# Scheduling

En audio : important de gérer précisément la temporalité des événements !

On a accès à une horloge de qualité : Web Audio Clock

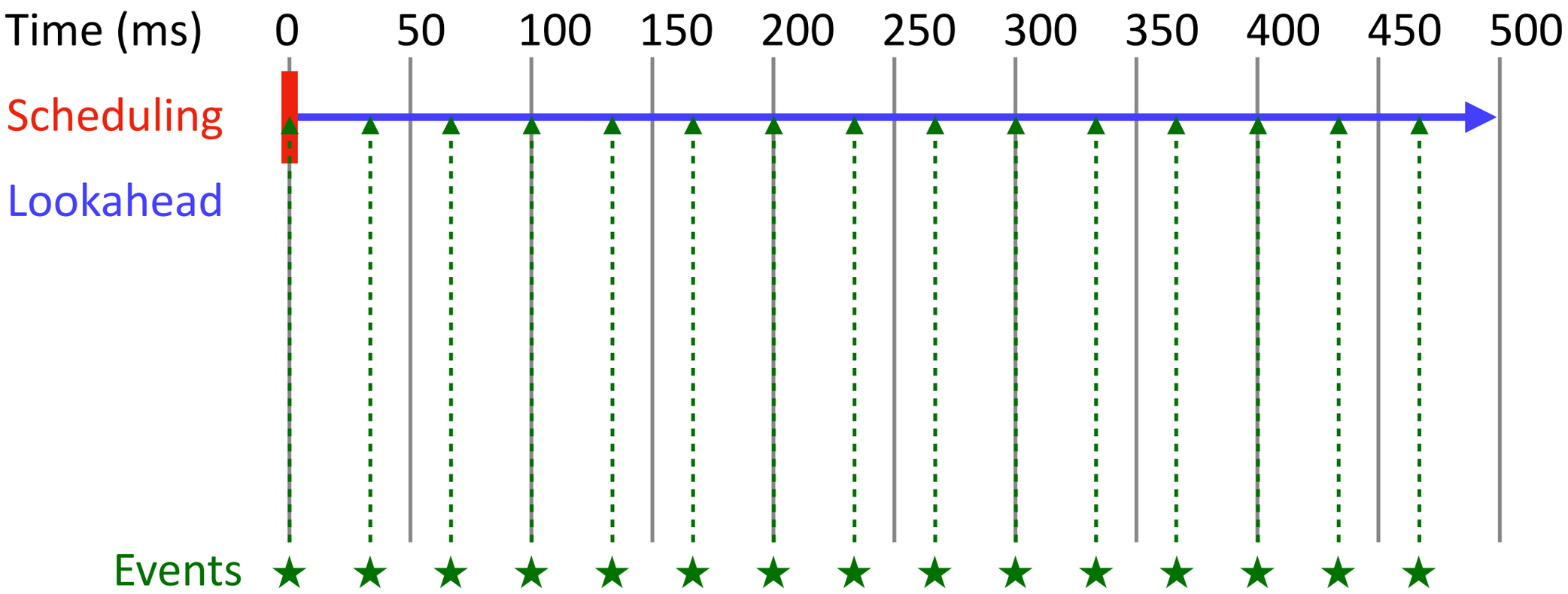
‘*audioContext.currentTime*‘

- Horloge de l’hardware audio
- Précision permettant des opérations au niveau du sample

—> Est utilisée notamment pour les événements :

- src.start(time)
- src.stop(time)
- audioParam.setValueAtTime(value, time)

