Audio fingerprinting in WebAssembly per l'esecuzione in browser web

Candidato: Davide Pisanò *Relatore*: Antonio Servetti

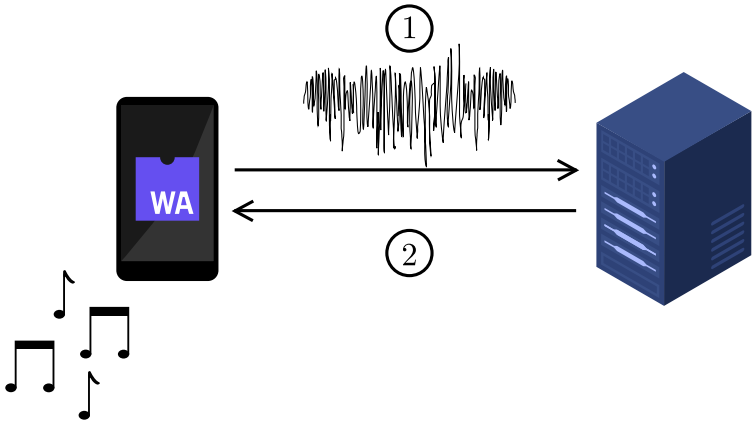
Politecnico di Torino

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



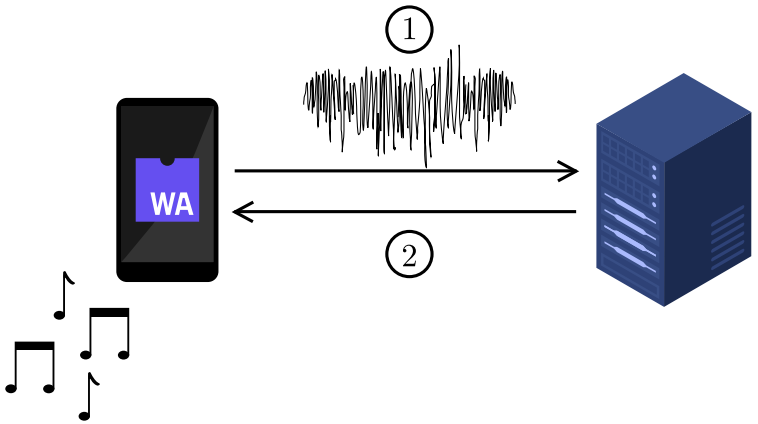
1. Client invia audio
2. Server risponde

