

# Audio fingerprinting in WebAssembly per l'esecuzione in browser web

*Candidato:* Davide Pisanò    *Relatore:* Antonio Servetti

Politecnico di Torino

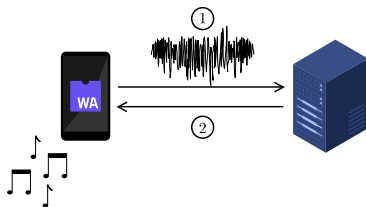
Laurea Magistrale in Ingegneria Informatica  
25 Luglio 2023



**Obiettivo:** fingerprinting di segnali audio all'interno del browser

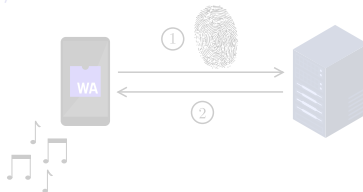
# Rivisitare il Modello C/S

## C/S classico



1. Client invia audio
2. Server risponde

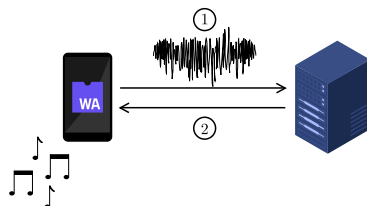
## C/S rivisitato



0. Client calcola fingerprint
1. Client invia fingerprint
2. Server risponde

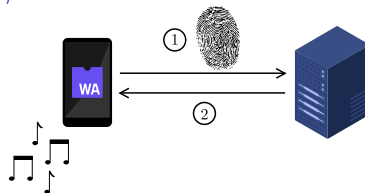
# Rivisitare il Modello C/S

## C/S classico



1. Client invia audio
2. Server risponde

## C/S rivisitato



0. Client calcola fingerprint
1. Client invia fingerprint
2. Server risponde

# Strumenti a Disposizione

Inizialmente capacità dei browser limitate:

- ▶ Javascript ritenuto *anziano*
  - ▶ **Soluzione:** JavaScript ES6



- ▶ Scarse performance con Javascript
  - ▶ **Soluzione:** WebAssembly



- ▶ Niente API *multimediali*
  - ▶ **Soluzione:** WebAudio



- ▶ Niente API per rendering 3D
  - ▶ **Soluzione:** WebGL



# Strumenti a Disposizione

Inizialmente capacità dei browser limitate:

- ▶ Javascript ritenuto *anziano*
  - ▶ **Soluzione:** JavaScript ES6
- ▶ Scarse performance con Javascript
  - ▶ **Soluzione:** WebAssembly
- ▶ Niente API *multimediali*
  - ▶ **Soluzione:** WebAudio
- ▶ Niente API per rendering 3D
  - ▶ **Soluzione:** WebGL



# Strumenti a Disposizione

Inizialmente capacità dei browser limitate:

- ▶ Javascript ritenuto *anziano*
  - ▶ **Soluzione:** JavaScript ES6
- ▶ Scarse performance con Javascript
  - ▶ **Soluzione:** WebAssembly
- ▶ Niente API *multimediali*
  - ▶ **Soluzione:** WebAudio
- ▶ Niente API per rendering 3D
  - ▶ **Soluzione:** WebGL



# Strumenti a Disposizione

Inizialmente capacità dei browser limitate:

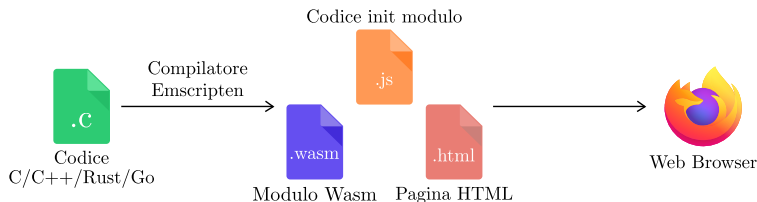
- ▶ Javascript ritenuto *anziano*
  - ▶ **Soluzione:** JavaScript ES6
- ▶ Scarse performance con Javascript
  - ▶ **Soluzione:** WebAssembly
- ▶ Niente API *multimediali*
  - ▶ **Soluzione:** WebAudio
- ▶ Niente API per rendering 3D
  - ▶ **Soluzione:** WebGL



