Introduction au son numérique

Sylvain Marchand

sylvain.marchand@labri.fr

Bibliographie (1/2)

- Introduction au son numérique :
  - John R. Pierce †
    The Science of Musical Sound
    Scientific American Books Inc., 1983
  - en plusieurs volumes :
    Le livre des techniques du son
    Éditions Fréquences. Eyrolles, 1994
- Informatique musicale :
  - (Traité IC2, série Informatique et systèmes d'information)
    Informatique musicale : du signal au signe musical
    François Pachet and Jean-Pierre Briot (éditeurs)
    HERMES Sciences, 2004

Bibliographie (2/2)

- Traitement du signal :
  - Alan V. Oppenheim et Ronald W. Schafer
    Discrete-Time Signal Processing
    Prentice-Hall, 1989
- Psychoacoustique :
  - Eberhard Zwicker et Richard Feldtkeller
    Psychoacoustique – L'oreille, récepteur d'information
    Masson, 1981
  - Eberhard Zwicker et Hugo Fastl
    Psychoacoustics : Facts and Models
    Springer Verlag, 1990

Définition

Le mot **son** est ambigu.

En effet, il désigne à la fois :

- la vibration physique $\longrightarrow$ acoustique
- la sensation qu'elle procure $\longrightarrow$ psychoacoustique

Physique du son : nature

onde de pression acoustique :

- créée par des molécules (d'air) qui vibrent (oscillent) autour d'une position d'équilibre
  $\longrightarrow$ ne se propage pas dans le vide
- transmission de l'énergie de proche en proche
  $\longrightarrow$ pas de déplacement de molécules

Physique du son : nature

onde de pression acoustique :

- créée par des molécules (d'air) qui vibrent (oscillent) autour d'une position d'équilibre
  $\longrightarrow$ ne se propage pas dans le vide
- transmission de l'énergie de proche en proche
  $\longrightarrow$ pas de déplacement de molécules

"Dans l'espace, personne ne vous entend crier."

Alien (slogan du film), 1979.

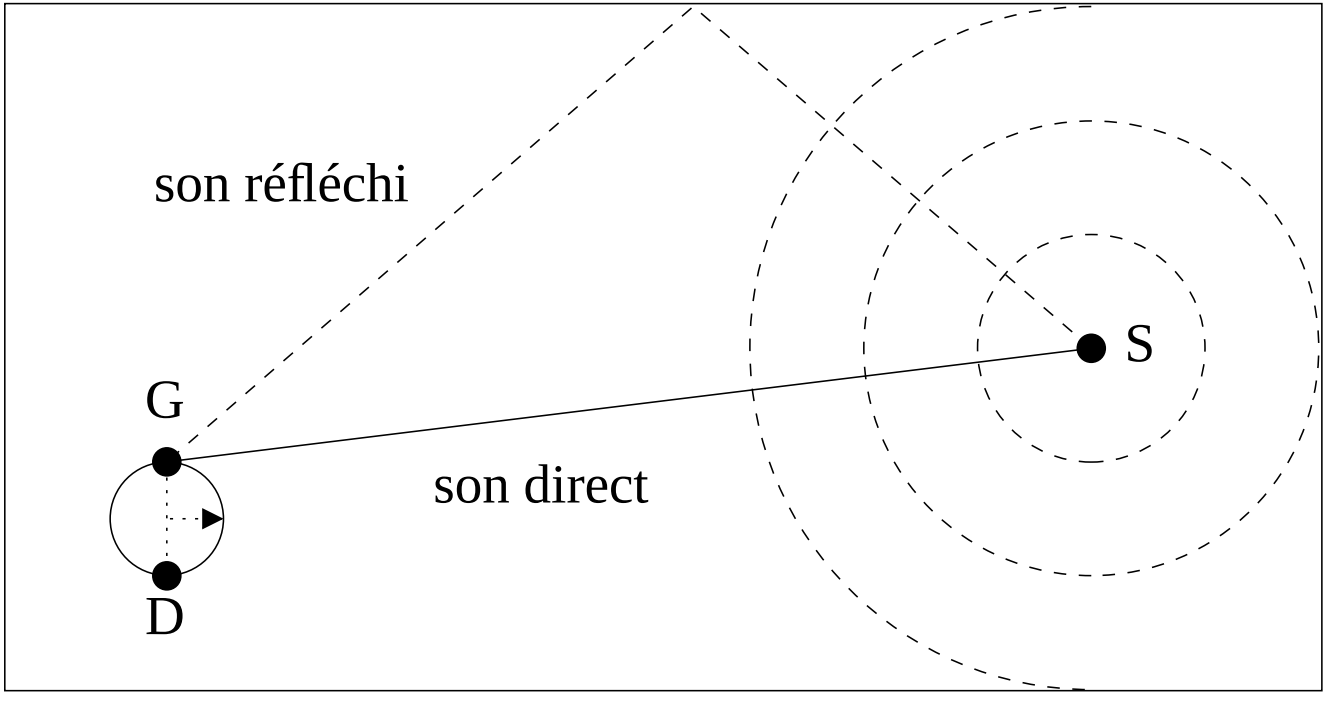
Physique du son : émission

onde émise par un objet qui vibre

2 grandes familles de sources sonores :

- à excitation entretenue (voix, violon, *etc*.)
- à impulsion initiale : percussions (cloche, gong, *etc*.)

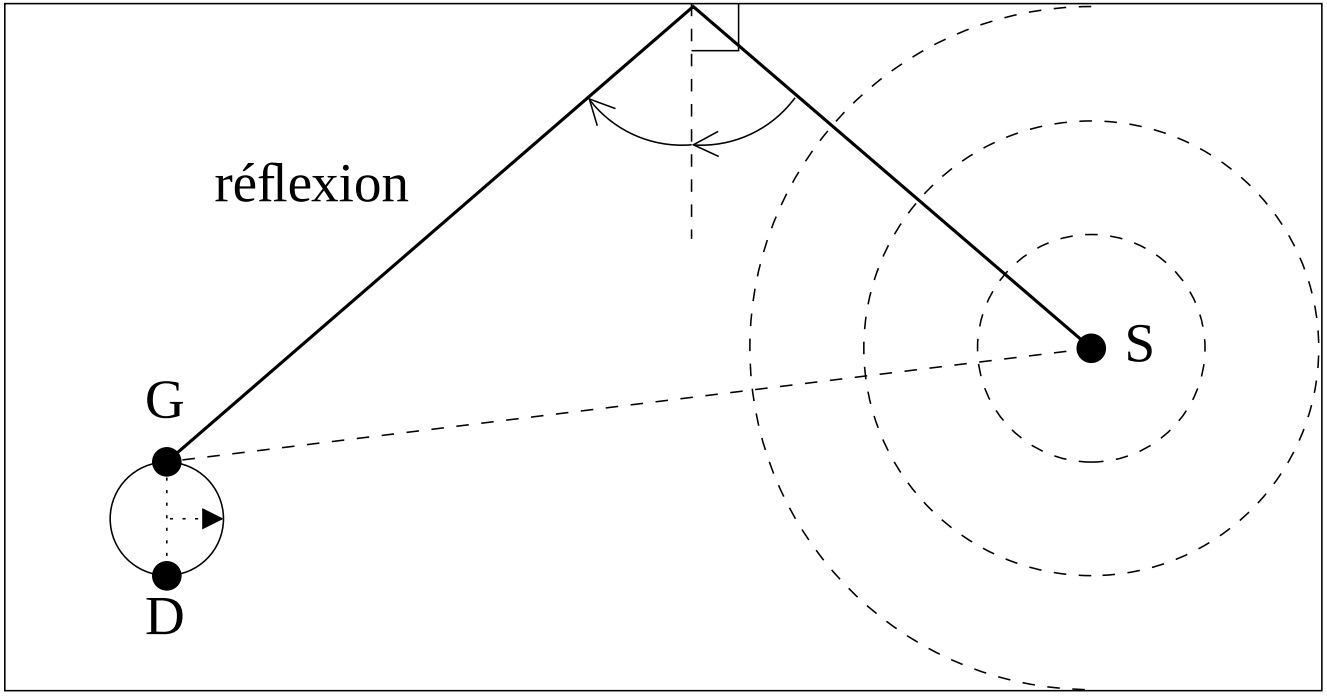
Physique du son : propagation



onde de pression **sphérique**

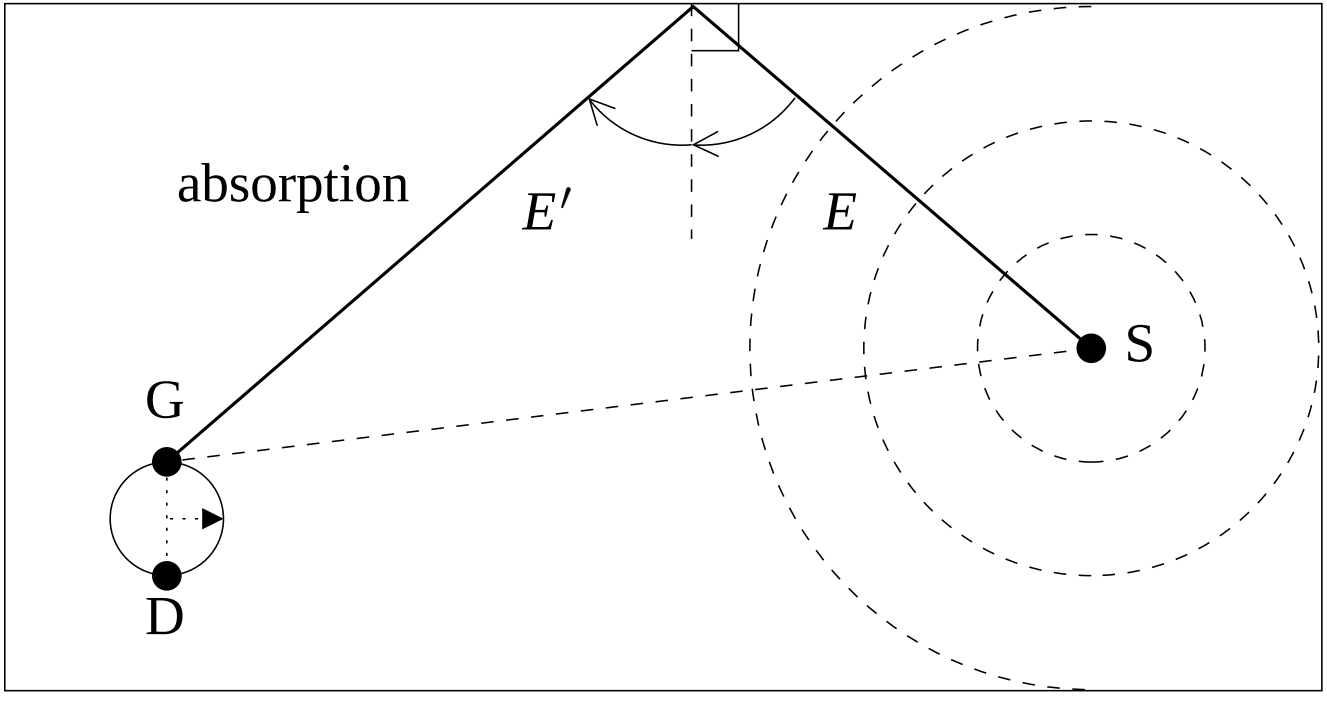
célérité : $\approx$ 330 m.s^{-1} (dans l'air)

Physique du son : propagation. . .



