# Over-the-air Audio Identification

**Arda Yalçıner**

**FOSDEM'16**, Brussels

*Open Media Devroom*

# Speaker

*Yes, a pizza lover!*

Software Architect @
**Oto.net** / *Istanbul*

*B.Sc.* Astronautical Eng.
*M.Sc.* Software Eng.

arda.yalciner@gmail.com

wizardctp

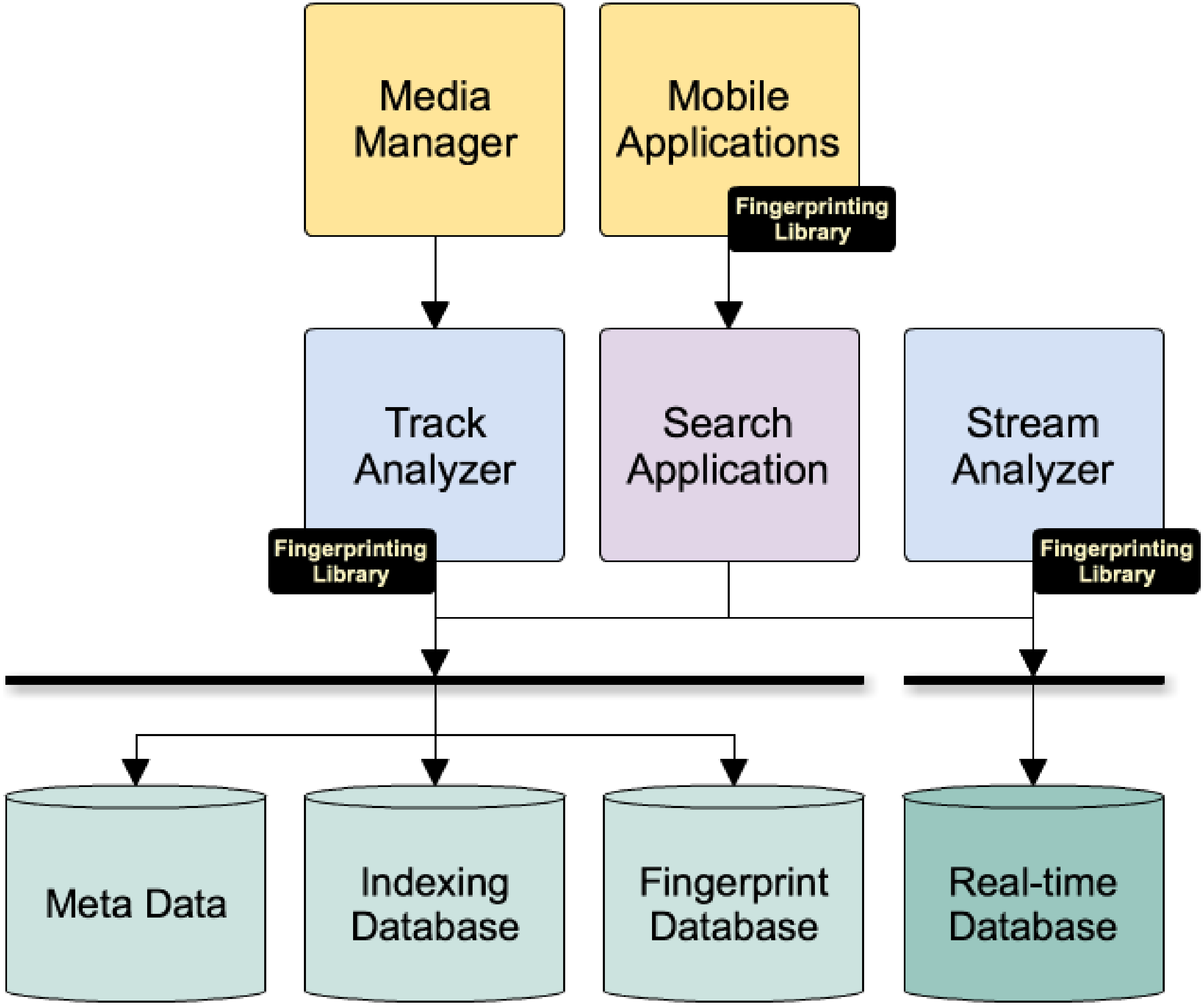
ardayalciner

# OTA Audio Identification

***Matching an audio sample***
***with a pre-recorded sound clip***

- **Music track** recognition
- **Radio / TV station** detection
- **Licensing**
- **Second screen** applications
  - Previously on <*insert TV Show here*>
  - Track watched movies / TV shows
  - Nearby concerts of playing artist
  - Information on a currently speaking movie / TV show character

