always go through the central server, which can introduce unnecessary network hops even if the peers are geographically close.

WebRTC

WebRTC

Lower latency!

client

client

Real Time Communication between browsers. Data never touches server

WebSockets + WebRTC = 👍

~~WebSockets for audio & video~~

WebRTC transports its data over UDP, UDP is FAST

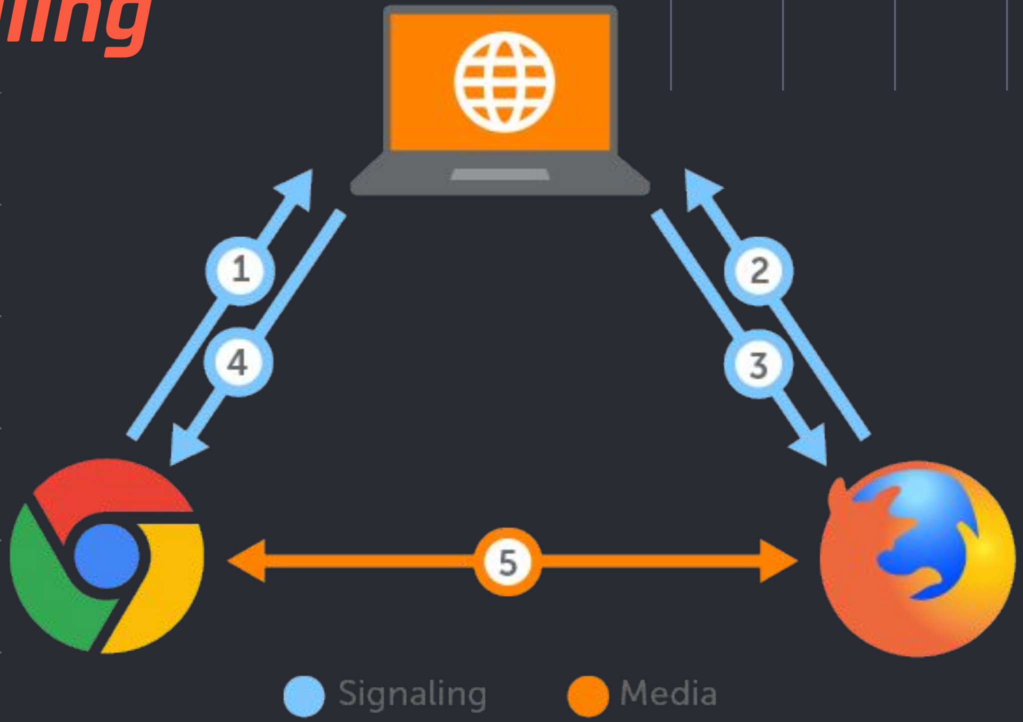
No built in signaling!

But UDP is not reliable for transfering important data!

So no data validation

How do the clients know about each other?

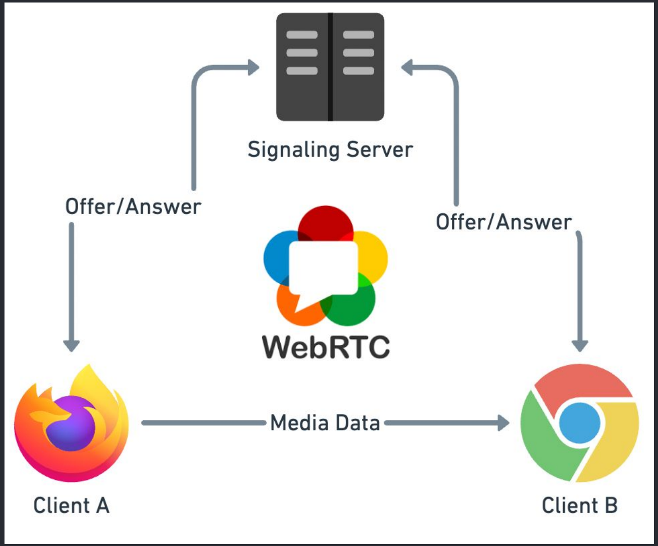
signaling



WHAT?

webRTC is a set of javascript APIs that allow us to establish peer to peer connection between 2 browsers to exchange data such as audio and video in real time.