C/S rivisitato

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

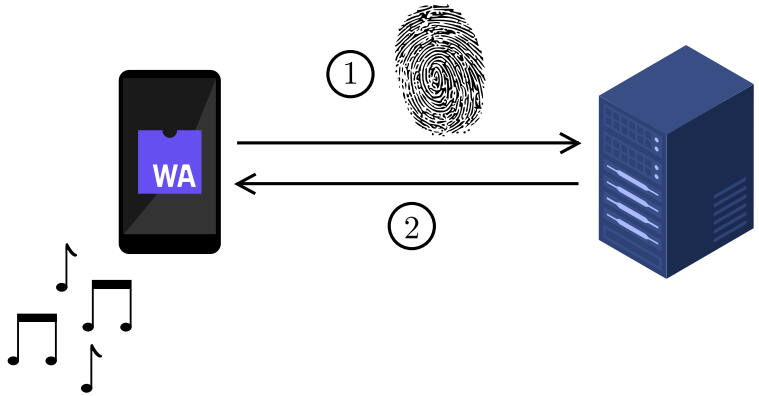
Rivisitare il Modello C/S

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

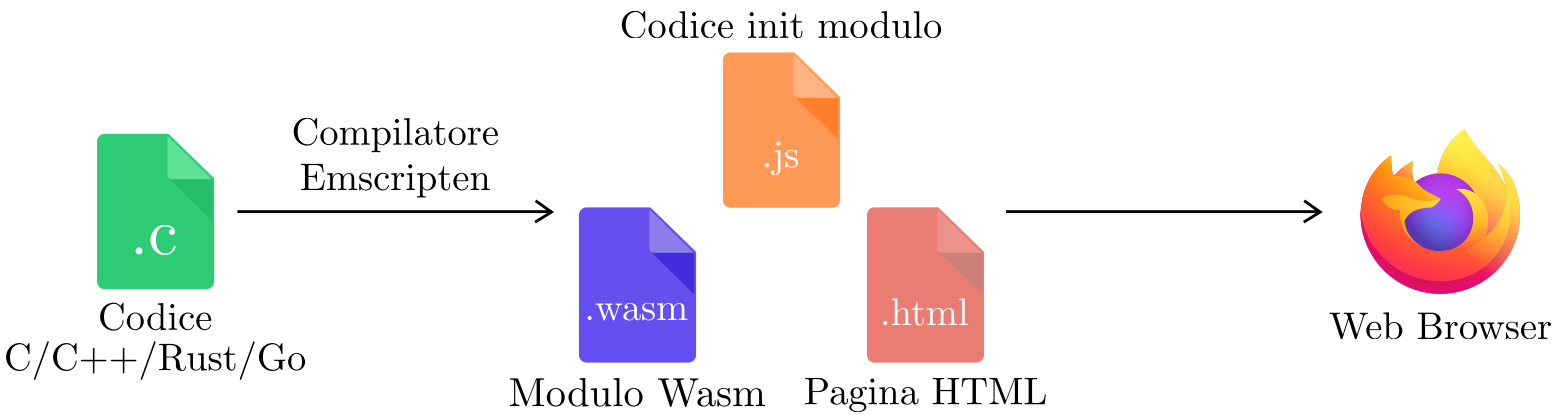
Strumenti a Disposizione

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

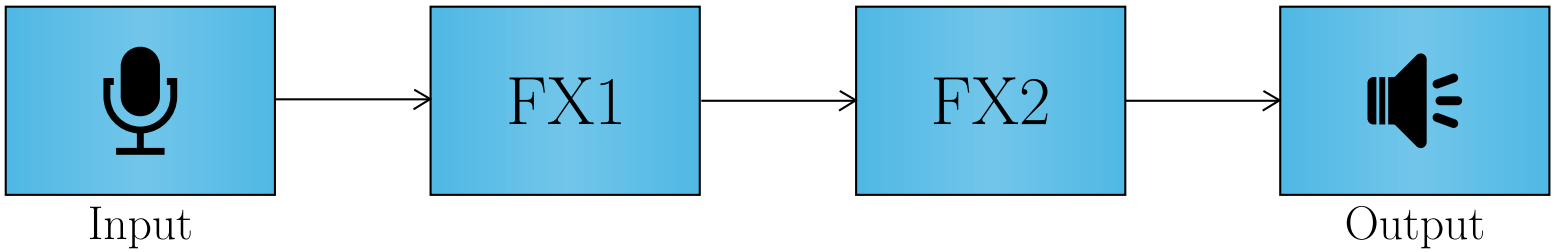
WebAssembly

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



WebAudio

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

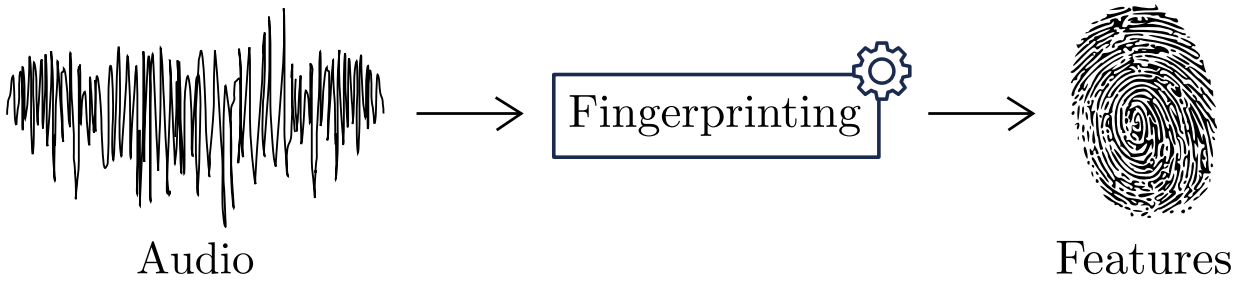


Il Fingerprinting Audio

- Il **fingerprint audio** è l'impronta digitale di un audio
  - Ogni fingerprint identifica univocamente un audio
- Il **fingerprinting audio** crea il fingerprint dell'audio
  - L'algoritmo estrae delle features *riassuntive* dell'audio
  - Un audio può essere ricercato note *alcune* sue features

Il Fingerprinting Audio

- Il **fingerprint audio** è l'impronta digitale di un audio
  - Ogni fingerprint identifica univocamente un audio
- Il **fingerprinting audio** crea il fingerprint dell'audio
  - L'algoritmo estrae delle features *riassuntive* dell'audio
  - Un audio può essere ricercato note *alcune* sue features