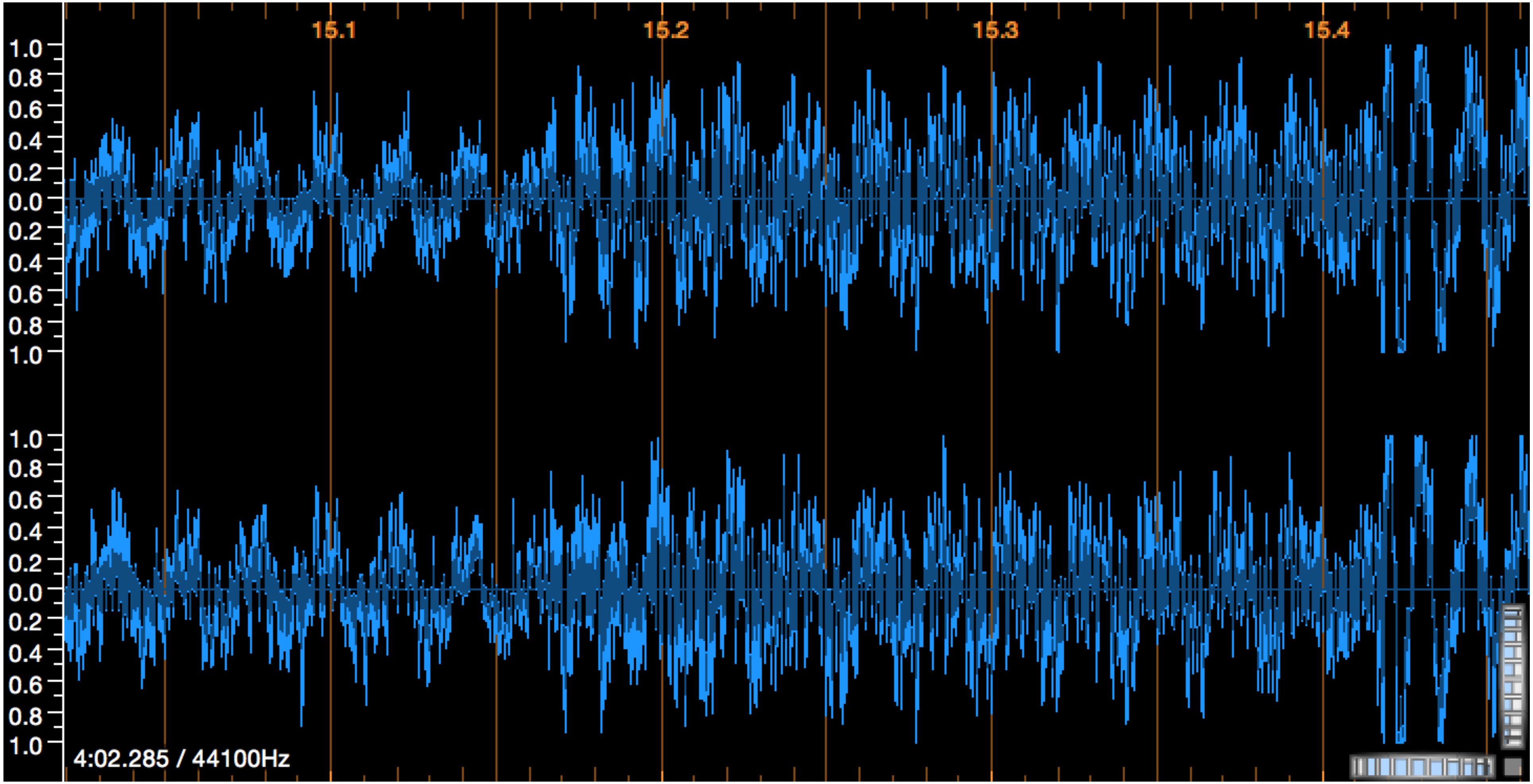
# Reference Architecture



# Digital Sound Signals

- In nature, sound propagates as **sound waves.**
- We measure **sound pressure** at specific intervals. This interval is called **sample rate.**
- A sample rate of 44.1 kHz means, we measured the sound pressure 44100 times per second.
- These discrete signals represent sound in a digital form.