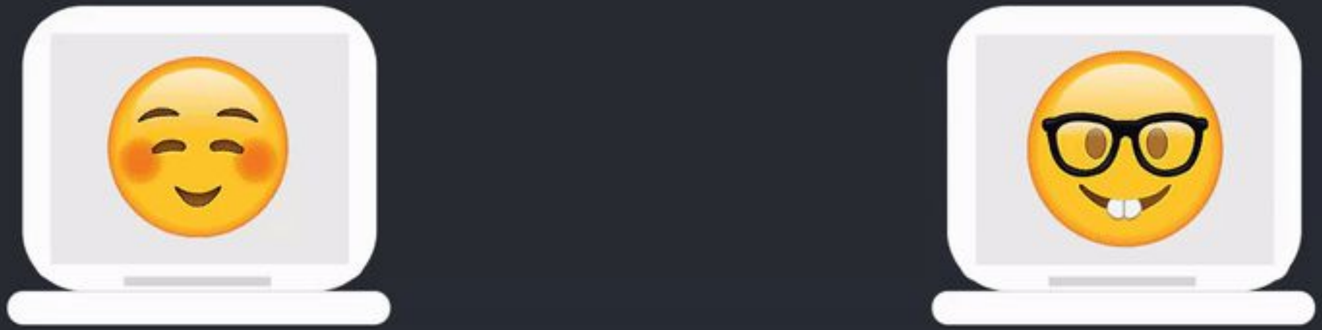
- enable Direct connections between browsers
    - No plugins. Intended to be in standard browser
    - No relays required (but relays possible)
    - Real time = 100ms timescale; "interactive"
    - Media = Audio, Video and "other stuff"



How does it work?

02

WebRTC



What is sent between clients?

SDP's

A Session Description Protocol (SDP), is an object containing information about the session connection such as the codec, address, media type, audio and video and so on.

ICE Candidates

An ICE candidate is a public IP address and port that could potentially be an address that receives data

When a WebRTC application initiates a connection, one of the peers (the offerer) creates an SDP offer using the browser's WebRTC API.

