

Audio fingerprinting in WebAssembly per l'esecuzione in browser web

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

Politecnico di Torino

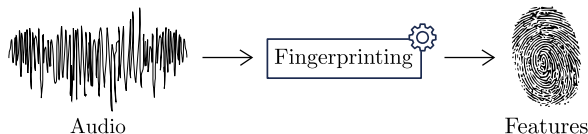
Laurea Magistrale in Ingegneria Informatica
25 Luglio 2023



Obiettivo: fingerprinting di segnali
audio nel browser

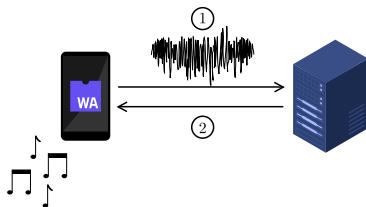
II Fingerprinting

- ▶ **Fingerprint audio** → impronta digitale di un audio
 - ▶ Ogni fingerprint identifica univocamente un audio
 - ▶ Insieme di features che *riassumono* l'audio
 - ▶ A segnali simili corrispondono features simili



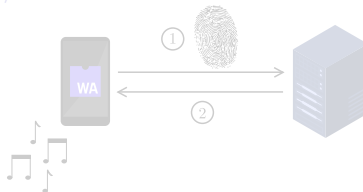
Rivisitare il Modello C/S

C/S classico



1. Client invia audio
2. Server risponde

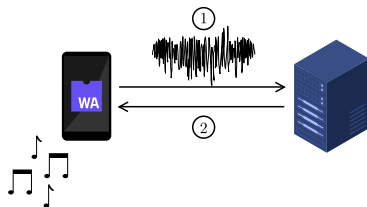
C/S rivisitato



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

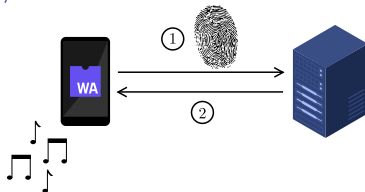
Rivisitare il Modello C/S

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



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



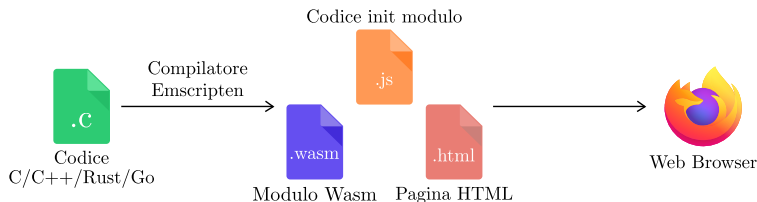
Inizialmente capacità dei browser limitate:

- ▶ Javascript ritenuto *anziano*
 - ▶ **Soluzione:** JavaScript ES6
- ▶ Scarse performance con Javascript
 - ▶ **Soluzione:** WebAssembly
- ▶ Niente API *multimediali*
 - ▶ **Soluzione:** WebAudio

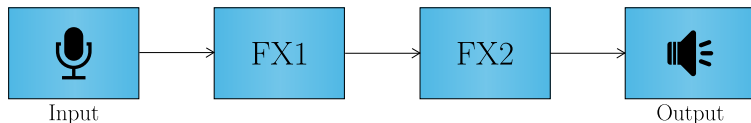


WebAssembly

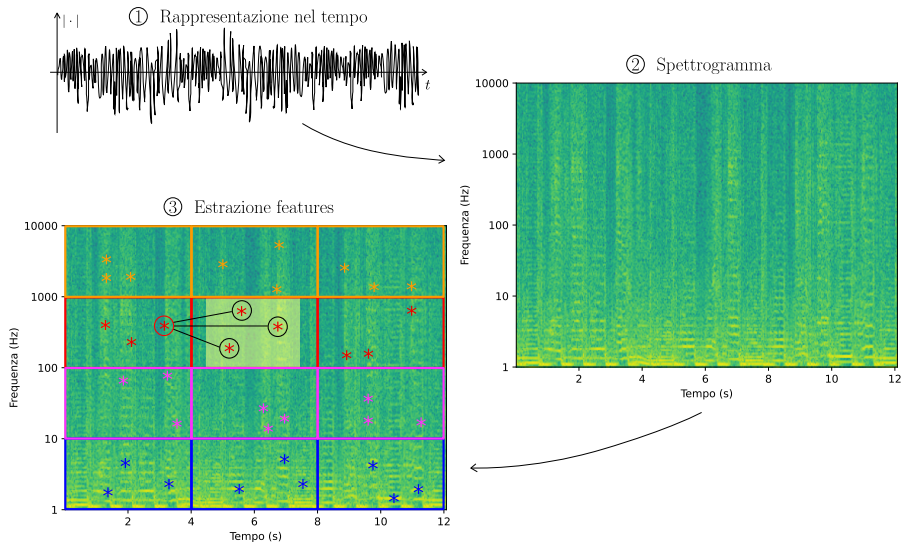
- ▶ Formato per codice binario
- ▶ Eseguitibile nel browser
 - ▶ Performance comparabili a quelle native
- ▶ Target per linguaggi di basso livello