# Digital Sound Signals



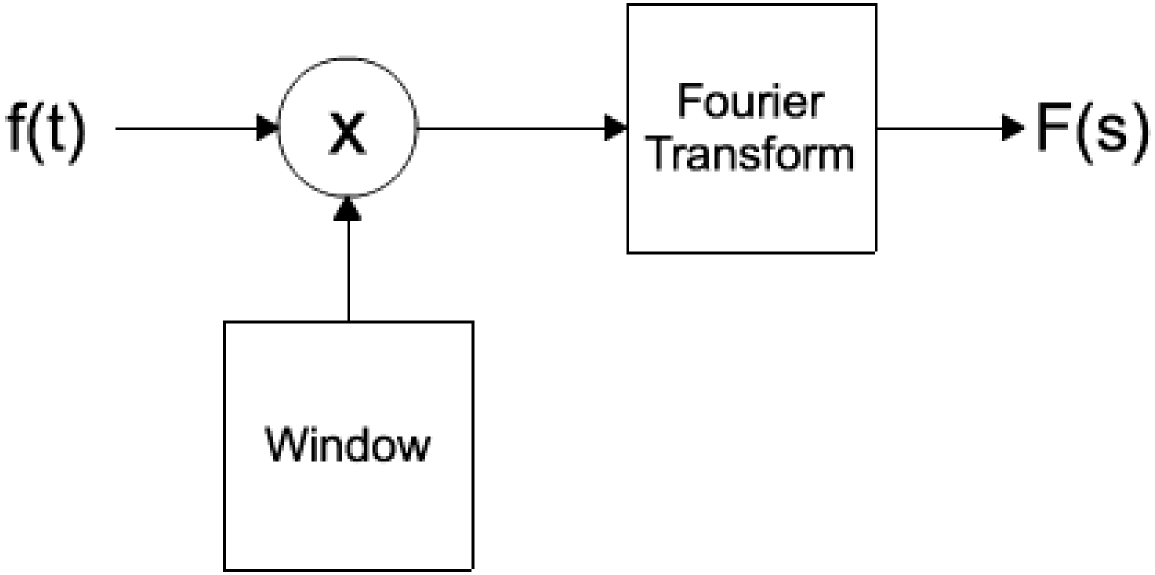
# Digital Sound Signals

- **Properties:**
  - **Bit depth:** # of bits a sample occupies
  - **Channels:** # of simultaneous recordings (*1: mono, 2: stereo, etc.*)
  - <u>**Endianness:** Big-endian vs. Little-endian</u>

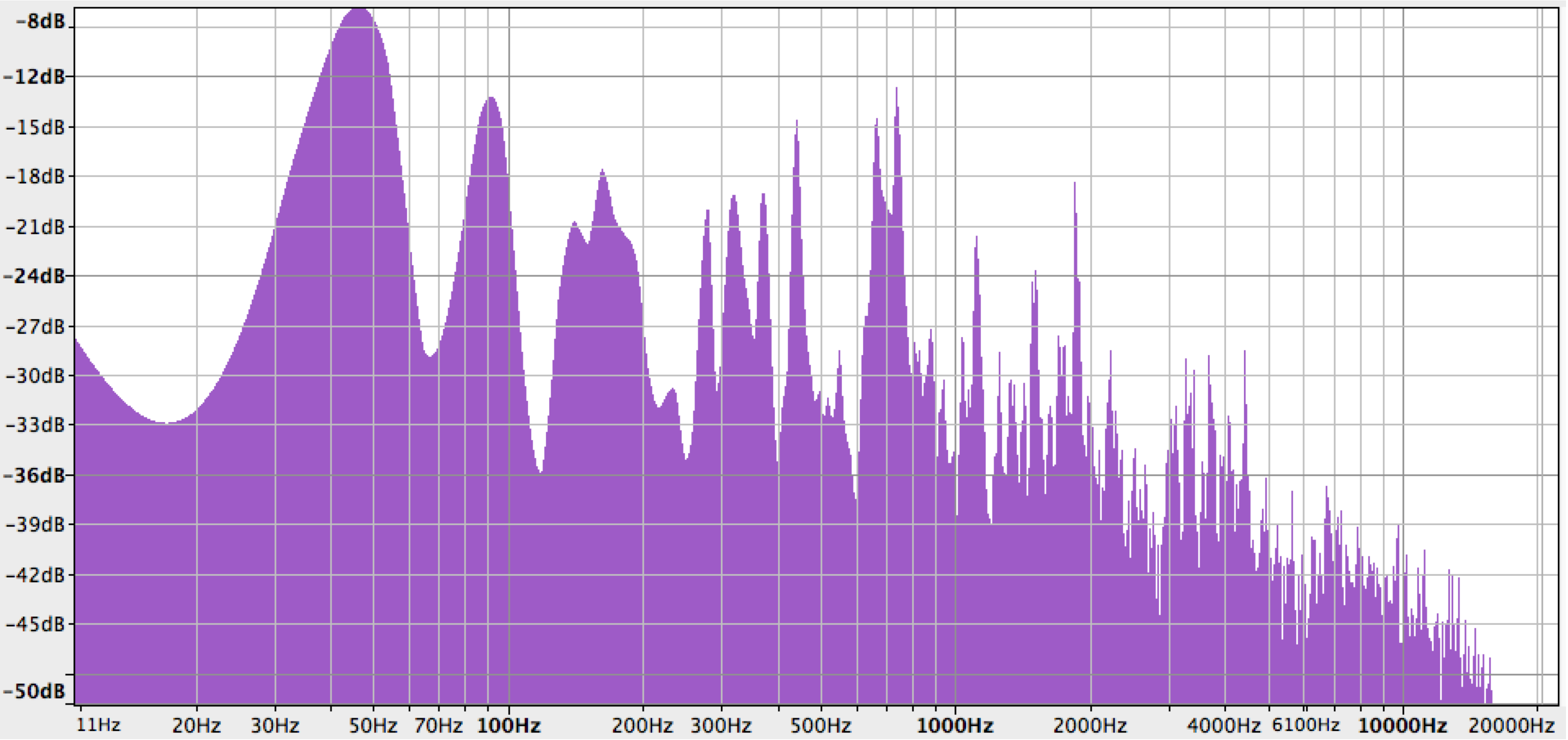
- **File Formats:**
  - Uncompressed: **PCM, Wave**
  - Compressed:
    - Lossless: **FLAC**
    - Lossy: **MP3**, **AAC**, **Ogg**

# Frequency Analysis

- Record or play audio signals in the *time domain*: **SPL vs. Time**
- Analyze audio signals in the *frequency domain*: **Frequency vs. Amplitude vs. Time**