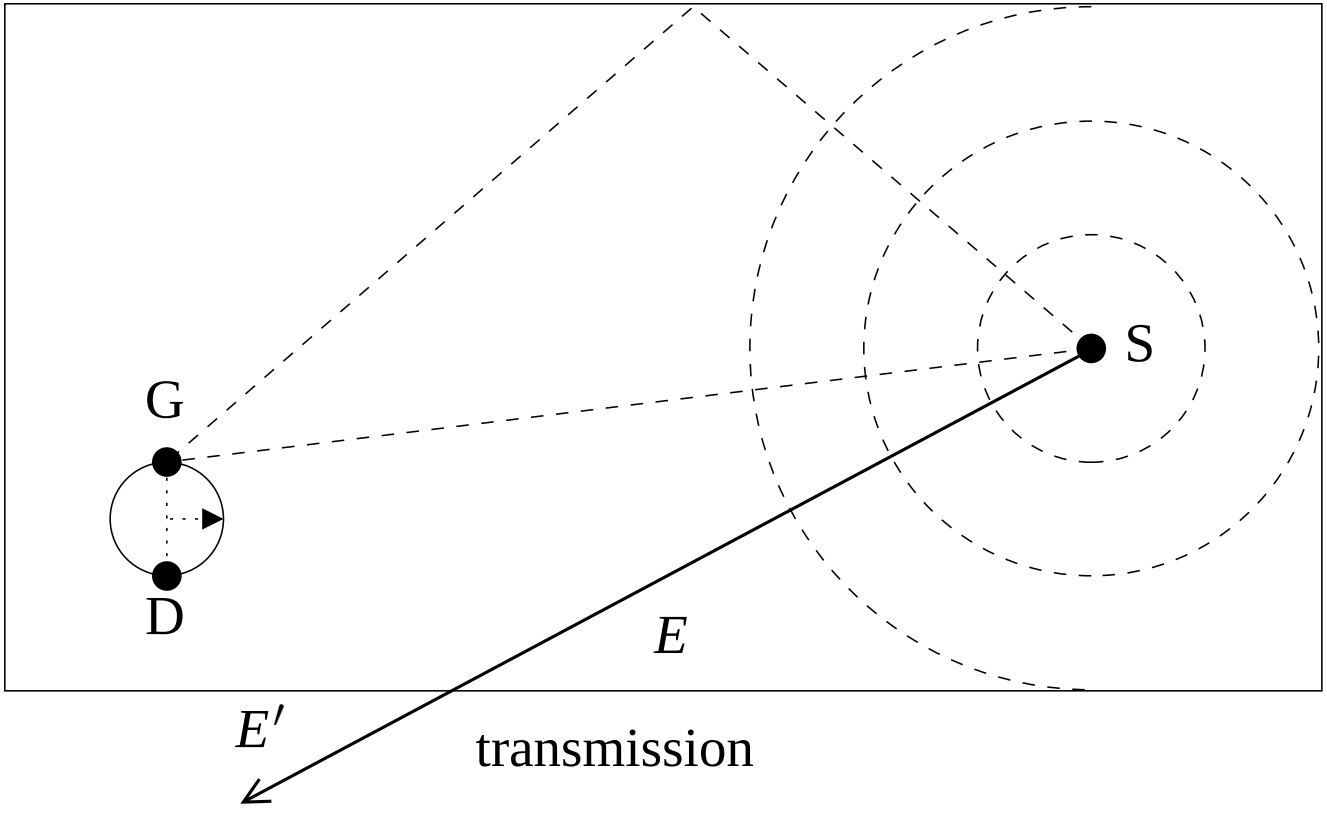
réflexion (*cf.* lois de Descartes en optique)

Physique du son : propagation. . .



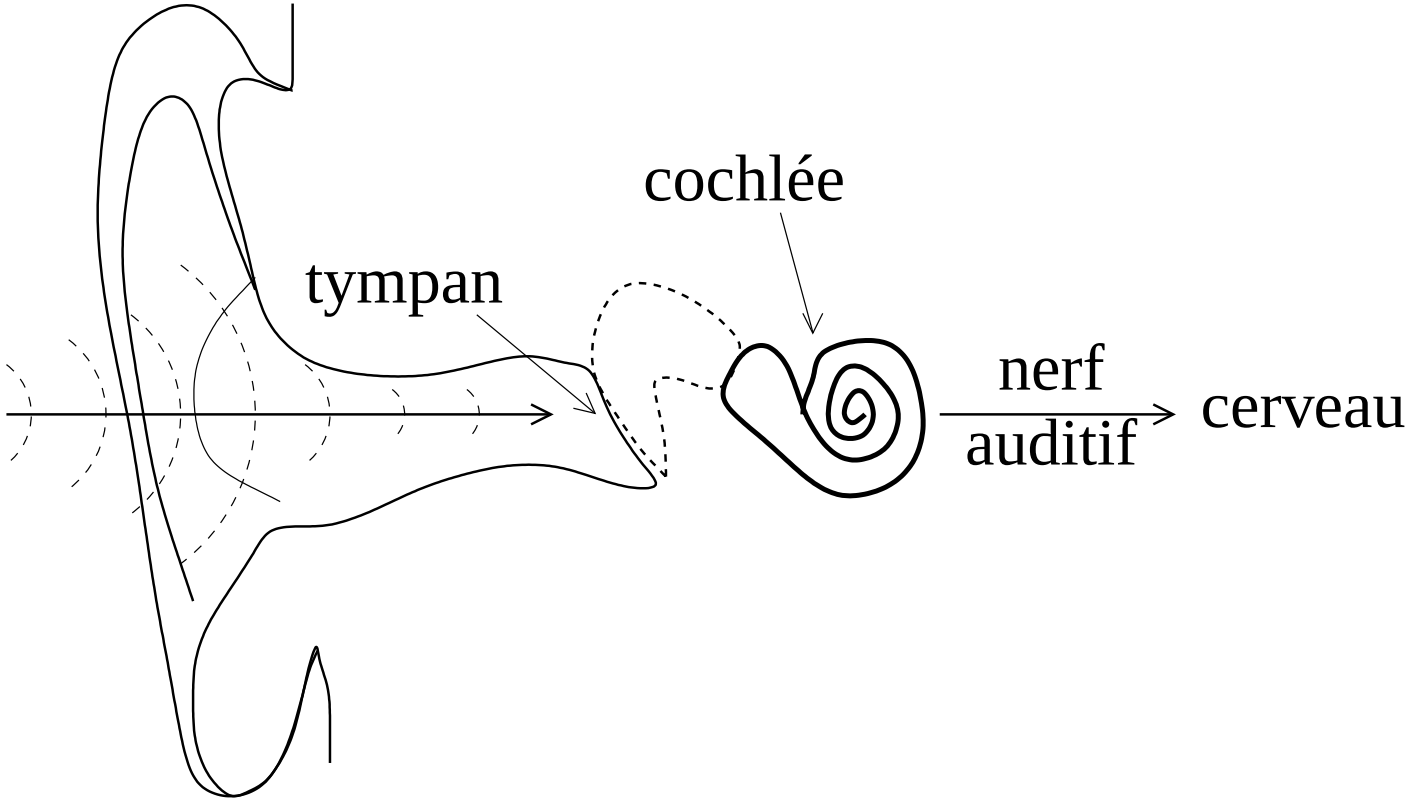
absorption : $E \neq E'$

Physique du son : propagation. . .



Physique du son : réception

réception (⟶ perception)



Perception du son

3 paramètres :

- intensité (sonie) $\longrightarrow$ volume
- hauteur (tonie) $\longrightarrow$ grave / aigu
- timbre

Perception du son

3 paramètres :

- intensité (sonie) $\longrightarrow$ volume
- hauteur (tonie) $\longrightarrow$ grave / aigu
- timbre

Le timbre est défini par ce qu'il n'est pas. . .

(c'est ce qui permet de différencier deux sons de même intensité et de même hauteur)

Perception du son

3 paramètres :

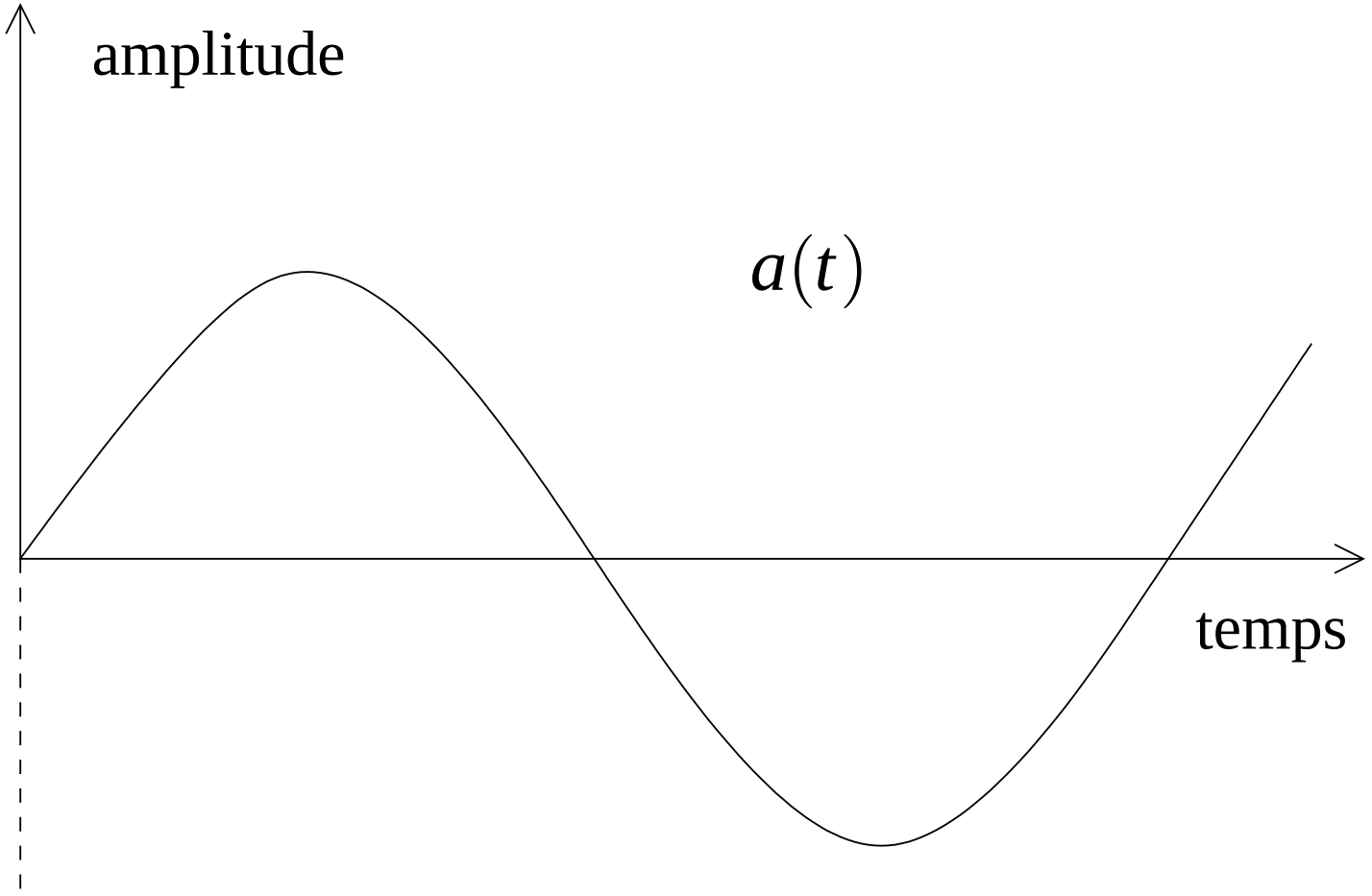
- intensité (sonie) $\longrightarrow$ volume
- hauteur (tonie) $\longrightarrow$ grave / aigu
- timbre

Le timbre est défini par ce qu'il n'est pas. . .

(c'est ce qui permet de différencier deux sons de même intensité et de même hauteur)

- (durée)
- (position dans l'espace)

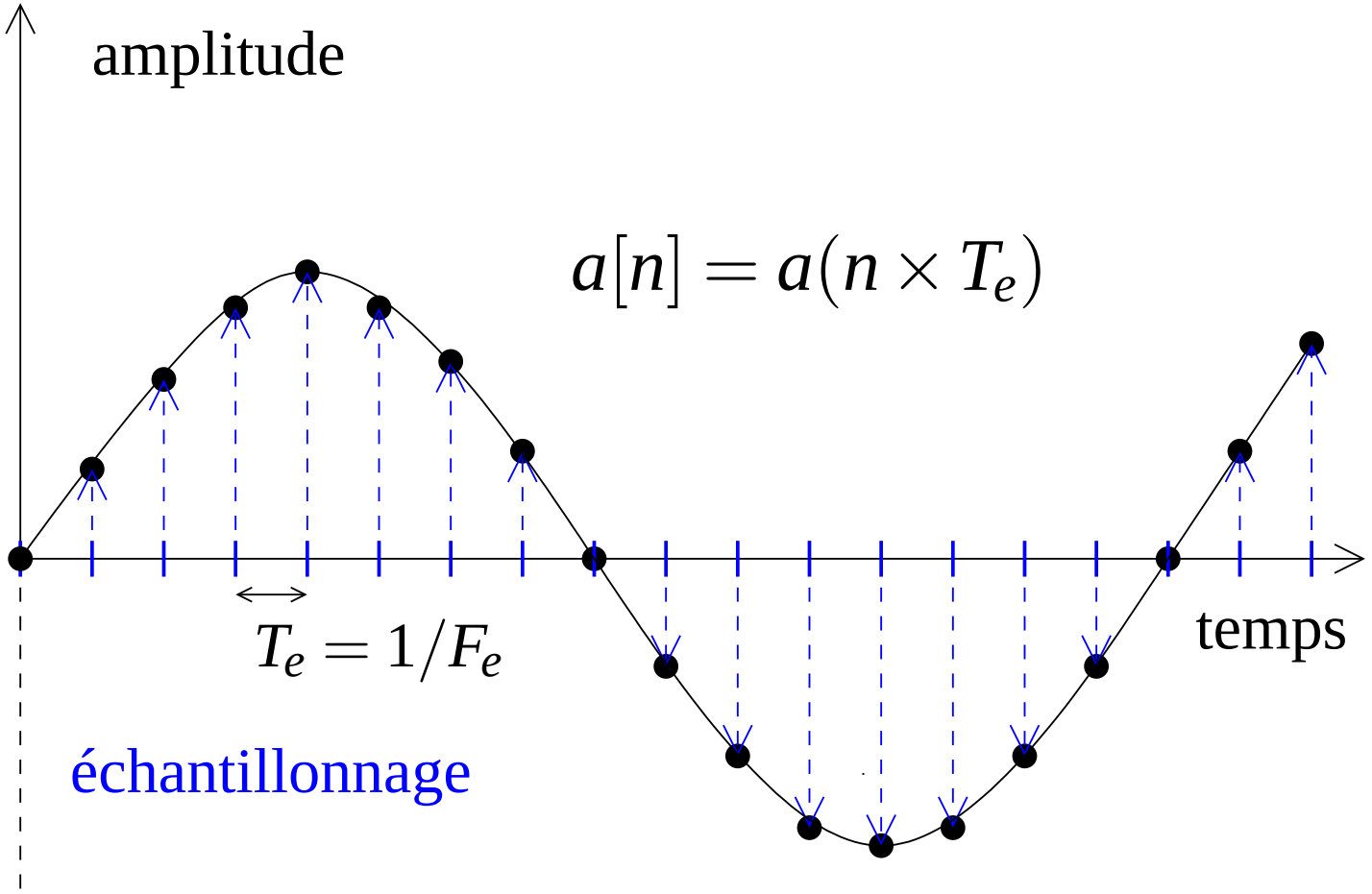
Représentation temporelle du son



$a(t)$: amplitude de l'onde en fonction du temps

(mesurée au niveau du tympan [oreille externe])