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



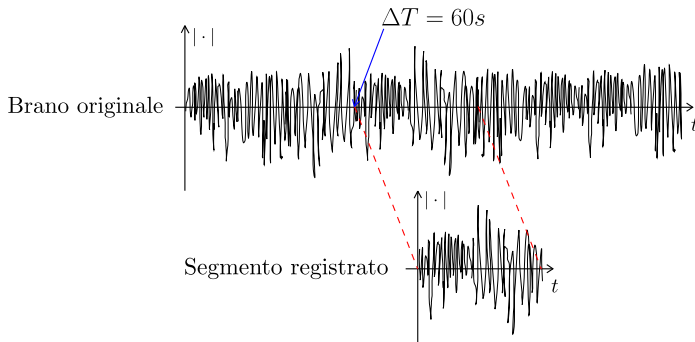
Il Fingerprinting nel Dettaglio



Matching (1)

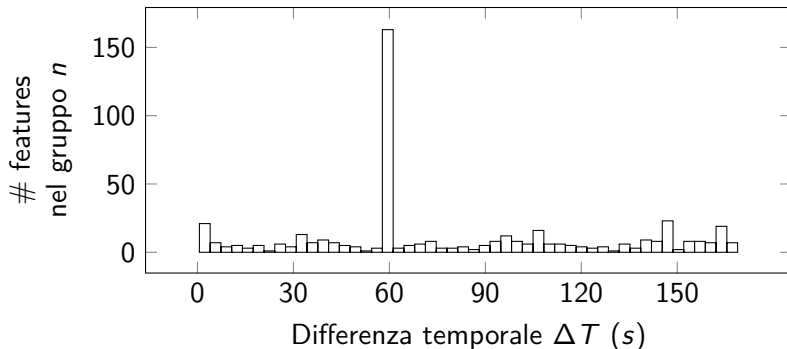
► Basato su due idee:

1. $\Delta T \geq 0$ tra features brano originale e registrazione
2. ΔT costante tra features brano originale e registrazione

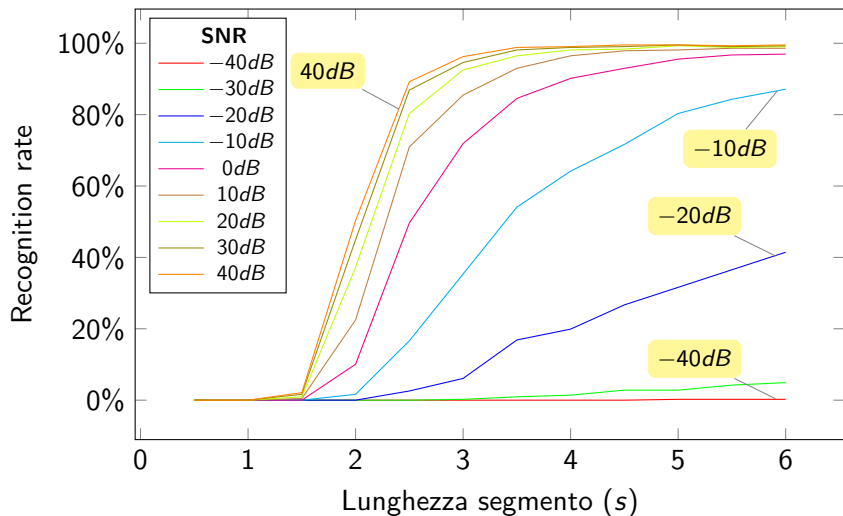


Matching (2)

- ▶ Il *matching* consiste quindi nel:
 1. Raggruppare per ΔT e per id brano originale
 2. Ogni gruppo da n features
 3. **Miglior match**: gruppo con n maggiore



Recognition Rate



▶ **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: 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: 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: riconoscimento di trigger in videogiochi

Conclusioni

- ▶ È possibile fare fingerprinting di segnali audio nel browser?
- ▶ **Si.**