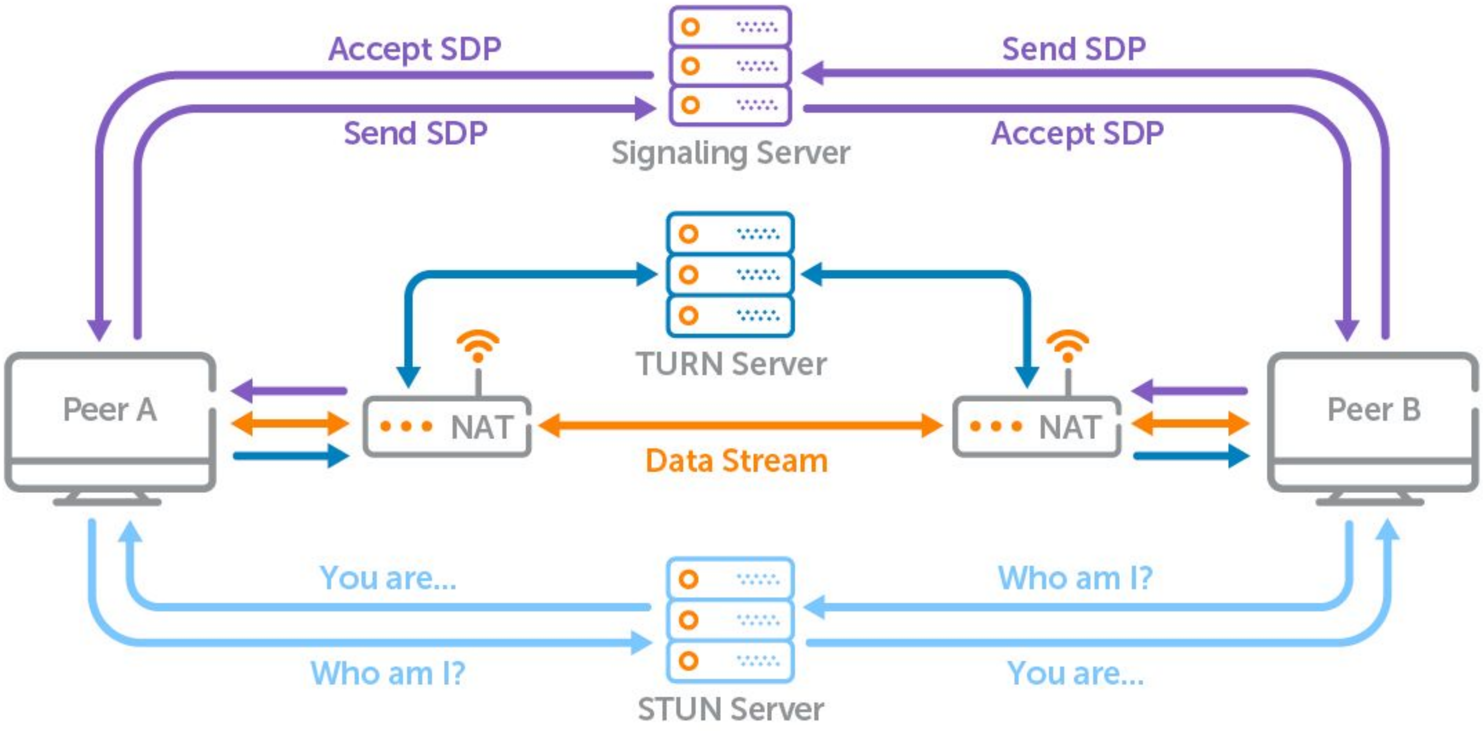
```
v=  (protocol version number)
o=  (originator and session identifier)
s=  (session name)
i=* (session title or short information)
u=* (URI of description)
e=* (zero or more email address with optional name of contacts)
p=* (zero or more phone number with optional name of contacts)
b=* (zero or more bandwidth information lines)
z=* (time zone adjustments)
k=* (encryption key)
a=* (zero or more session attribute lines)
m=  (media name and transport address)
i=* (media title or information field)
c=* (connection information)
b=* (zero or more bandwidth information lines)
k=* (encryption key)
a=* (zero or more media attribute lines)
```