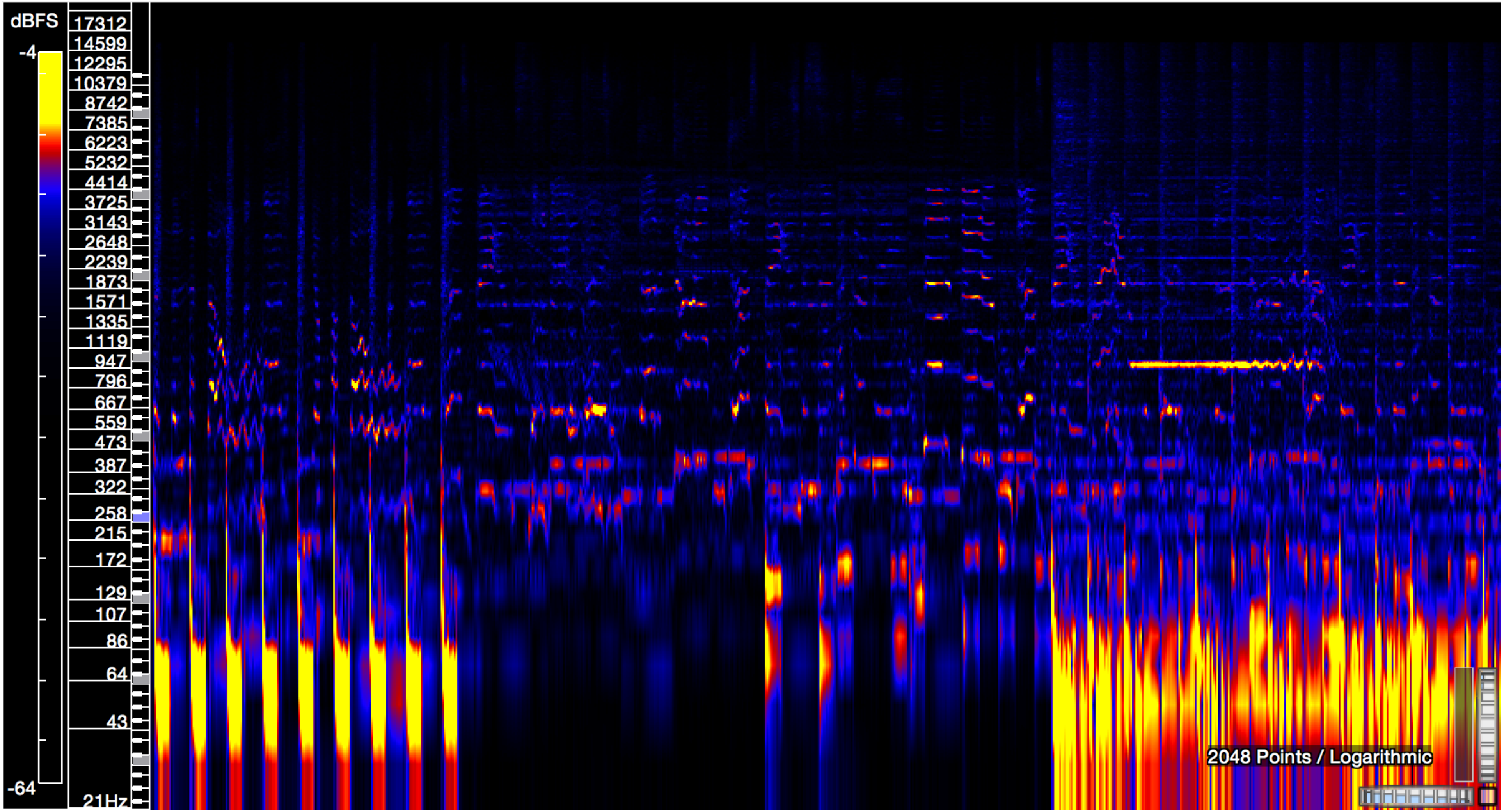


# Frequency Analysis: Spectrum




- Covers frequencies up to `0.5 * sample_rate [Hz]`
- Divided into bins. Each bin represents the average amplitude for `0.5 * sample_rate / fft_points` wide of frequencies

# Frequency Analysis: Spectrogram




- Sensitive either in time dimension or frequency dimension: not both

# Fingerprinting

**Problem:**

We need to **uniquely** summarize a **part of** an audio recording despite various **challenges**

**Approach Using:**

- Music information retrieval (*MIR*)
- Acoustic fingerprinting

# Fingerprinting: MIR

## ***“What can we retrieve?”***

More **specific**:

- Musical features (*notes, chords, harmony, rhythm, …*)
- Speech
- Instruments
- Melody: *Query by Humming*