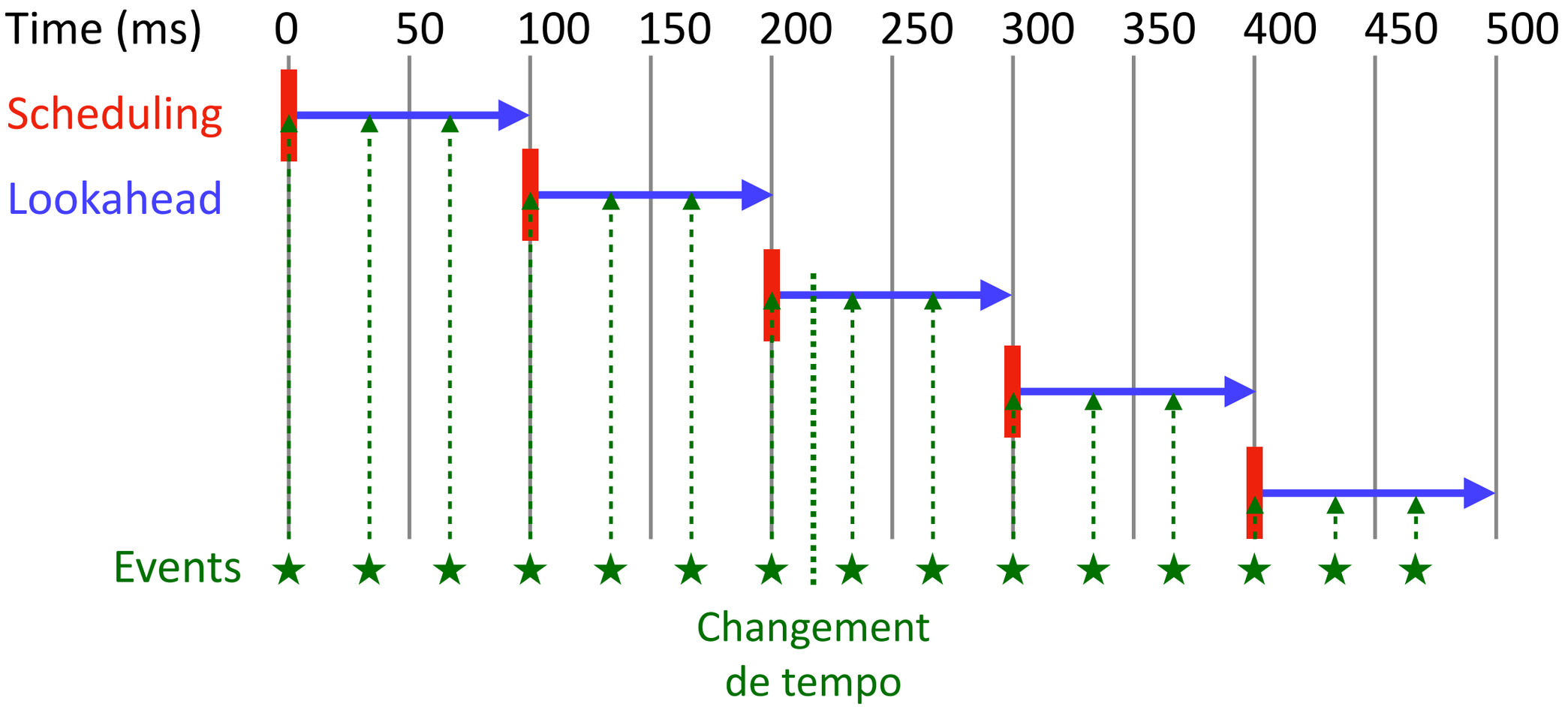
# Scheduling




Possible de planifier précisément des événements audio
MAIS manque de flexibilité

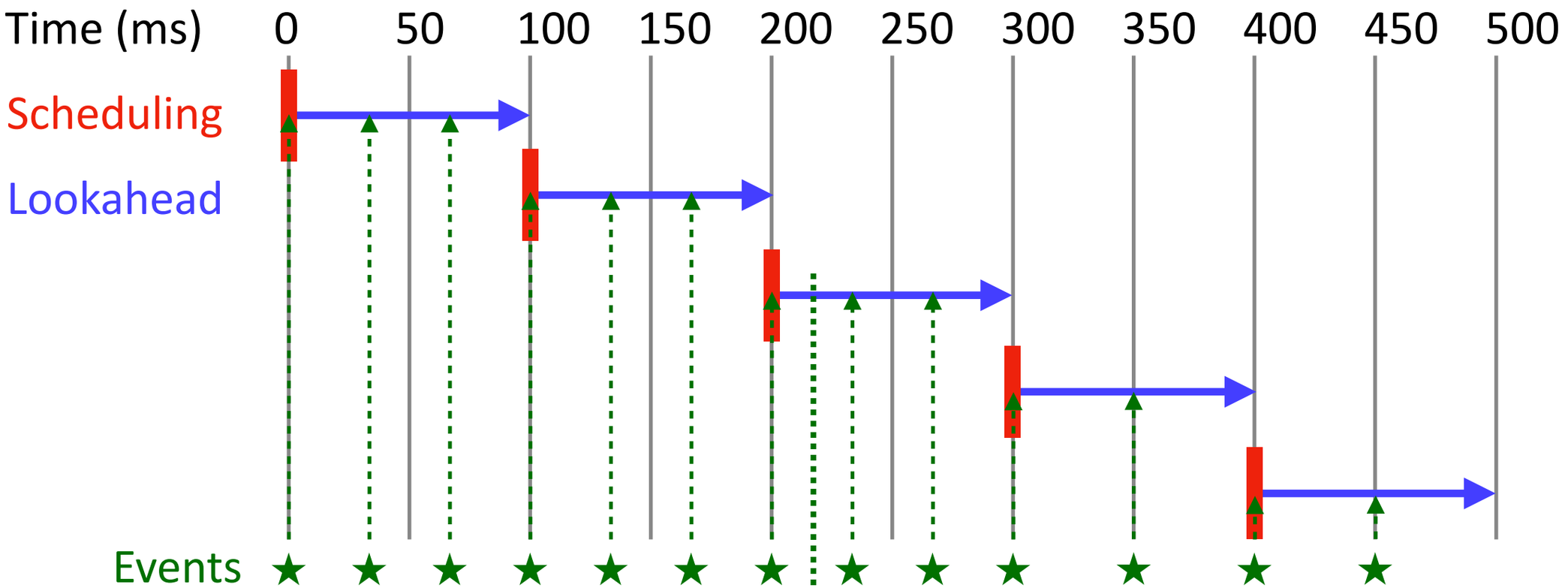
# Scheduling

Idée : Planifier seulement les événements audio à venir dans un petit lapse de temps