STUN (Session Traversal Utilities for NAT)

- Helps devices discover their public (unique) IP address and port when behind a NAT (Network Address Translation)
- Enables peers to establish direct connections by providing information about their network setup
- Limitation: Fails when both peers are behind restrictive NATs or firewalls

TURN (Traversal Using Relays around NAT)

- if a browser network masks the true ip address for a user, then STUN server can't supply specific enough info to establish a p2p conn
- Acts as a relay server for data when direct peer-to-peer connections cannot be established due to NAT/firewall restrictions
- Ensures reliable communication by relaying traffic through a TURN server
- Limitation: Higher latency and increased server cost compared to direct connections

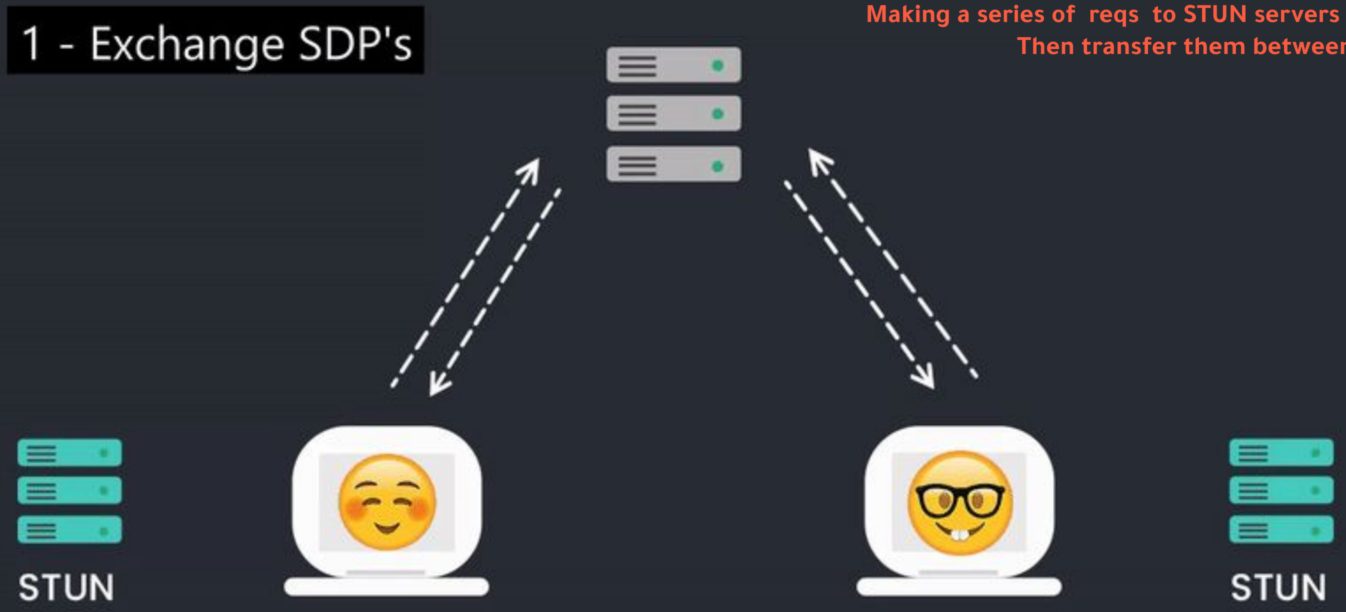

Accept SDP
Send SDP
Send SDP
Accept SDP
Signaling Server
TURN Server
Peer A
NAT
NAT
Peer B
Data Stream
You are...
Who am I?
Who am I?
You are...
STUN Server

Through signaling

Exchanging SDP and ICE candidates

Problem! Most devices sit behind firewalls and NAT devices

Making a series of reqs to STUN servers to get out ICE candidate
Then transfer them between two peers



Cheap
Easy to maintain
e.g. google STUN server

HOW?

- **API**: WebRTC provides APIs for peer-to-peer communication, allowing developers to build real-time video, audio, and data-sharing applications directly in the browser without plugins
- **Identify**: mechanisms like session descriptions and signaling protocols to identify and connect peers in a WebRTC session
- **Type of data**: supports multiple types of data such as video, audio, and arbitrary data (e.g., files, text) using data channels
- **NAT traversal**: uses protocols like ICE (Interactive Connectivity Establishment) to traverse NAT (Network Address Translation) and connect peers behind firewalls
- **Security**: ensures secure communication with mandatory encryption via DTLS (for signaling) and SRTP (for media)
- **Codec**: supports various codecs (e.g., VP8, H.264 for video; Opus for audio) to encode and decode media streams efficiently

Firebase
as
signaling server

```
➜ presentation npm install firebase
```

For creating a vanilla JS project

Vite
Next Generation Frontend Tooling

```
➜ presentation npm init @vitejs/app
Need to install the following packages:
@vitejs/create-app@2.5.2
Ok to proceed? (y) y

npm warn deprecated @vitejs/create-app@
te' instead

> npx
> create-app

@vitejs/create-app is deprecated, use n

✔ Project name: … vite-project
✔ Select a framework: › Vanilla
✔ Select a variant: › JavaScript
```

```
➜ webrtc-presentation npm run dev

> webrtc-presentation@0.0.0 dev
> vite

  VITE v6.0.3  ready in 328 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Firebase   webRTC-presentation ▾   Project settings

Project Overview
Generative AI
Build with Gemini
Project shortcuts
Firestore Database
Product categories
Build
Run
Analytics
All products
Related development tools
IDX
Checks
Spark
No-cost ($0/month)
Upgrade
NEW

present-webrtc
Web App
ss
Web App
webrtc
Web App
Pending delete apps
1 app pending deletion

present-webrtc
App ID
1:720699494638:web:9ea9be9a35d1cc5cb8e3ff
Link to a Firebase Hosting site

This app isn't linked to a Google Analytics stream. To enable Analytics reporting, create a stream for this app.
Create stream

SDK setup and configuration
npm   CDN   Config

Get the snippet for your app's Firebase config object. Learn more.

Firebase configuration object containing keys and identifiers for your app:

```
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBeVhcMr8oUqXh4VzAYkuCKM287pY_YdLs",
  authDomain: "webrtc-presentation-7e0f0.firebaseapp.com",
  projectId: "webrtc-presentation-7e0f0",
  storageBucket: "webrtc-presentation-7e0f0.firebasestorage.app",
  messagingSenderId: "720699494638",
  appId: "1:720699494638:web:9ea9be9a35d1cc5cb8e3ff",
  measurementId: "G-XNE3RZYSY5"
};
```

03 Demo

04

Challenges of webRTC

challenges

WebRTC works via UDP

uses UDP for fast, low-latency communication. However, UDP can be less reliable than TCP, as it doesn't guarantee message delivery or order.

No standard signaling protocol

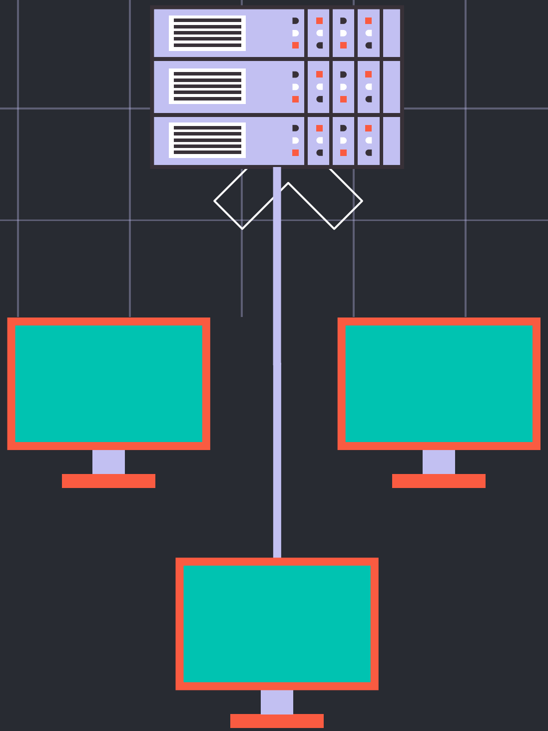
itself doesn’t define a signaling protocol, leaving developers to implement their own solutions (e.g., using WebSockets or REST APIs) for session establishment and peer negotiation.

Not fully compatible with all browser

Although major browsers support WebRTC, implementation differences can cause compatibility issues, requiring additional testing and adjustments.

05

Why webRTC?



WHY?

Removes the need for extra apps

allows real-time communication directly in web browsers without requiring additional applications or plugins, simplifying deployment and user experience

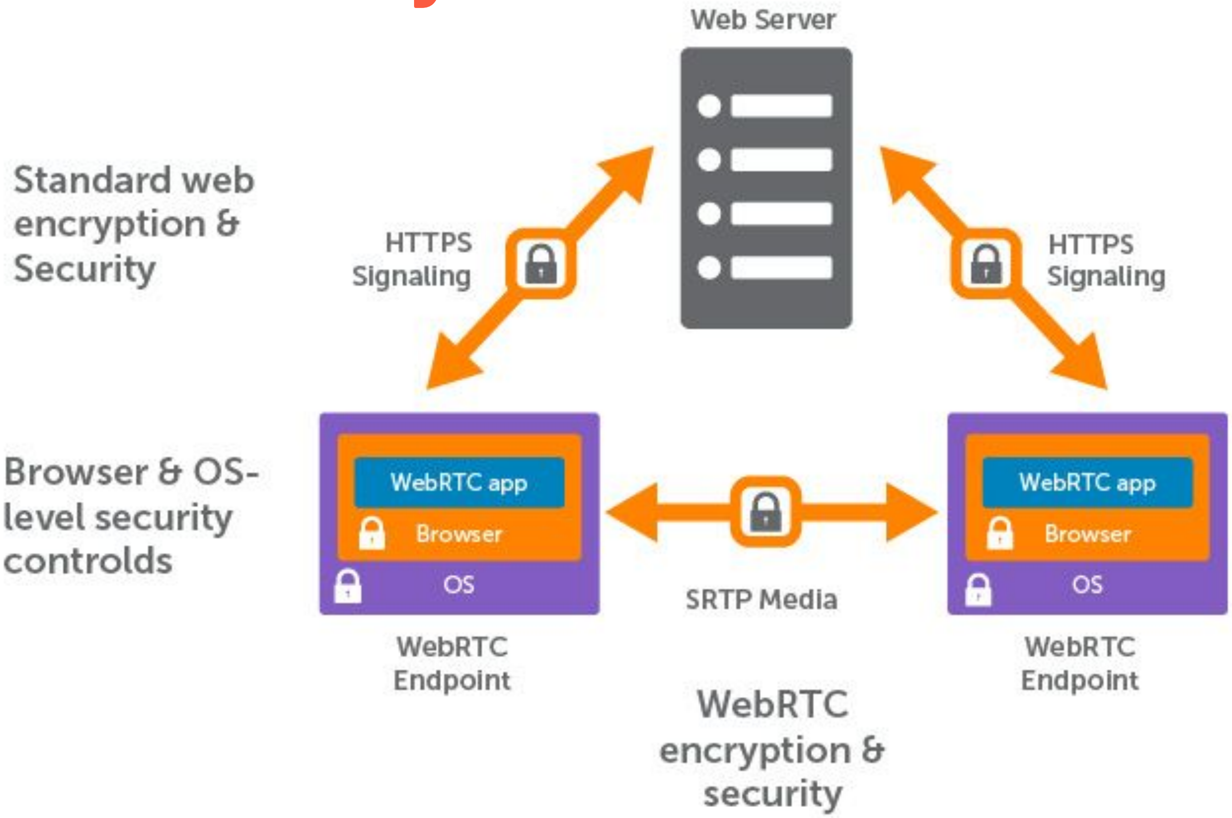
Embedded in web technologies

WebRTC is built into modern browsers and uses familiar web technologies like JavaScript, making it easy for developers to integrate into websites and applications

Secure

ensures secure communication by default, using encryption protocols like DTLS and SRTP to protect data and media streams

webRTC security



06

Applications

1. Video Conferencing Platforms