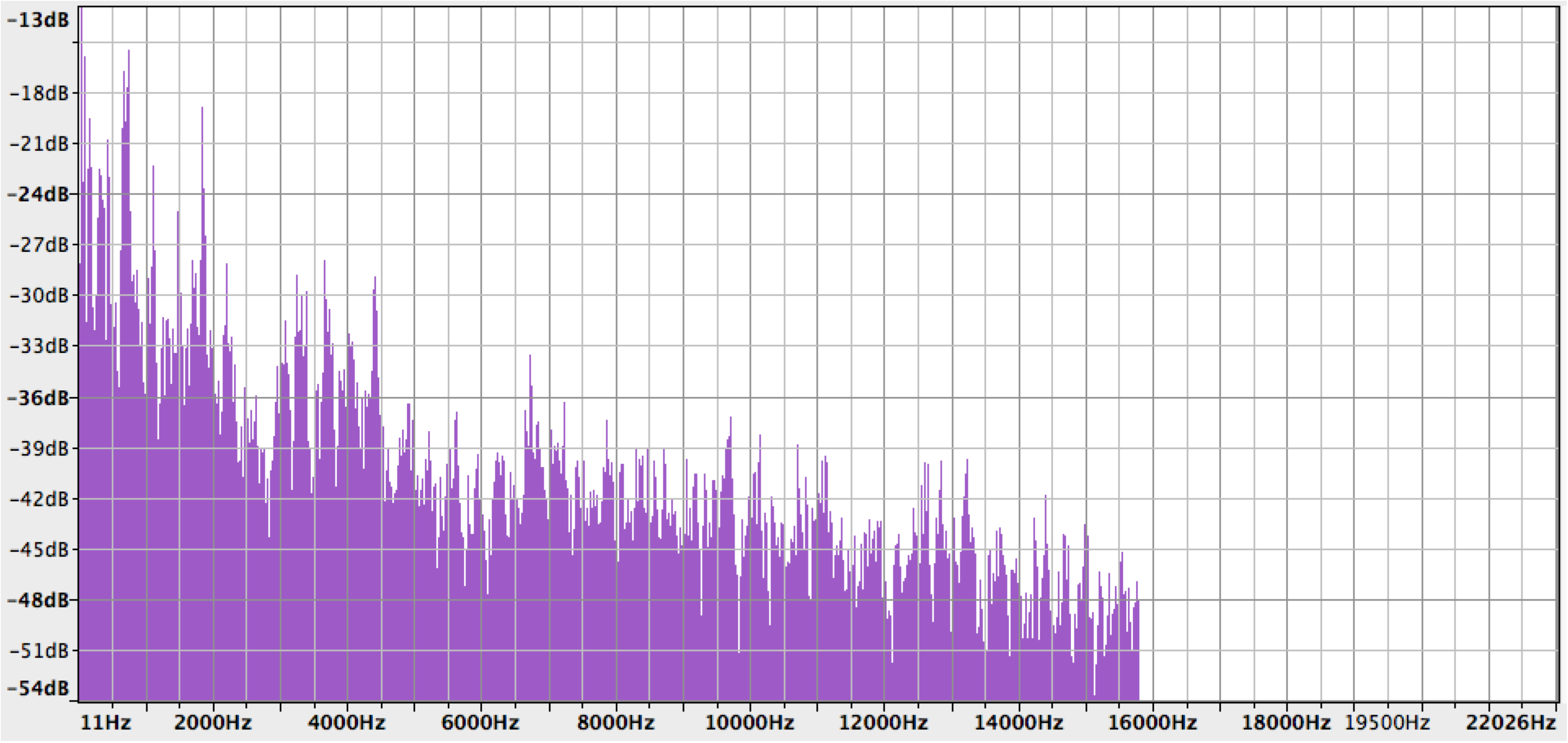
More **abstract**:

- Time-frequency peaks

# Fingerprinting: Challenges

- Noise
  - **Duration**: *instantaneous / continuous*
  - **Frequency range**: *small / wide*
  - **Loudness**: *quiet / loud*
- Echo
- Changes in tempo
- Changes in pitch
- Attenuation or boost in certain frequencies (*e.g., Equalization*)

# Fingerprinting: Time-Frequency Peaks




- **Divide** the spectrum into **N** equal areas (e.g., 16 parts)
- For each area, find the **frequency bin** that provides the **peak amplitude**

# Fingerprinting: Packing

| | |
|---|---|
| **FFT Points** | P = **1024** |
| **# of Areas** | N = **16** |
| **# of Bins / Area** | 0.5 * P / N = **32** |
| **Sample Rate** | SR = **11025** |
| **Max. Frequency** | SR / 2 = **5513** |