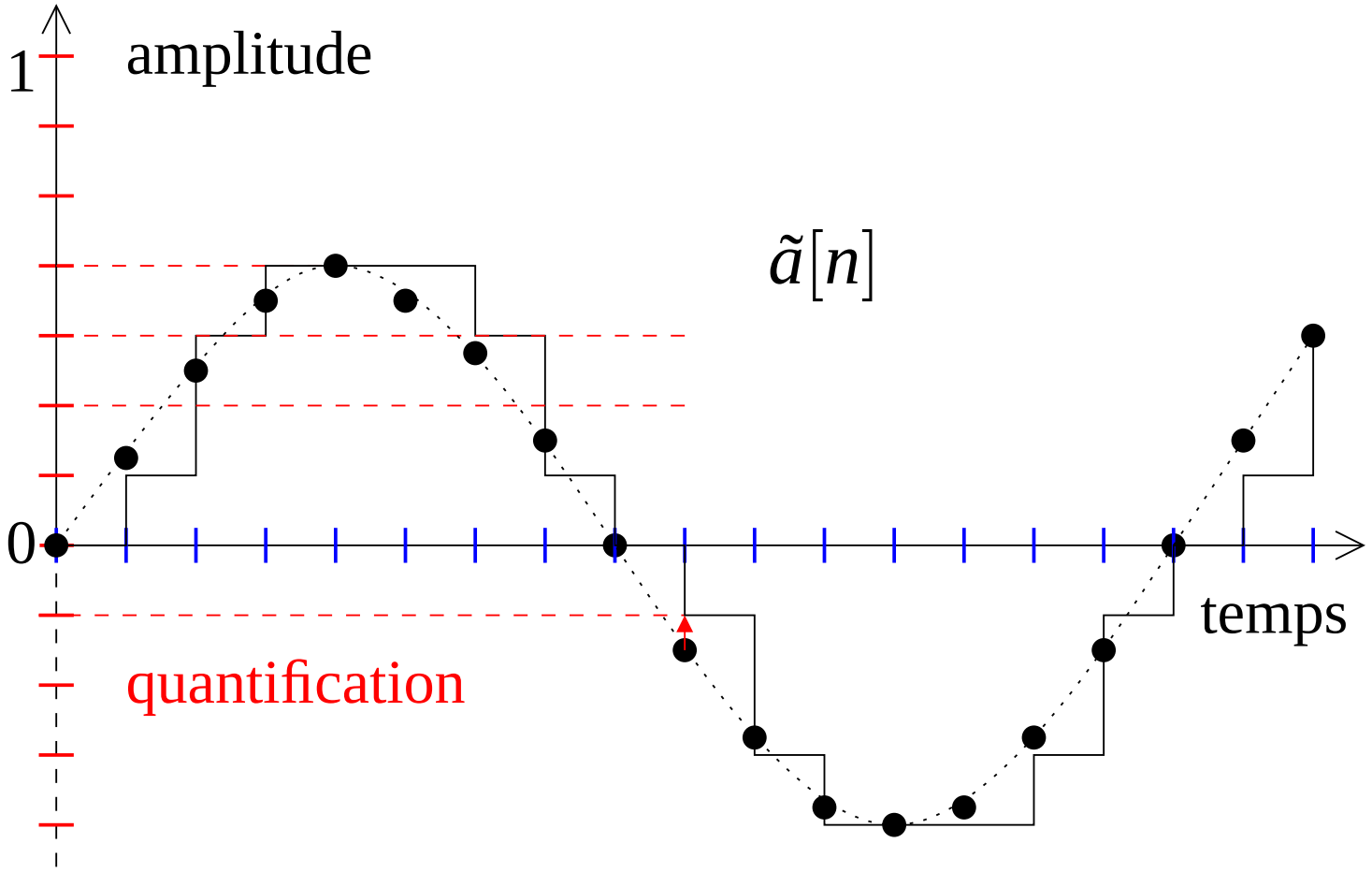
Discrétisation (1/2) : échantillonnage



fréquence d'échantillonnage F_e (inverse de la période T_e)

(CD : 44100 Hz, DAT : 48000 Hz, DVD Audio : 96000 Hz)

Discrétisation (2/2) : quantification

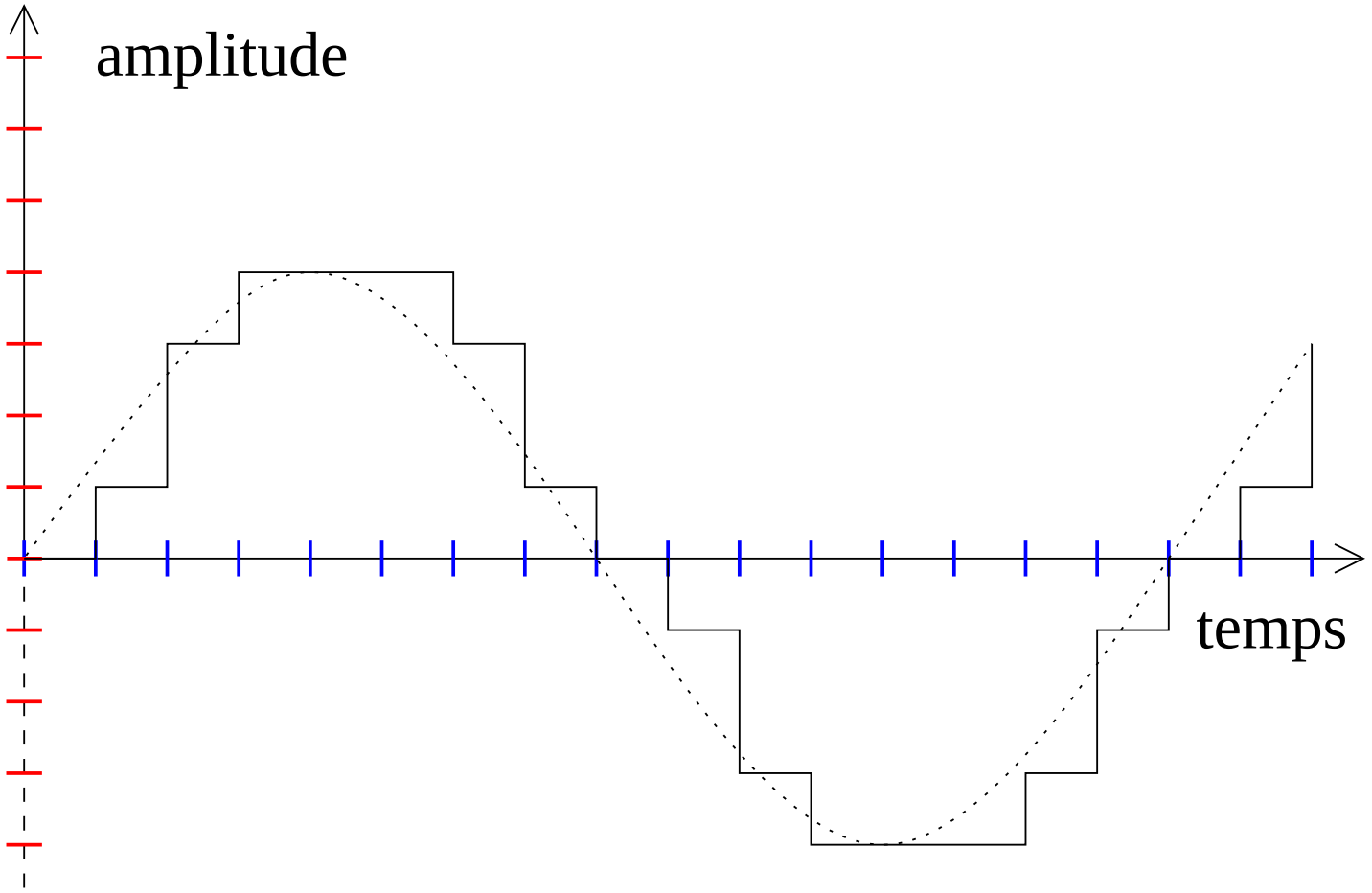


quantification sur k bits $\longrightarrow$ 2^k valeurs possibles

(anciennement : 8 bits, CD : 16 bits, DVD Audio : 24 bits)

$\longrightarrow$ erreur de quantification ($a - \tilde{a}$) : $2/2^k = 2^{1-k}$

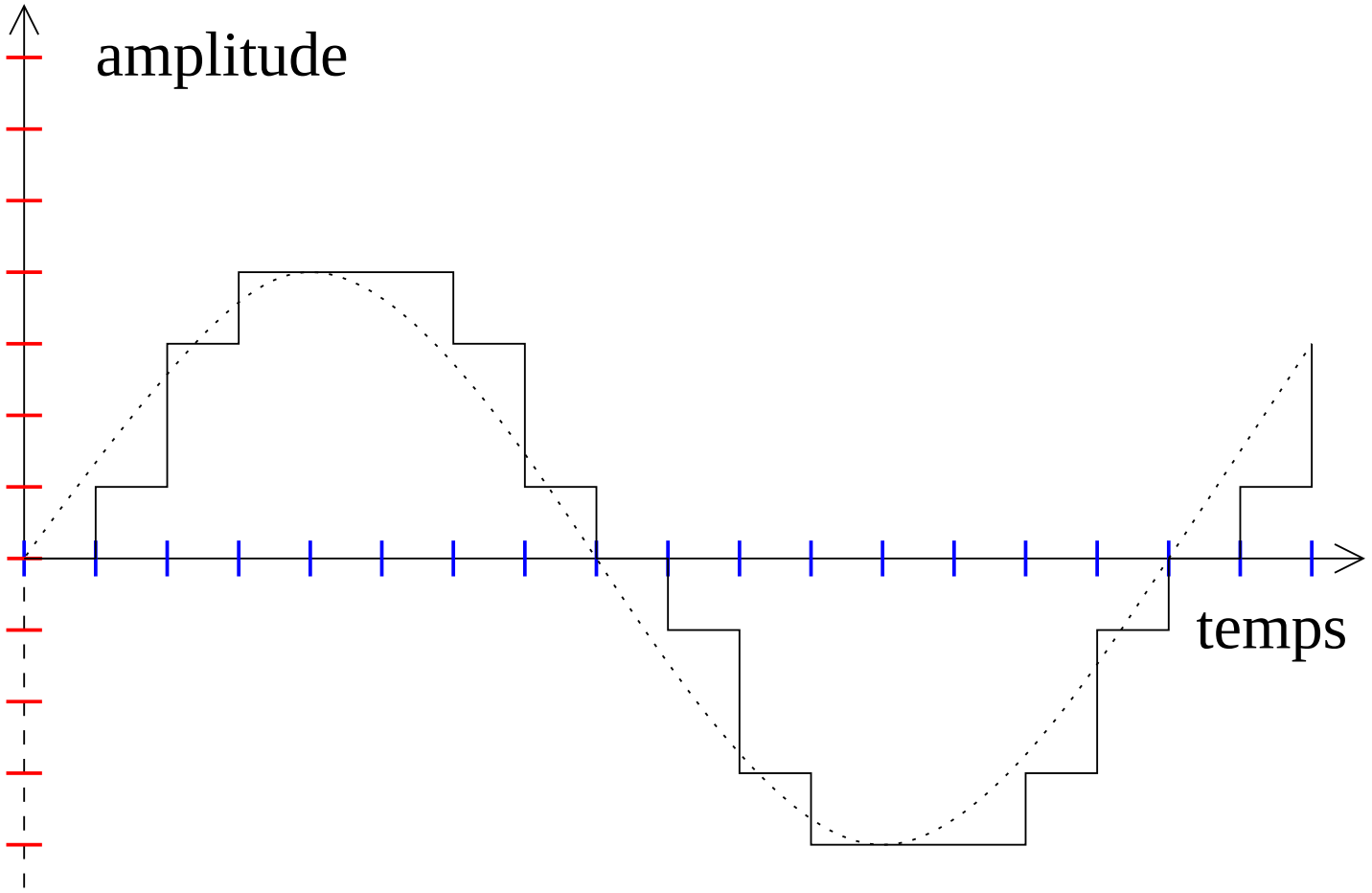
Représentation discrète



discrétisation (numérisation) =
échantillonnage + quantification

→ *Pulse-Code Modulation* (PCM)