- Google Meet, Zoom (browser version), Microsoft Teams (browser)
- Enables peer-to-peer video and audio communication directly in browsers without additional plugins

2. Online Gaming Voice Chat

- Discord (browser), in-game chat in multiplayer games
- Real-time voice and sometimes video communication between players

3. Customer Support Chat

- Intercom, Zendesk (voice and video features)
- Allows businesses to integrate real-time communication for customer support

4. Telemedicine

- Doxy.me, Teladoc (browser-based consultations)
- Provides secure video/audio for remote doctor-patient consultations

5. Live Streaming

- Twitch (web screen-sharing), YouTube Live for low-latency streams
- Enables low-latency video and screen-sharing for live broadcasts

6. E-Learning Platforms

- Google Classroom, Khan Academy (live classes)
- Enables interactive video and audio for remote learning

7. Remote Collaboration Tools

- Miro (collaborative whiteboard), Slack (audio/video calls)
- Supports real-time audio, video, and data sharing for team collaboration

8. File Sharing

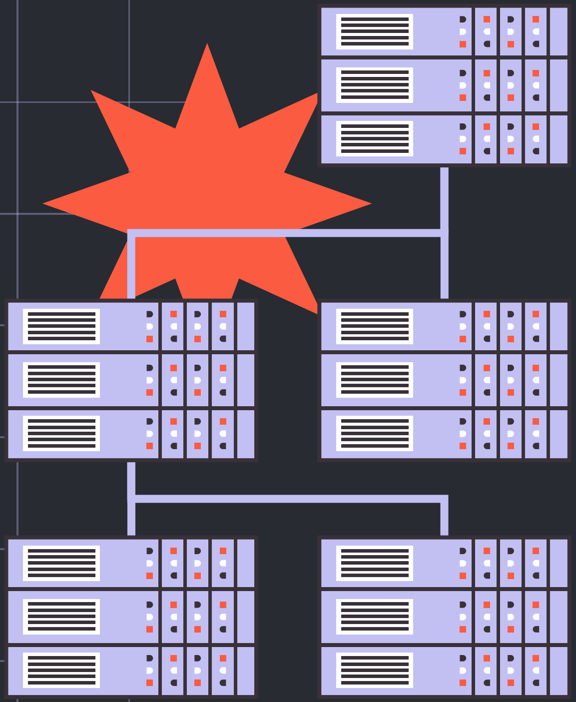
- FilePizza, ShareDrop
- Uses WebRTC's data channels to enable peer-to-peer file sharing without servers

9. IoT and Smart Devices

- Google Nest cameras, Ring doorbells
- Allows real-time video streaming and two-way audio for smart home security systems

10. Virtual Events Platforms

- Hopin, Airmeet.
- Enables real-time video interaction for virtual conferences and events



THANKS!

ANY QUESTIONS?

Credits:

[WebRTC introduction and complete project based tutorial by dennis ivy]

https://github.com/fireship-io/webrtc-firebase-demo?tab=readme-ov-file