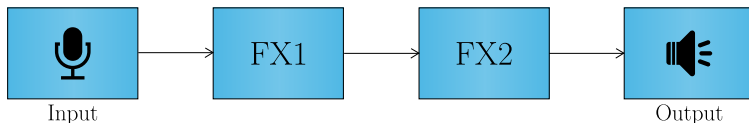


# WebAssembly

- ▶ Formato per codice binario
- ▶ Eseguitibile nel browser
  - ▶ Performance comparabili a quelle native
- ▶ Target per linguaggi di basso livello

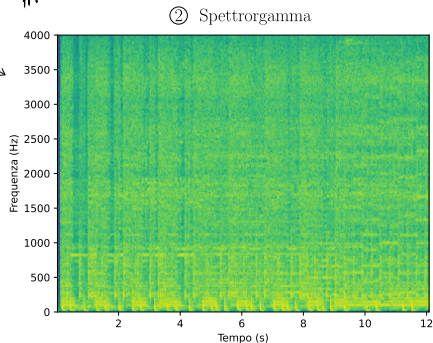
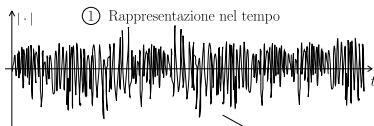


- ▶ API JavaScript
- ▶ Manipolazione e acquisizione audio
- ▶ Processing in tempo reale
- ▶ Architettura basata su nodi



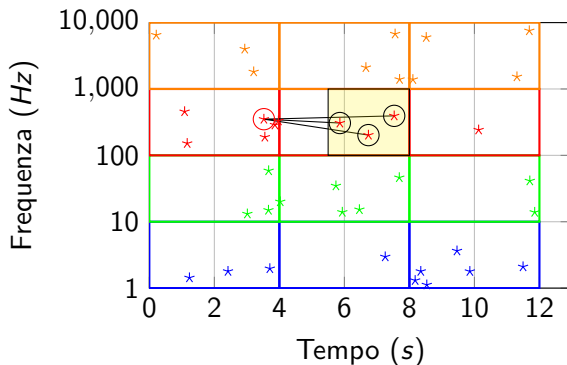
# Il Fingerprinting (1)

- ▶ Il **fingerprint audio** è l'impronta digitale di un audio
  - ▶ Ogni fingerprint identifica univocamente un audio
  - ▶ Approccio simile a Shazam



## Il Fingerprinting (2)

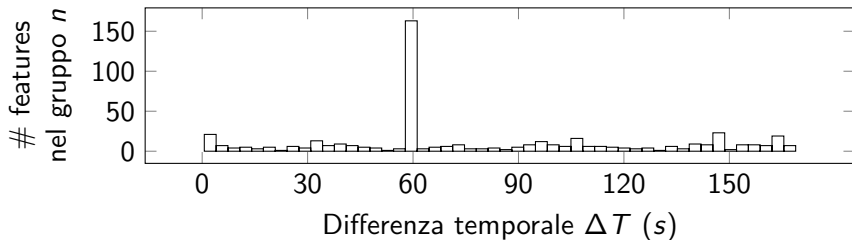
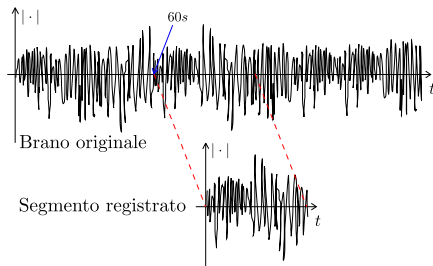
- ▶ Dallo spettrogramma vengono estratti dei picchi
  - ▶ Ovvero delle feature *riassuntive* del segnale
- ▶ Picchi correlati con un picco *anchor*
- ▶ Per ogni coppia *anchor-picco* calcolare l'hash