Représentation discrète



discrétisation (numérisation) =
échantillonnage + quantification

[0, 1, 3, 4, 4, 4, 3, 1, 0, -1, -3, -4, -4, -4, -3, -1, 0, 1, 3]

Représentation en machine

- vecteur d'échantillons (scalaires)
  - représentation entière :
    échantillons entre -2^{k-1} et $2^{k-1}-1$

```
short son[] =
  { 0, 1, 3, 4, 4, 4, 3, 1, 0, -1, -3, -4, -4, -4, -3, -1, 0, 1, 3 };
```

  - représentation flottante : **échantillons entre -1.0 et 1.0**

```
#define LENGTH 19
#define K 4
#define STEP (1.0 / pow(2,K-1))

float son2[LENGTH];

for (i=0; i<LENGTH; i++)
  {
    son2[i] = son[i] * STEP;
  }
```

Reconstruction...

- condition de Nyquist : $F_e > 2 \times F_{\text{max}}$
  (F_{max} : plus grande fréquence présente dans le son a)
- théorême de Shannon :
  le signal continu $a(t)$ peut être reconstruit parfaitement
  (sans erreur) à partir du signal discret $a[n]$
  si et seulement si
  la condition de Nyquist est respectée
- exemples :
  - CD : $F_e = 44100$ Hz $\Rightarrow F_{\text{max}} < 22050$ Hz
  - DVD : $F_e = 96000$ Hz $\Rightarrow F_{\text{max}} < 48000$ Hz

remarque : la plus grande fréquence audible $\approx$ 20 kHz...

Représentation des échantillons

son **numérique** $\longrightarrow$ nombres :

- arithmétique : entière / flottante
- calcul : non signé / signé
- bits : 8 / 16 / 24 / 32 / 64
- représentation machine : *little / big endian*

exemples :

- anciennes cartes son : entiers 8 bits non signés
- CD *(Compact Disc)* : entiers 16 bits signés *big-endian*
- DVD *(Digital Versatile Disc)* : entiers 24 bits signés
- JACK (voir plus loin) : flottants 32 (voire 64) bits

+ choix pour la polyphonie (ex : stéréo Gauche / Droite) :

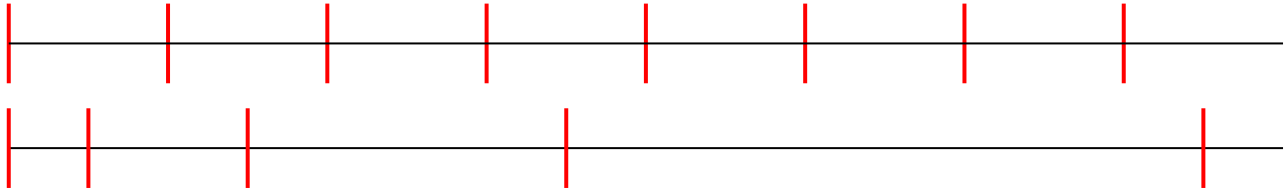
- séparé : G G $\cdots$ G D D $\cdots$ D
- **entrelacé** : G D G D $\cdots$ G D

Formats sonores (1/3)

- formats non compressés
  - formats bruts :
    - PCM *(Pulse-Code Modulation)*
    - CDA *(Compact Disc Audio)* *big-endian*
    - CDR *(Compact Disc Raw)* *big-endian*
    - “raw”...
  - structurés (en-tête) :
    - AIFF *(Audio Interchange File Format)*
    - WAV *(Waveform Audio File Format)*

Formats sonores (2/3)

- compressions sans perte
  - `gzip` : inefficace. . .
  - quantification logarithmique : codage μ-*law* AU, SND



8 bits logarithmiques sont perceptivement équivalents à 12 bits linéaires

- ADPCM *(Adaptive-Delta Pulse-Code Modulation)*

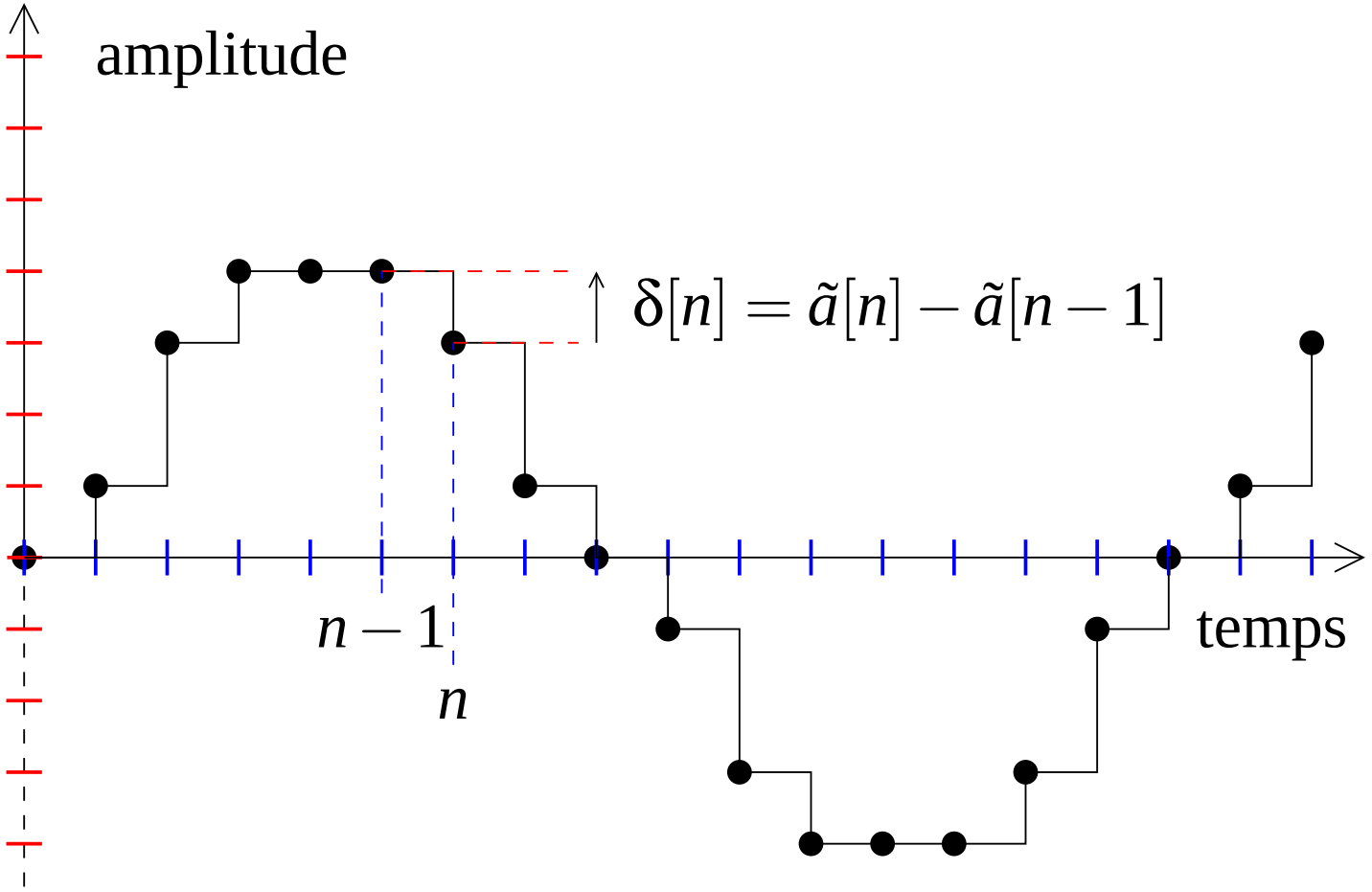
  au lieu de coder $\tilde{a}[n]$,

  calcul du multi-ensemble des différences $\Delta = \bigcup_n \{\delta[n] = \tilde{a}[n] - \tilde{a}[n-1]\}$,

  puis calcul l'histogramme des occurences de δ dans Δ,

  et codage de δ avec un nombre de bits utilisés inversement proportionnel au nombre de ses occurrences dans Δ (Huffman)

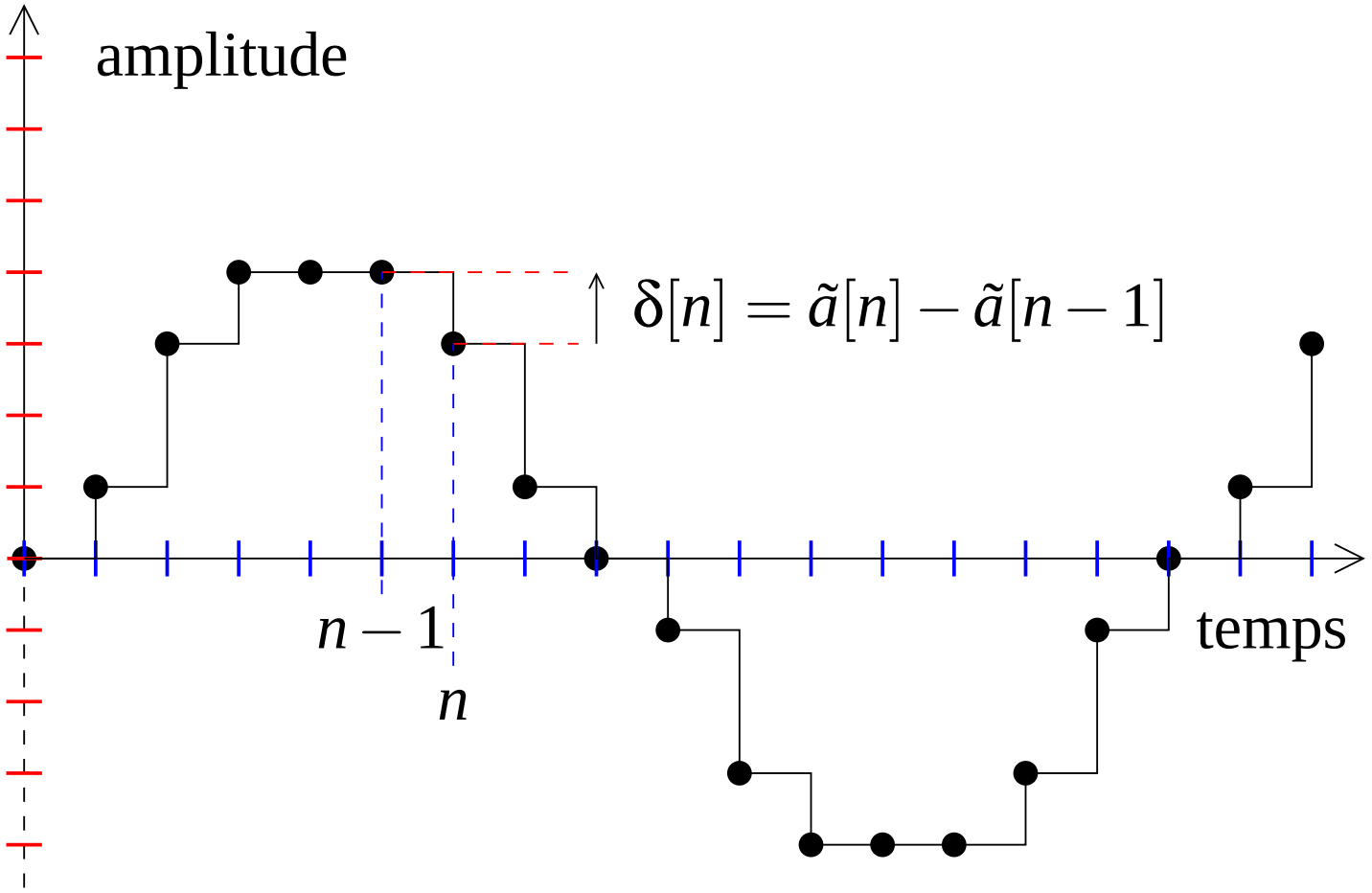
Codage ADPCM



$$\Delta = \{-2(2), -1(4), 0(4), 1(5), 2(3)\}$$

$\rightarrow$ *Adaptive-Delta Pulse-Code Modulation* (ADPCM)