Applicazioni del Fingerprinting

- **Riconoscimento audio**
  - Audio simili convideranno 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

Applicazioni del Fingerprinting

- **Riconoscimento audio**
  - Audio simili convideranno 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

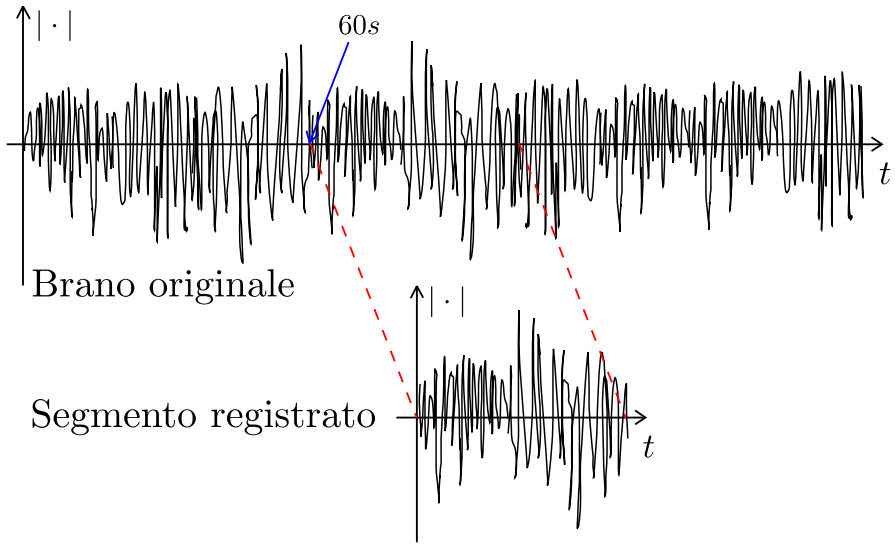
Applicazioni del Fingerprinting

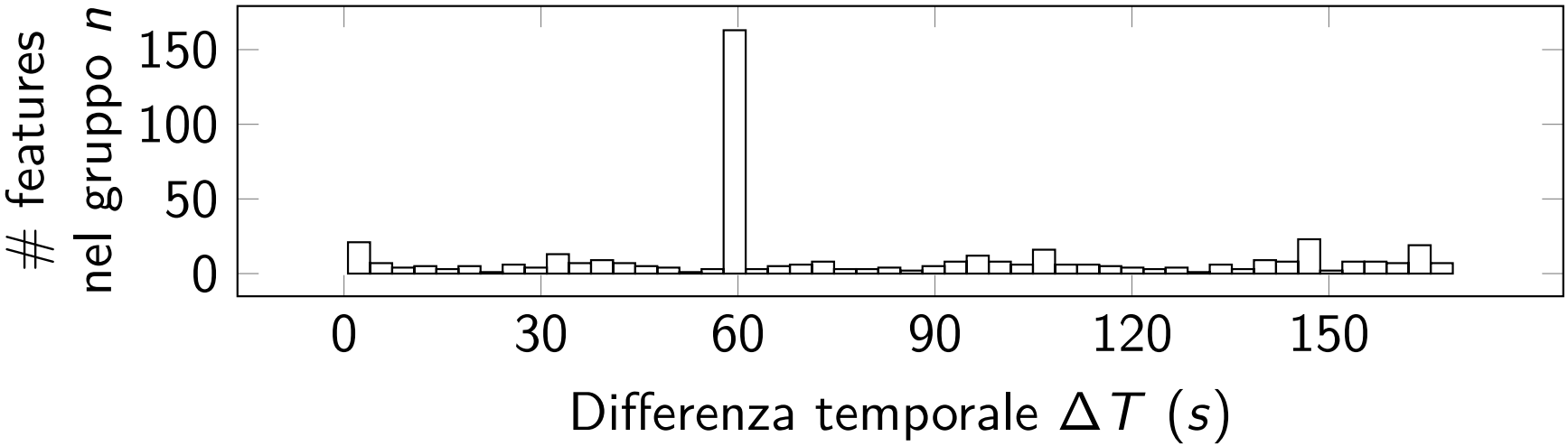
- **Riconoscimento audio**
  - Audio simili convideranno molte features
  - Es: *Shazam*, *ACRCloud*
- **Sincronizzazione** di fonti multimediali
  - Più stream provenienti da fonti differenti
  - Es: second screen application
- **Riconoscimento** di eventi
  - Reagire alla presenza di un *marker* audio
  - Es: inserimento di un segmento audio personalizzato