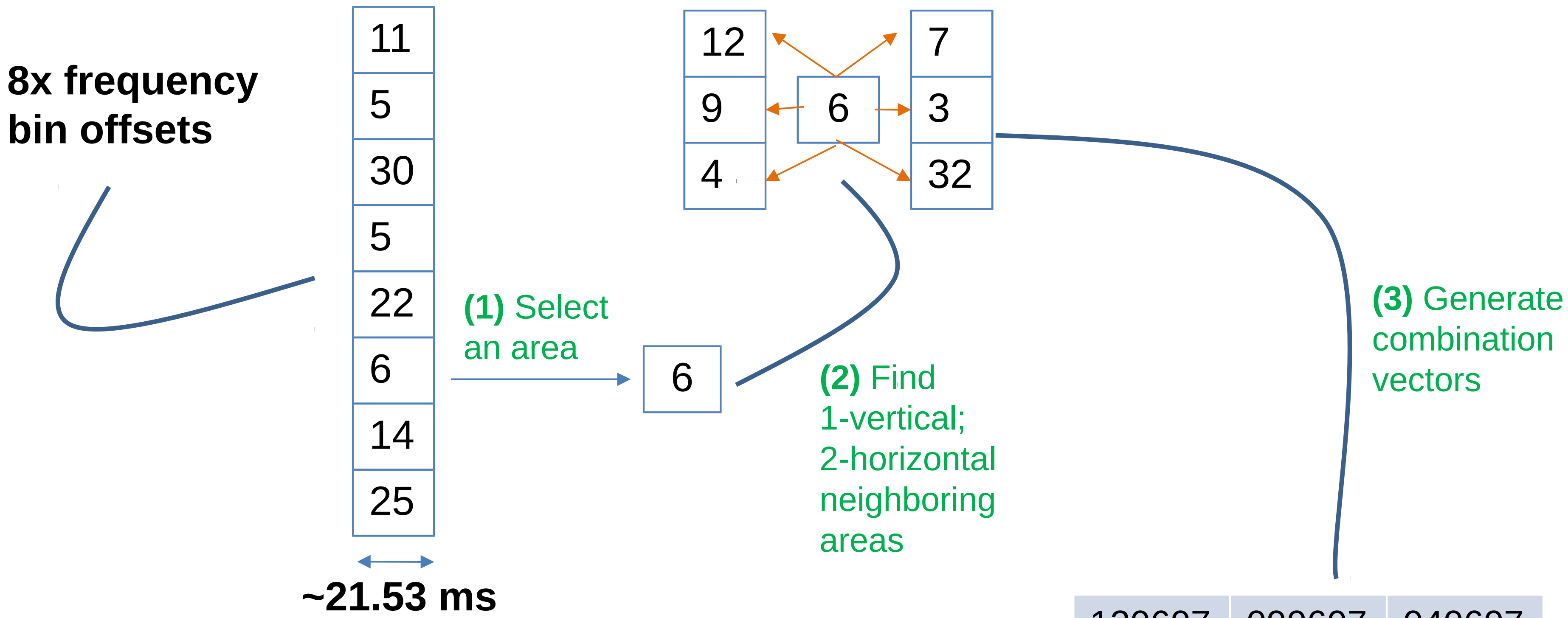
We can represent 5513 using a 16-bits integer.
16 of them occupies **256-bits** (32 bytes).

However, we can represent 32 with 5-bits.
It is possible to store them in **80-bits** (10 bytes).

| **i** | **0** | **1** | **2** | **3** | **4** | **5** | **...** | **...** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|
| **F** | 269 | 495 | 753 | 1270 | 1431 | 2045 | ... | ... | 4876 | 5285 |
| **b** | 25 | 14 | 6 | 22 | 5 | 30 | ... | ... | 5 | 11 |

$$fp = \sum_{i=0}^{16-1} \left( b[i] - i * 32 \right) * 2^{5*i}$$

# Fingerprinting: Hashing




| | | |
|---|---|---|
| 120607 | 090607 | 040607 |
| 120603 | 090603 | 040603 |
| 120632 | 090632 | 040632 |