Codage ADPCM



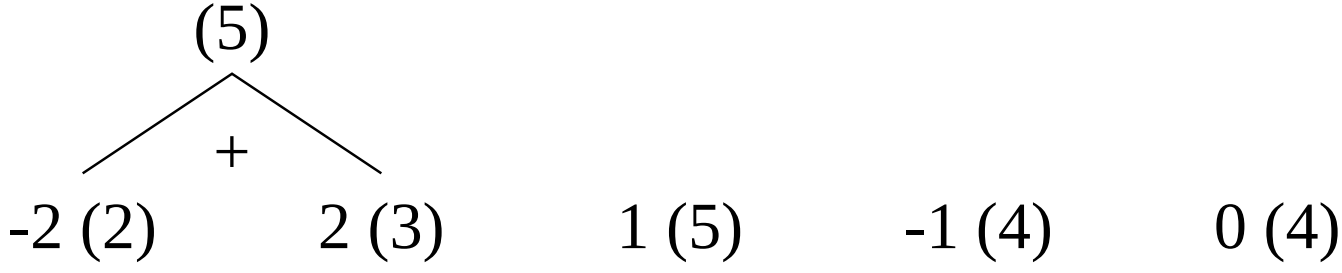
$$\Delta = \{-2(2), -1(4), 0(4), 1(5), 2(3)\}$$

```
000 10 11 01 001
```

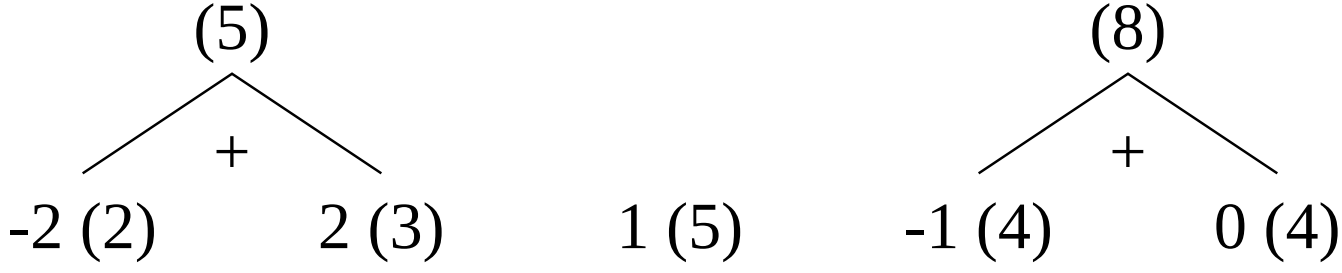
Codage de Huffman. . .

-2 (2) 2 (3) 1 (5) -1 (4) 0 (4)

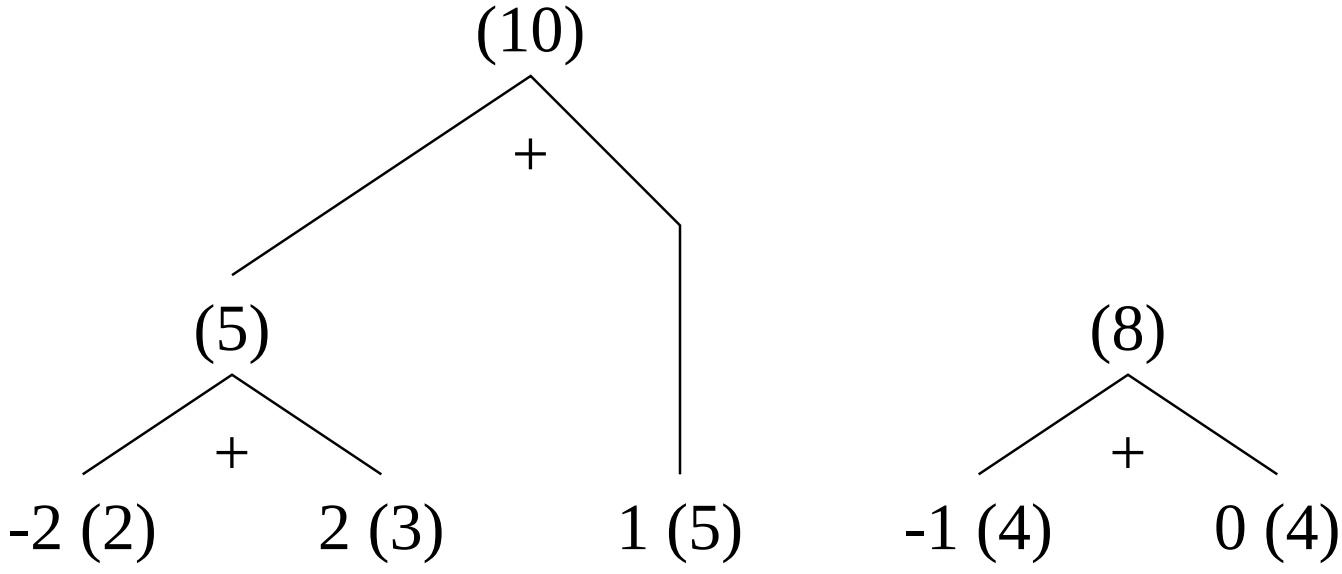
Codage de Huffman. . .



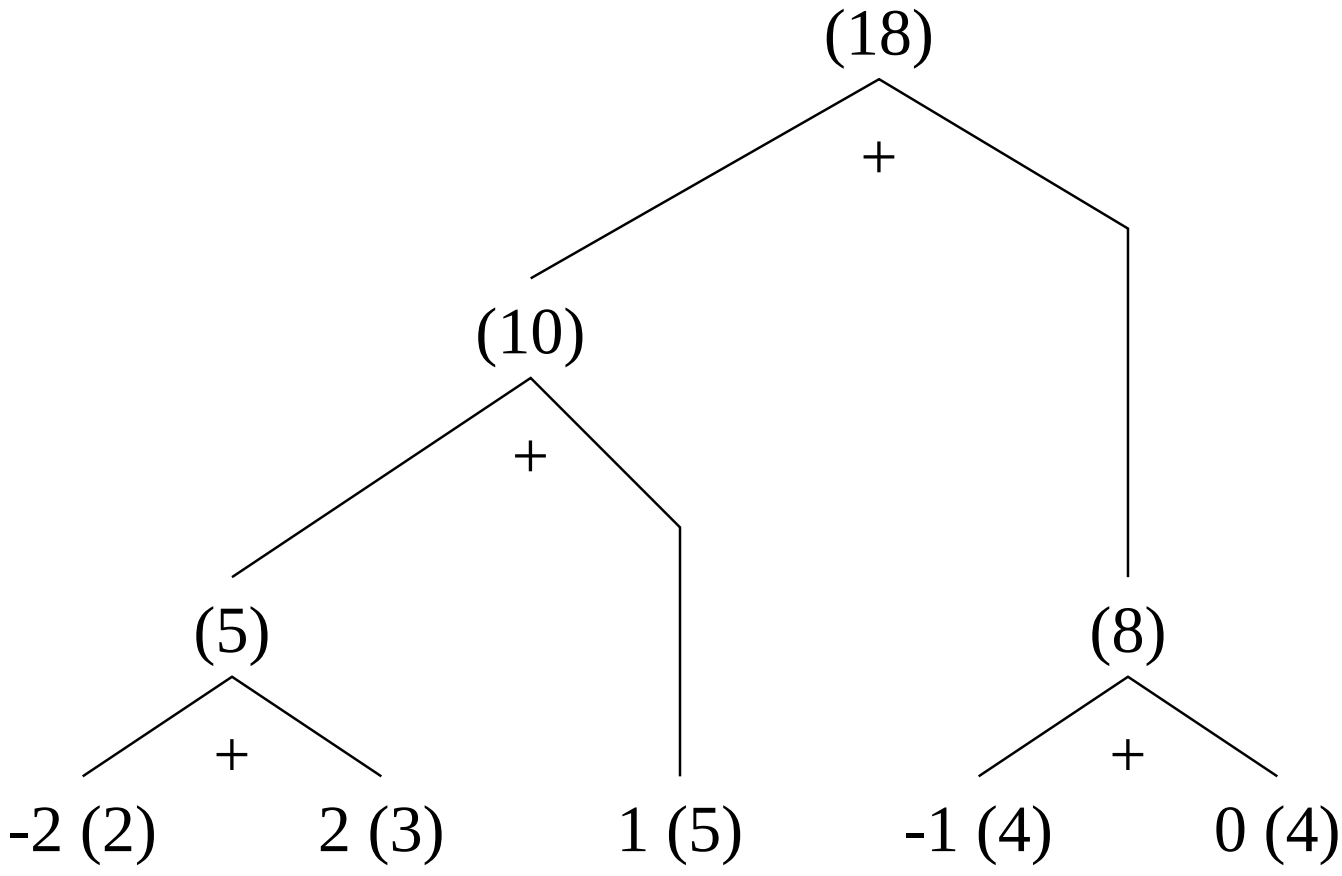
Codage de Huffman. . .



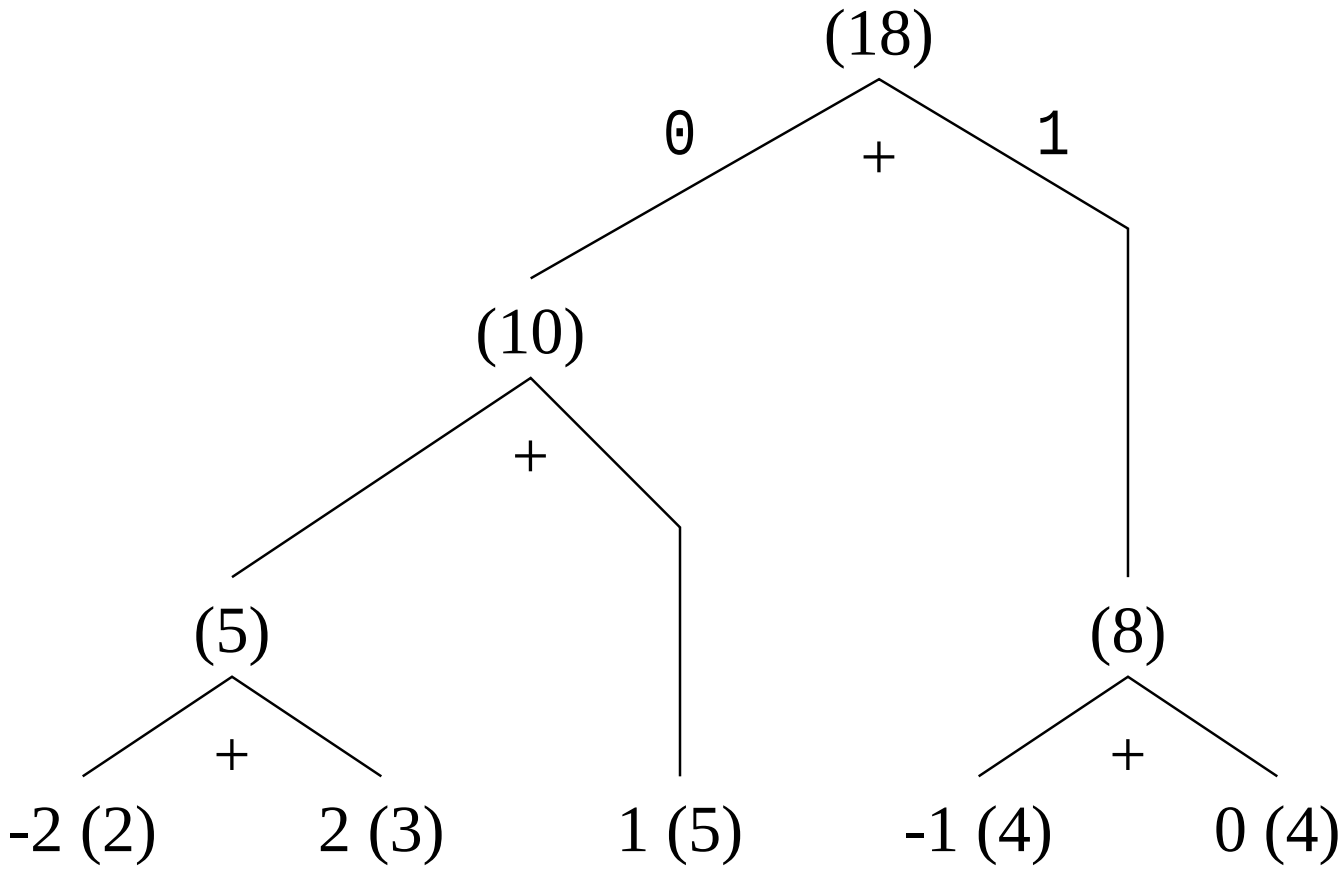
Codage de Huffman. . .