Matching

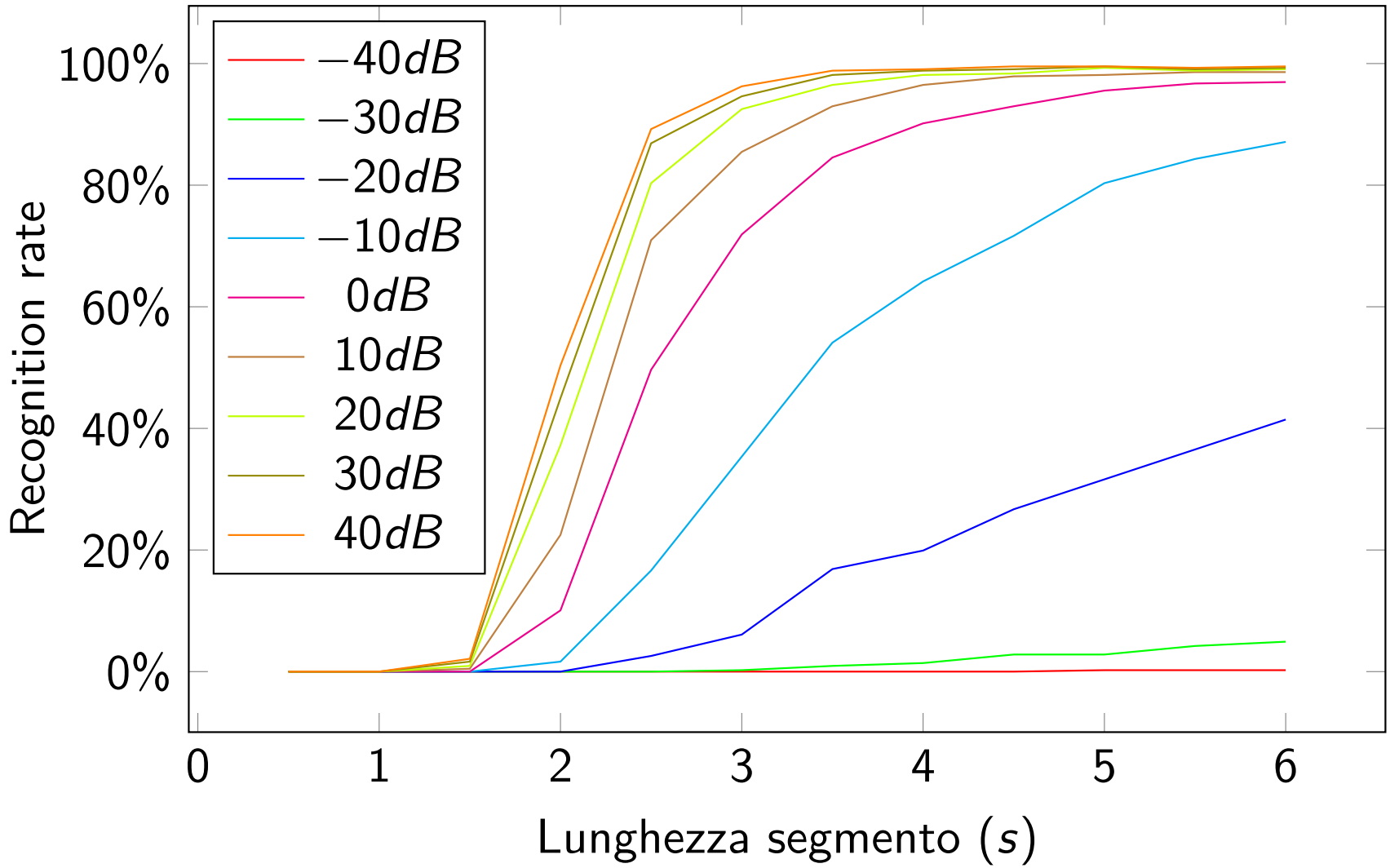
- Basato su tre idee:
  1. $\Delta T \geq 0$ tra features brano originale e registrazione
  2. ΔT costante tra features brano originale e registrazione
  3. Uguaglianza tra features basata su hash
- Quindi bisognerà:
  1. Raggruppare per ΔT e per id brano originale
  2. Ogni gruppo da n features
  3.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

Riconoscimento: Visualizzazione





Recognition Rate



Second Screen Application