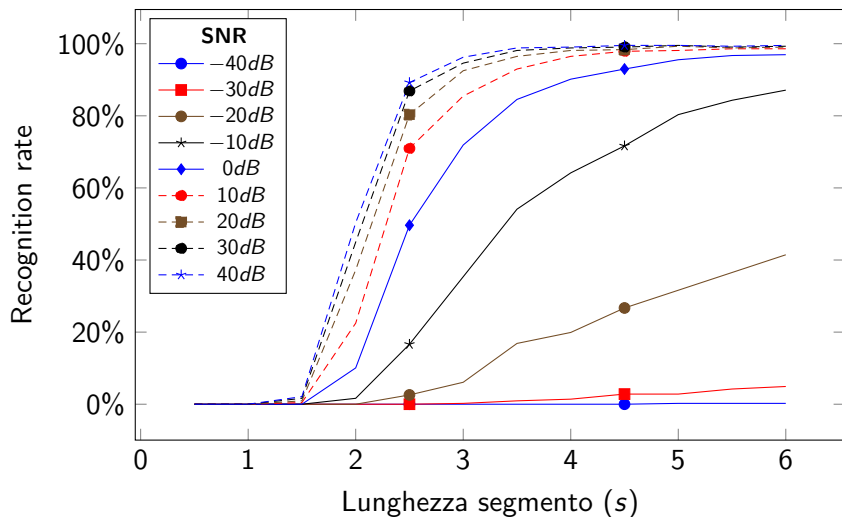
# Matching

- ▶ Basato su tre idee:
  1.  $\Delta T \geq 0$  tra features brano originale e registrazione
  2.  $\Delta T$  costante tra features brano originale e registrazione
  3. Uguaglianza tra features basata su hash
- ▶ Quindi bisognerà:
  1. Raggruppare per  $\Delta T$  e per id brano originale
  2. Ogni gruppo da  $n$  features
  3. Ordinare per  $n$  crescente
  4. **Miglior match:** gruppo con  $n$  maggiore

# Matching: Visualizzazione



# Recognition Rate



# Applicazioni del Fingerprinting

## ▶ Riconoscimento audio

- ▶ Audio simili condivideranno molte features
- ▶ Es: *Shazam*, *ACRCloud*

## ▶ Sincronizzazione di fonti multimediali

- ▶ Più stream provenienti da fonti differenti
- ▶ Es: second screen application

## ▶ Riconoscimento di eventi

- ▶ Reagire alla presenza di un *marker* audio
- ▶ Es: inserimento di un segmento audio personalizzato, riconoscimento di trigger in videogiochi



# Applicazioni del Fingerprinting

## ▶ **Riconoscimento audio**

- ▶ Audio simili condivideranno molte features
- ▶ Es: *Shazam*, *ACRCloud*

## ▶ **Sincronizzazione** di fonti multimediali

- ▶ Più stream provenienti da fonti differenti
- ▶ Es: second screen application

## ▶ **Riconoscimento** di eventi

- ▶ Reagire alla presenza di un *marker* audio
- ▶ Es: inserimento di un segmento audio personalizzato, riconoscimento di trigger in videogiochi

## ► **Riconoscimento audio**

- Audio simili condivideranno molte features
- Es: *Shazam*, *ACRCloud*

## ► **Sincronizzazione** di fonti multimediali

- Più stream provenienti da fonti differenti
- Es: second screen application

## ► **Riconoscimento** di eventi

- Reagire alla presenza di un *marker* audio
- Es: inserimento di un segmento audio personalizzato, riconoscimento di trigger in videogiochi

# Second Screen Application