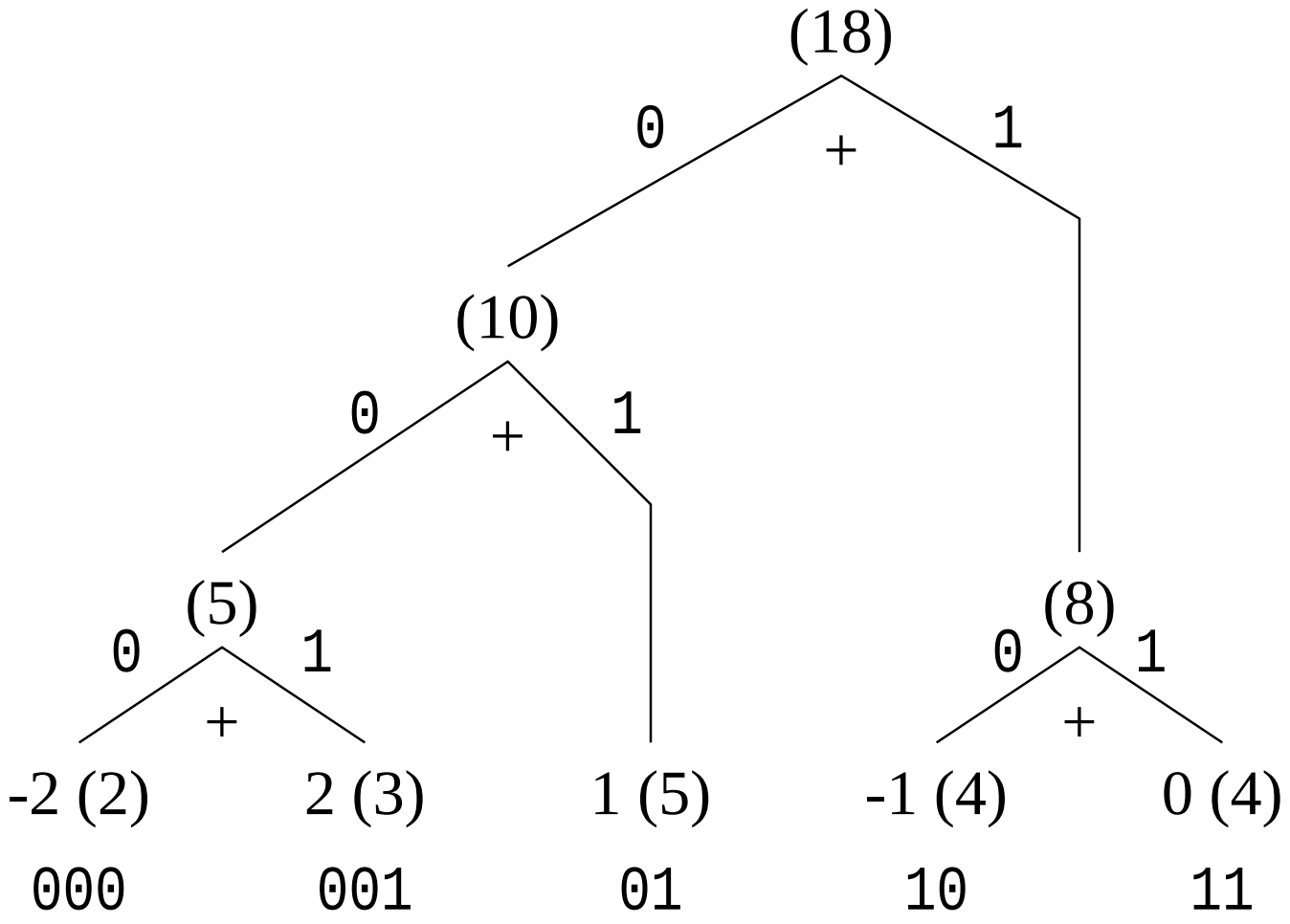
Codage de Huffman. . .



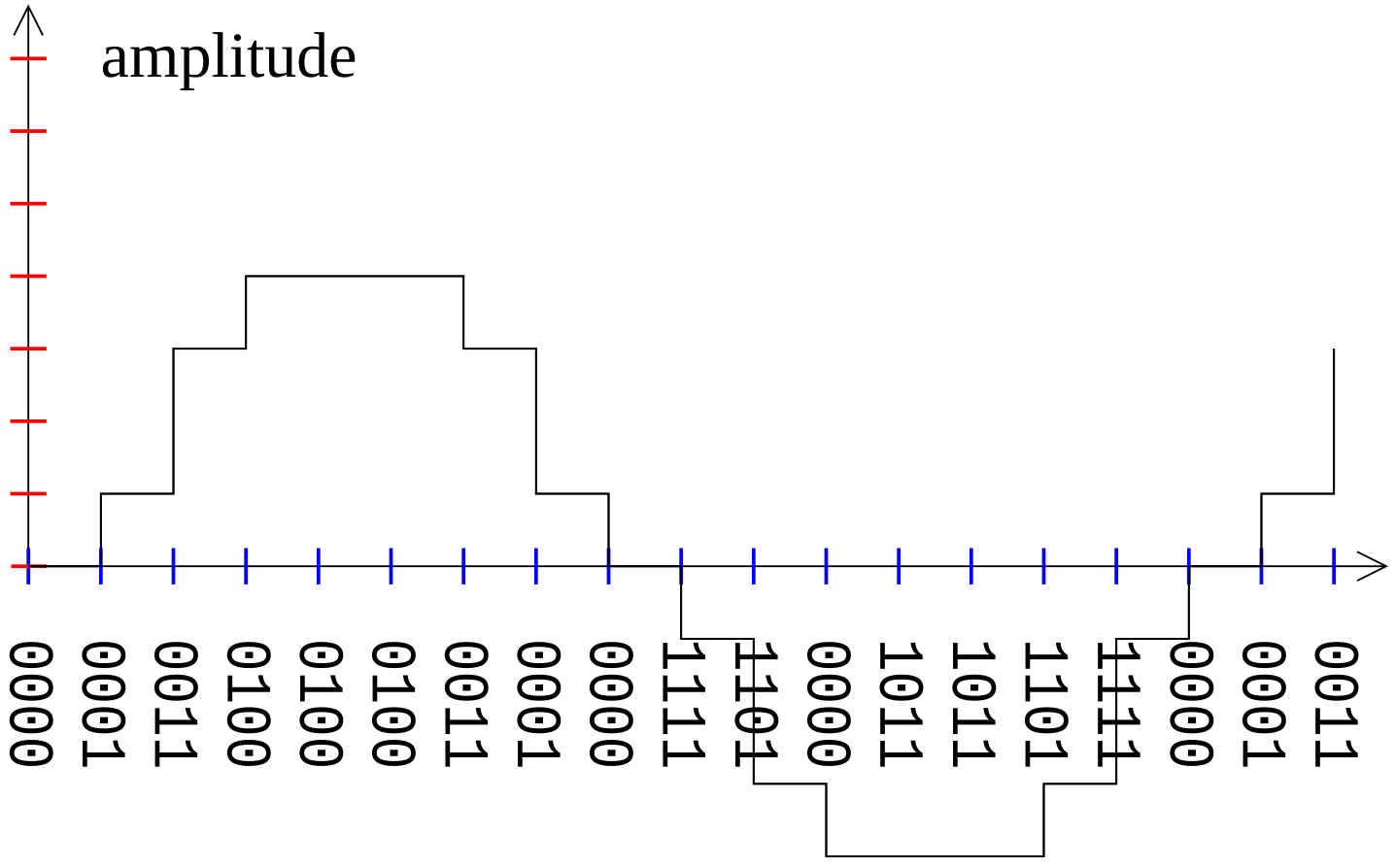
Codage de Huffman. . .



Codage de Huffman...

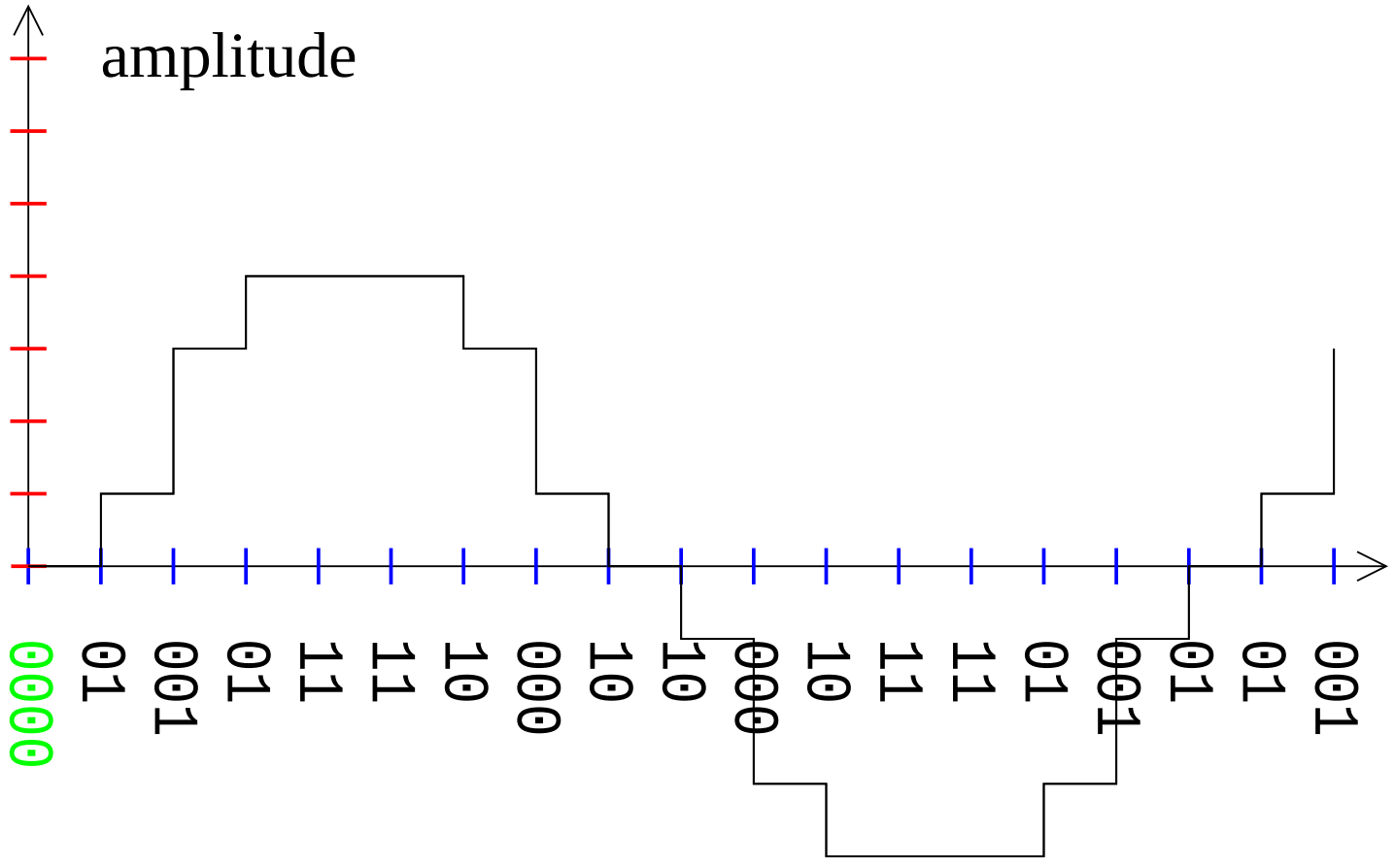


Comparatif PCM / ADPCM



19 mots de 4 bits $\longrightarrow$ 76 bits

Comparatif PCM / ADPCM



$\longrightarrow$ 41 bits seulement. . .

Formats sonores (3/3)

- compressions avec pertes (inaudibles. . .)
  - **MPEG Audio I Layer 3 (MP3)**
  - MP3 Pro
  - Dolby® AC3
  - MPEG-4 AAC *(Advanced Audio Coding)*
  - . . .

principe général :

- analyse spectrale $\longrightarrow$ domaine fréquentiel
- modèle psychoacoustique $\longrightarrow$ degré de perceptibilité
- allocation de bits proportionnelle à la perceptibilité
- compression sans perte (Huffman)

Interchange File Format **(IFF)**

- principe général (texte, son, image, vidéo, *etc*.)
- standard indépendant
- compatibilité totale :
  - compatibilité ascendante :

    les nouveaux logiciels doivent être capables de lire (**sans effort**) les anciens fichiers
  - compatibilité **descendante** :

    les anciens logiciels doivent être capables de lire aussi les nouveaux fichiers. . .