# Fingerprinting: Key Choices

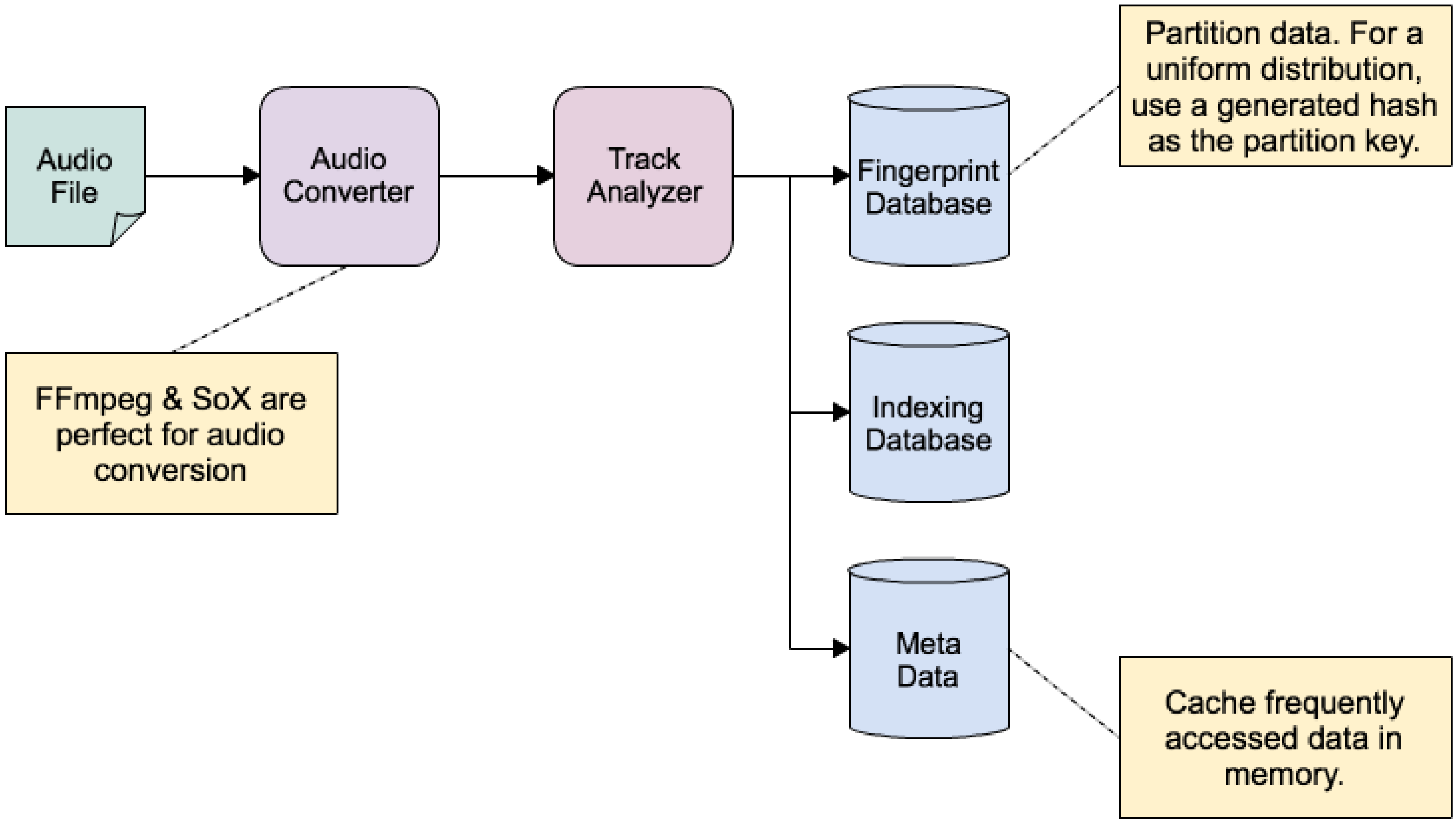
Selection of audio information

- Should be **robust**
- Should be as **unique** as possible

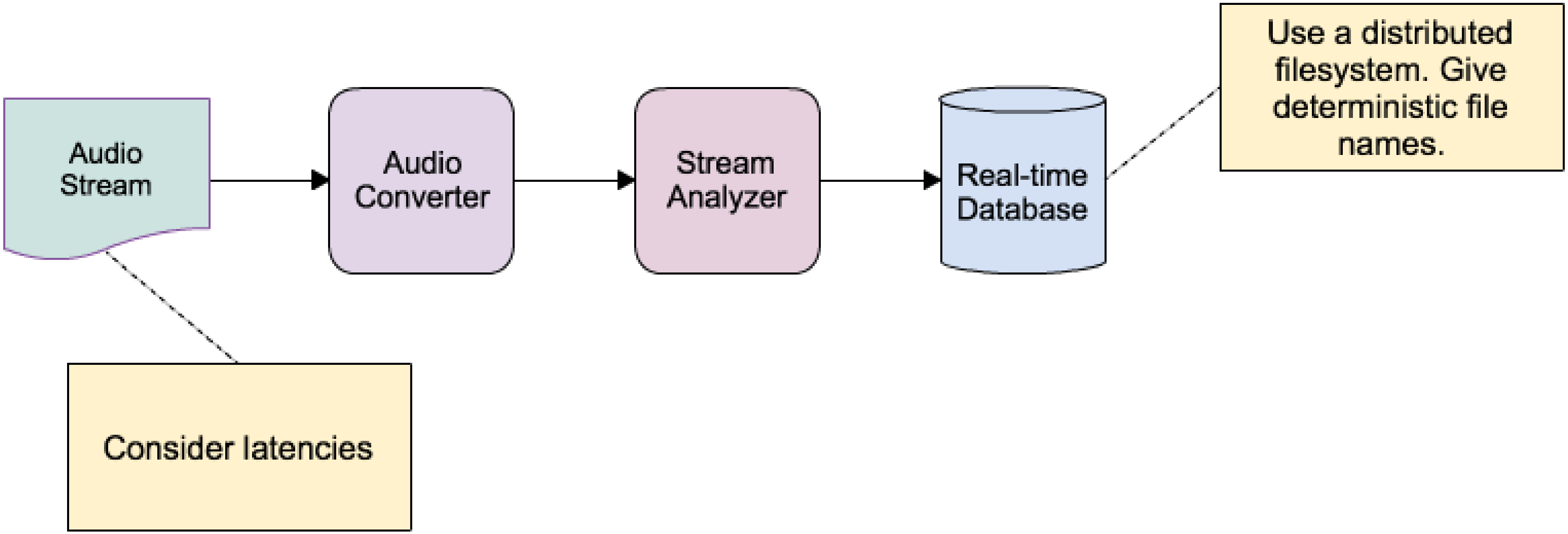
The FFT algorithm

- Managing losses due to the uncertainty principle
  - Time-resolution = 1 / Frequency-resolution
- Discrete-time FT or Short-time FT
- # of FFT points

# Static Database



# Streaming Database



# Streaming Database

**Stream name** **Timestamp**

In **YYYYMMDDHHAB** format
A: {0, 1, 2, 3, 4, 5} → High minute
B: {0, 2, 4, 6, 8} → Low minute

**FOSDEM / 201601301648.fingerprint**

**Content**: T = YYYYMMDDHHAB file contains fingerprints from the moment T to T + 4 minutes

**Reading**: At t = YYYYMMDDHHAB moment, the file corresponding to the

T = t – 2 – (B & 1)

timestamp will be opened.

**Writing**: At t = YYYYMMDDHHAB moment, files corresponding to

T1 = t – 2 – (B & 1)

T2 = T1 + 2

timestamps will be written.