“Chunks”

- solution adoptée : système en-tête *(header)* + *chunks*
- structure d’un *chunk* :
  - type *(ID)*
  - taille *(size)*
  - données *(data)*
- compatibilité :
  - chunks extensibles (versions ultérieures)
  - nouveaux chunks “optionnels”
- algorithme :
  - *chunk* connu ?
  - non : l’ignorer, en passant au suivant (grâce à sa taille)
  - oui : selon que sa taille est
    - trop grande : lire les premières données et ignorer les autres
    - correcte : OK
    - trop petite : compléter avec des valeurs par défaut normalisées

Formats AIFF et RIFF Wave

- AIFF :
  Audio Interchange File Format
  - Apple + Commodore Amiga + SGI *(Silicon Graphics, Inc.)*
  - *big-endian* (Motorola, IBM, MIPS, *etc.*)
- RIFF WAVE :
  Resource Interchange File Format – Waveform
  - Microsoft
  - *little-endian* (Intel)

exemple : une minute de silence. . .qualité CD !

$n = 60 \times 44100$ échantillons de 2×2 octets (16 bits, stéréo)

Format AIFF : un exemple

champ	taille (octets)	type	valeur
name	4	texte	`FORM`
size	4	entier	$4+(8+18)+(8+(8+4n))$
type	4	texte	`AIFF`
name	4	texte	`COMM`
size	4	entier	18
channels	2	entier	2 (stéréo)
frames	4	entier	n
bits	2	entier	16
rate	10	réel *(IEEE extended double)*	44100.0
name	4	texte	`SSND`
size	4	entier	$8+4n$
offset	4	entier	0
block size	4	entier	0
data	$2\times 2\times n$	entiers (16 bits)	$0\ 0\ \cdots\ 0\ 0$

Format Wave : un exemple

champ	taille (octets)	type	valeur
name	4	texte	`RIFF`
size	4	entier	$4+(8+16)+(8+4n)$
type	4	texte	`WAVE`
name	4	texte	`fmt_` ($\leftarrow$ _ représentant un espace ' ')
size	4	entier	16
format	2	entier	1 (PCM)
channels	2	entier	2 (stéréo)
rate	4	entier	44100
bytes per second	4	entier	$rate \times channels \times \lceil bits/8 \rceil = rate \times 4$
block align	2	entier	2
bits	2	entier	16
name	4	texte	`data`
size	4	entier	$4n$
data	$2 \times 2 \times n$	entiers (16 bits)	$0\ 0 \cdots 0\ 0$

Utilitaire `sox` *(Sound eXchange)*

usage :

`sox` *source destination*

- conversion de types de fichiers :

  `sox son.aiff son.wav`
- retrait de l'en-tête...

  `sox son.wav -r 44100 -c 1 -sw son.raw`
- rajout de l'en-tête...

  `sox -r 44100 -c 1 -sw son.raw son.wav`
- ré-échantillonnage :

  `sox son.wav -r 48000 son2.wav`

Utilitaires divers...

- `file` : indique le type de fichier + informations

```
file son.wav
```

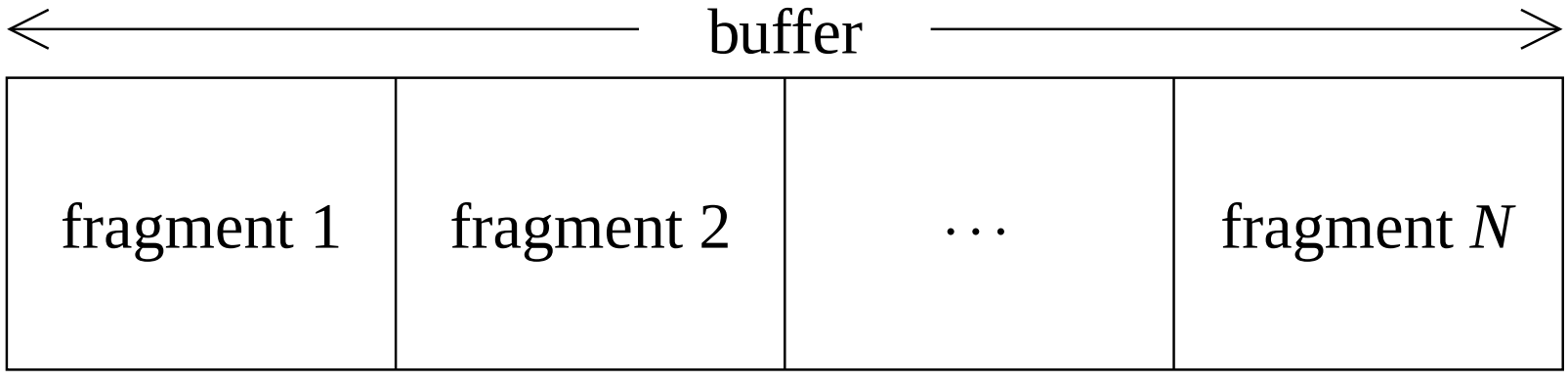
- `play` : joue un fichier son

```
play son.wav
```

```
play -d /dev/dsp1 son.wav
```

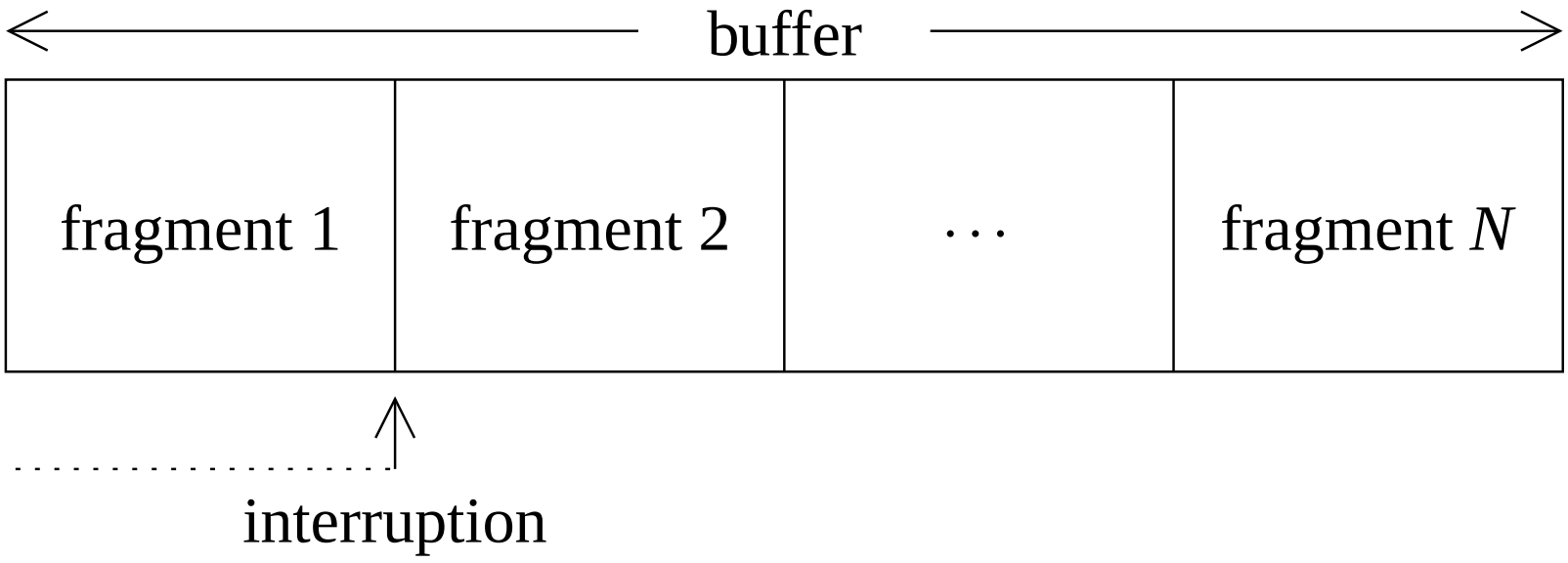
Carte son

- convertisseur numérique $\leftrightarrow$ analogique
- buffer interne d'échantillons
- division en fragments (parcours circulaire)
- mécanisme d'interruption



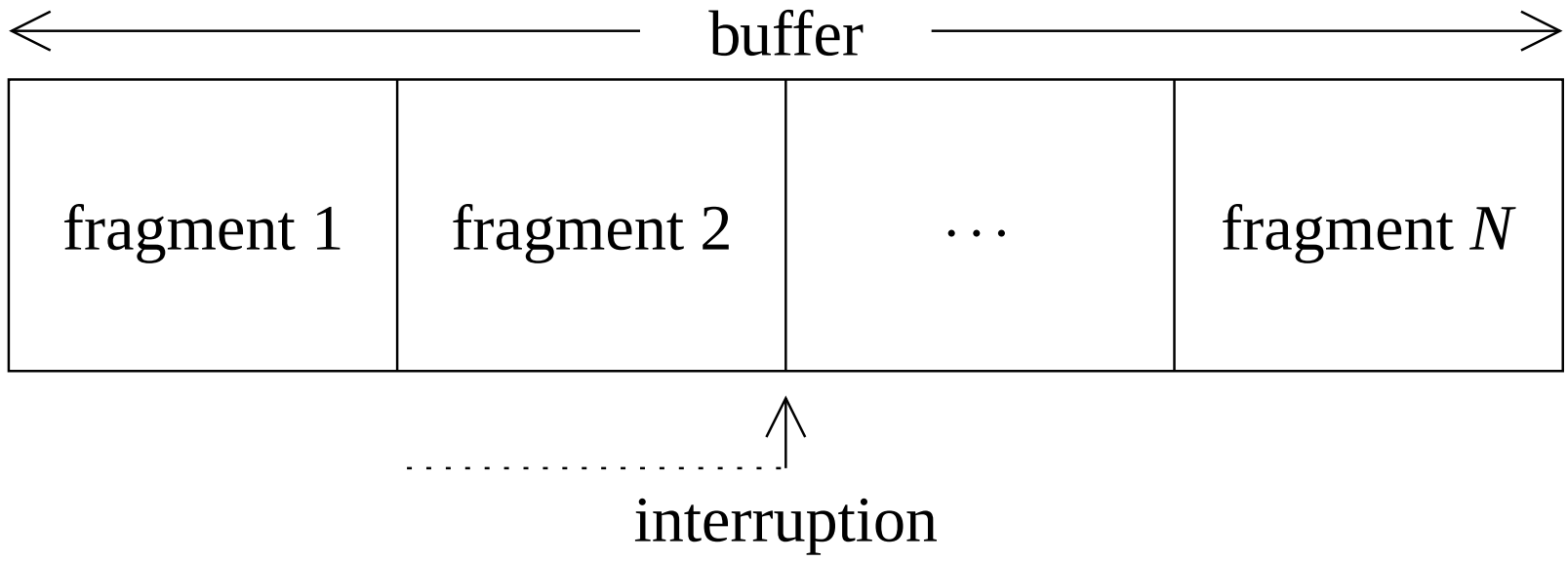
Carte son

- convertisseur numérique $\leftrightarrow$ analogique
- buffer interne d'échantillons
- division en fragments (parcours circulaire)
- mécanisme d'interruption



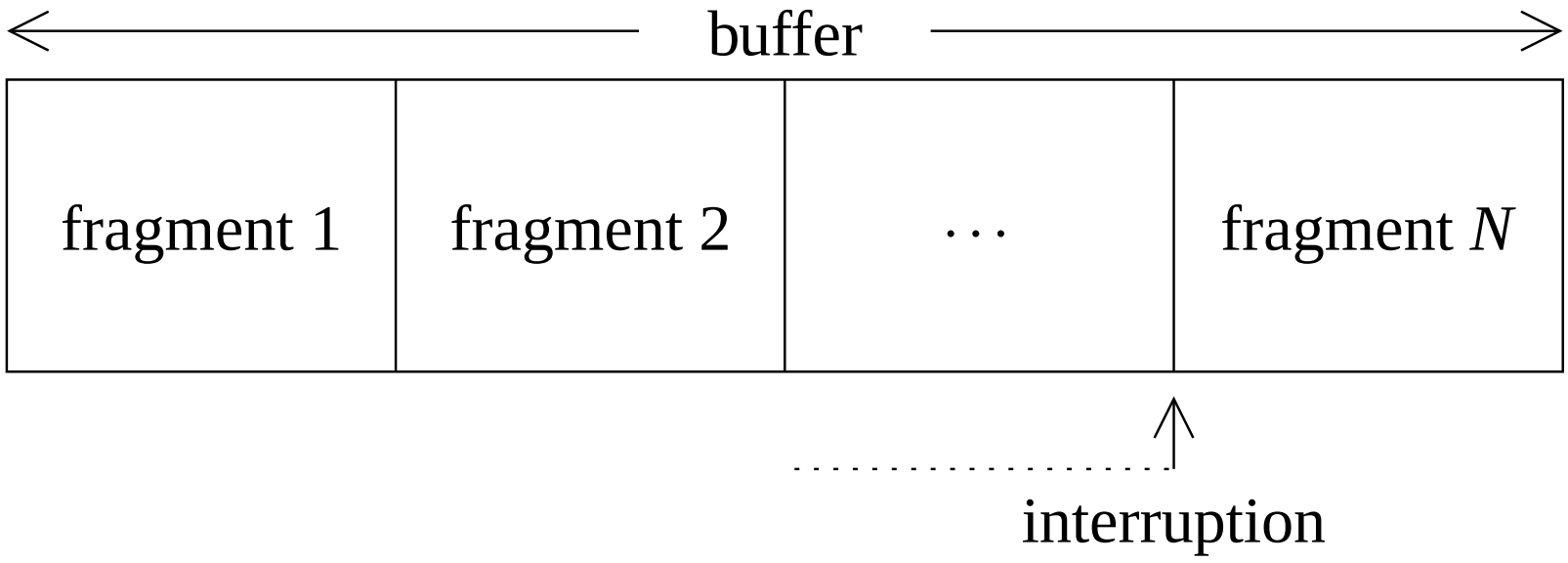
Carte son

- convertisseur numérique $\leftrightarrow$ analogique
- buffer interne d'échantillons
- division en fragments (parcours circulaire)
- mécanisme d'interruption



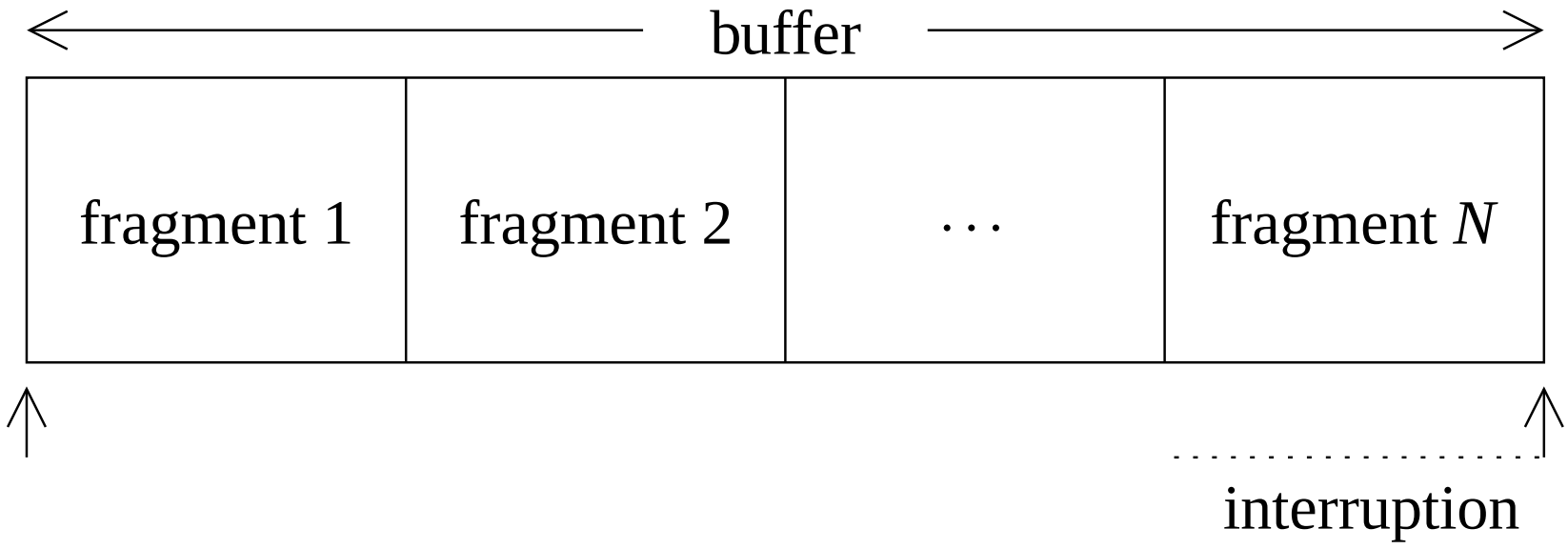
Carte son

- convertisseur numérique $\leftrightarrow$ analogique
- buffer interne d'échantillons
- division en fragments (parcours circulaire)
- mécanisme d'interruption



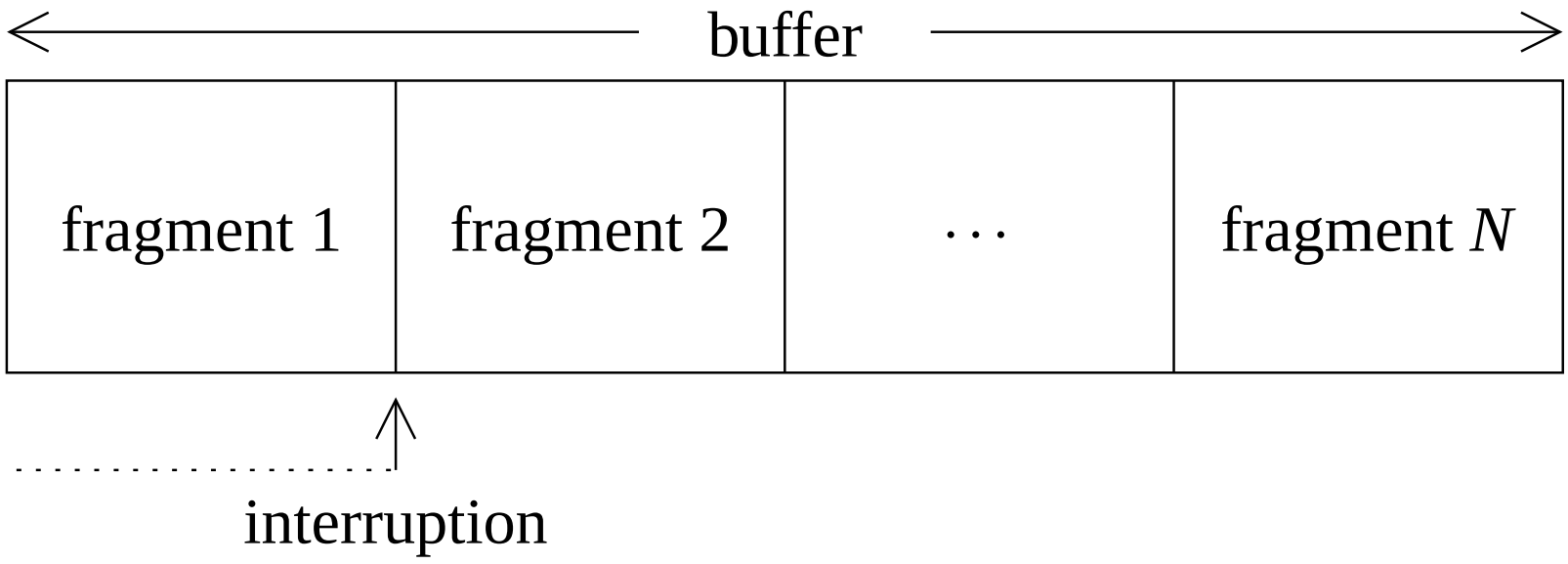
Carte son

- convertisseur numérique $\leftrightarrow$ analogique
- buffer interne d'échantillons
- division en fragments (parcours circulaire)
- mécanisme d'interruption



Carte son

- convertisseur numérique $\leftrightarrow$ analogique
- buffer interne d'échantillons
- division en fragments (parcours circulaire)
- mécanisme d'interruption



Linux : OSS (*Open Sound System*) $\longrightarrow$ ALSA (*Advanced Linux Sound Architecture*)

Réglage des niveaux. . .

mixer

(`gmix`, `kmix`, `alsamixer`, *etc.*)

réglages conseillés :

- volume : **80%** au maximum
- *bass* : 50% (amplification des graves)
- *trebble* : 50% (amplification des aigus)

Device **UNIX : `/dev/dsp`**

UNIX : principe du “tout est fichier”

- fichiers spéciaux `/dev/dsp` et/ou `/dev/audio`
  (géré par le noyau UNIX,
  pas d’existence sur le disque dur)
- commandes UNIX usuelles :
  `open`, `read` / `write`, `close`
- lecture / écriture : le plus vite possible...
  - processus bloqué (endormi)
    quand le buffer est vide / plein
  - pas de véritable temps-réel :
    risque de clics (*overrun* / *underrun*)
    bibliothèque portable *Sequential Audio Library*

Exemple : gestion du fichier spécial

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
...
int audio = open ("/dev/dsp", O_WRONLY);
if (audio == -1)
  {
    perror ("open(audio)");
    exit (EXIT_FAILURE);
  }
  ...
close (audio);
```

Exemple : réglages des paramètres

```
#include <sys/soundcard.h>

int format, channels, speed;

// sample format:
// signed 16-bit (int) little-endian
format = AFMT_S16_LE;
ioctl (audio, SOUND_PCM_SETFMT, &format);

// number of channels (polyphony)
channels = 2;  // stereo
ioctl (audio, SNDCTL_DSP_CHANNELS, &channels);

// sampling rate
speed = 44100;
ioctl (audio, SNDCTL_DSP_SPEED, &speed);
```

- l'ordre des réglages est important
- les paramètres peuvent être modifiés après l'appel
- `ioctl()` peut retourner une erreur…

Exemple : 1 minute de silence...

```
for (i=0; i<(60*speed); i++)
  {
    short samples[2] = { 0, 0 };

    write (audio, samples, 2*sizeof(short));
  }
```

JACK Audio Connection Kit **(JACK)**

- portable : UNIX (Linux), Windows, MacOS X
- basé sur l'appel d'une fonction réflexe *(callback)*
  bibliothèque PortAudio (*Portable Audio*)

JACK : système client-serveur

```
jack_client_t *client = jack_client_new ("play");

 ...

if (jack_activate (client))
  {
    fprintf (stderr, "cannot activate client.\n");
    exit (EXIT_FAILURE);
  }

 ...

jack_client_close (client);
```

valeur NULL en cas d'erreur / impossibilité

JACK : ports de communication

```
// ports de la carte son
const char **physical_out_ports =
 jack_get_ports (client, NULL, NULL,
                 JackPortIsPhysical|JackPortIsInput);

// port du client
jack_port_t *client_output_port =
 jack_port_register (client, "output", JACK_DEFAULT_AUDIO_TYPE,
                     JackPortIsOutput, 0);

// mise en connexion
jack_connect (client, jack_port_name (client_output_port),
              physical_out_ports[0]);
```

valeur de retour : 0 si succès

JACK : fonction réflexe

```
int
process (jack_nframes_t nframes, void *arg)
{
  jack_default_audio_sample_t *out =
   (jack_default_audio_sample_t *)
    jack_port_get_buffer (client_output_port, nframes);

  bzero (out, sizeof (jack_default_audio_sample_t) * nframes);

  return 0;
}

 ...

jack_set_process_callback (client, process, 0);

sleep (60);
```

JACK (fin)

- taux d'échantillonnage imposé par le serveur :

```
jack_get_sample_rate (client);
```

- type des échantillons imposé par JACK :

`jack_default_audio_sample_t` $\longrightarrow$ `float`

JACK (démon)

sous UNIX, le serveur JACK est implanté sous forme de “démon” `jackd` qui se lance ainsi :

```
jackd --help
jackd -d alsa --help
jackd -d alsa -d hw:0 -r 44100 -D
```

dans ce cas, les paramètres sont :

- utiliser ALSA
- carte (*hardware*) numéro 0
- fréquence d’échantillonnage de 44100 Hz
- mode duplex

Matériel au CREMI (salle 202)

- 20 cartes son Creative® SoundBlaster
  - échantillonnage : 44100 Hz (qualité CD, voire DAT : 48000 Hz)
  - quantification : 16 bits
  - polyphonie : 2 E/S
  - convertisseur numérique $\leftrightarrow$ analogique interne
- 20 cartes son M-Audio® MIDIMan Delta 44
  - échantillonnage : 96000 Hz (qualité DVD)
  - quantification : 24 bits
  - polyphonie : 4 E/S
  - convertisseur numérique $\leftrightarrow$ analogique externe
- 10 pré-amplificateurs casques Lunnar® Ω-Quadra
  - 2×2 entrées monophoniques (A et B)
  - 4 sorties stéréophoniques (A ou B)
- 20 casques audio AKG® K240 Studio