# Identification

***Find the best matching fingerprint, if there is any***

Strategy

- Reduce the search space by elimination
- Rank candidates by detailed comparison

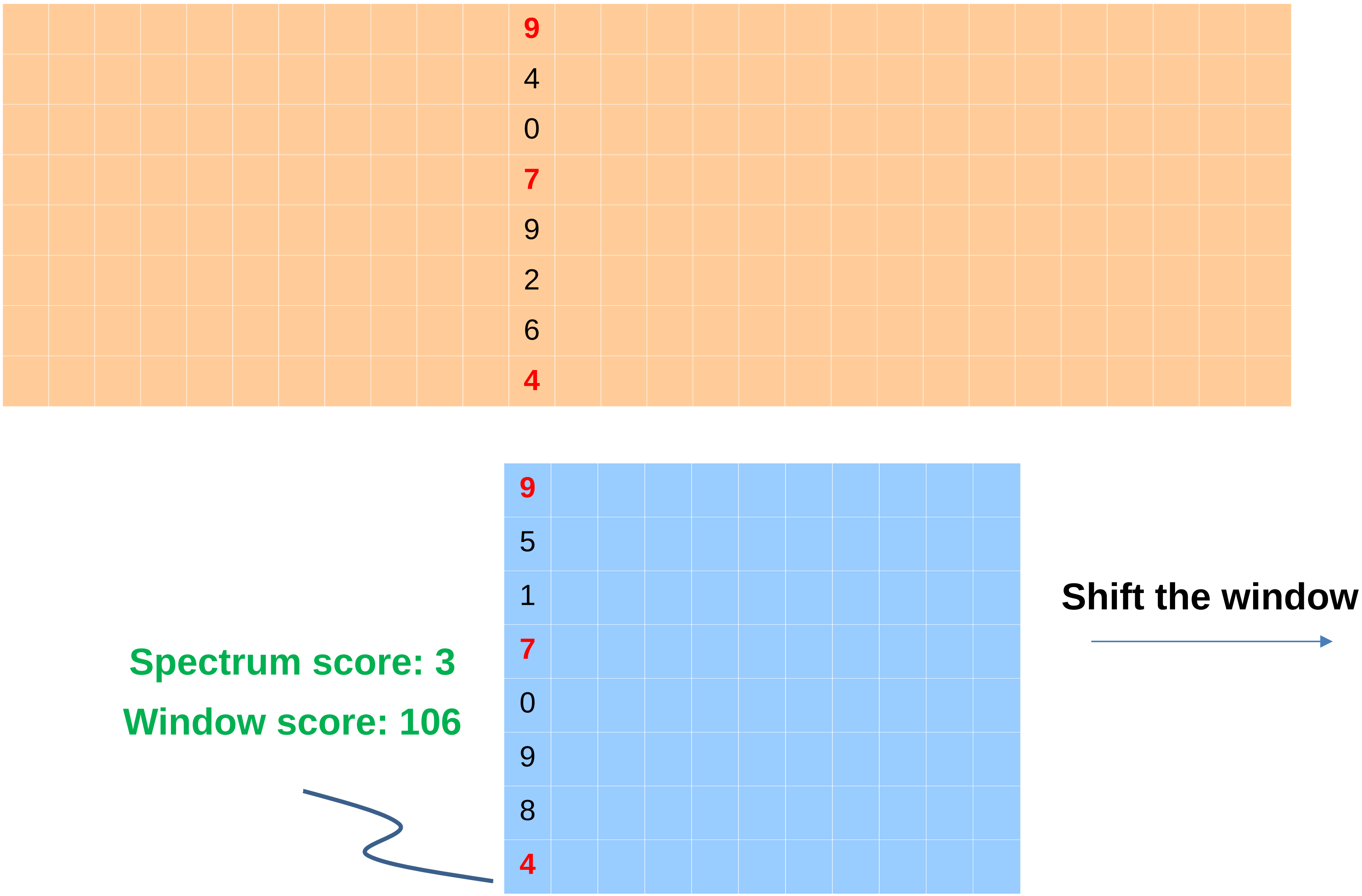
Outcomes

- **True positive:** We found the correct match
- **True negative:** We found a correct non-match
- **False negative:** We couldn't find the correct match
- **False positive:** We found an incorrect match

# Identification: Elimination

- For each hash, try to **find exact matches**.
- For each matching hash, **calculate the time difference**.
- Create a **histogram** for time difference vs. match count.
- Eliminate candidates where the best histogram **score is less than** a predefined value.

# Identification: Ranking



# Testing & Optimization

- Mix samples with:
  - White noise of varying volumes
  - Pre-recorded noise
- Record samples under different acoustic conditions
- Make the configuration dynamic and use a machine learning algorithm to select the best configuration

# THANKS!

More will be at:

github.com/wizard/fosdem2016

- Links to **open-source software**
- **Source code** for everything we talked about
- Markdown **documentation** for this presentation
- Dockerfile

# References


- FOSDEM icon: https://fosdem.org/2016/
- Email icon: https://thenounproject.com/term/mail-with-at-sign/71812/
- FFmpeg: https://www.ffmpeg.org/
- SoX: http://sox.sourceforge.net/
- Sonic Visualizer: http://www.sonicvisualiser.org/
- Audacity: http://audacityteam.org/
- PostgreSQL: http://www.postgresql.org/
- Redis: http://redis.io/
- Solr: http://lucene.apache.org/solr/