# Scheduling

Idée : Planifier seulement les événements audio à venir dans un petit lapse de temps




```
function tick() {
  *program events for the next 100ms using audio clock*
  setTimeout(tick, period * 1000);
```

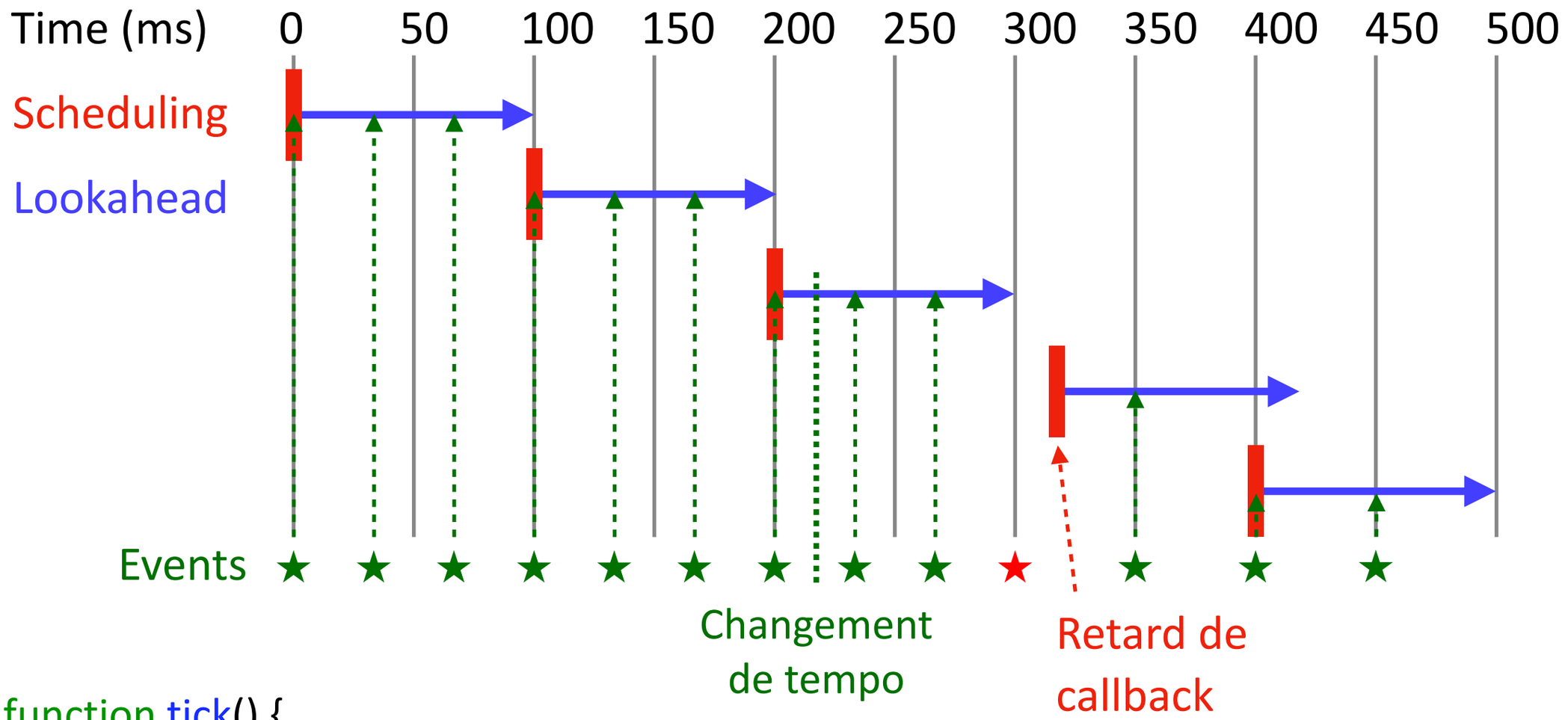
↑

Permet d'appeler une fonction après un certains temps

**Problème :** le callback de cette méthode peut être retardée par d'autres processus qui s'exécutent sur le thread principal —> le temps donné en argument peut ne pas être respecté !

# Scheduling

Idée : Planifier seulement les événements audio à venir dans un petit lapse de temps




```
function tick() {
  *program events for the next 100ms using audio clock*
  setTimeout(tick, period * 1000);
```

↑

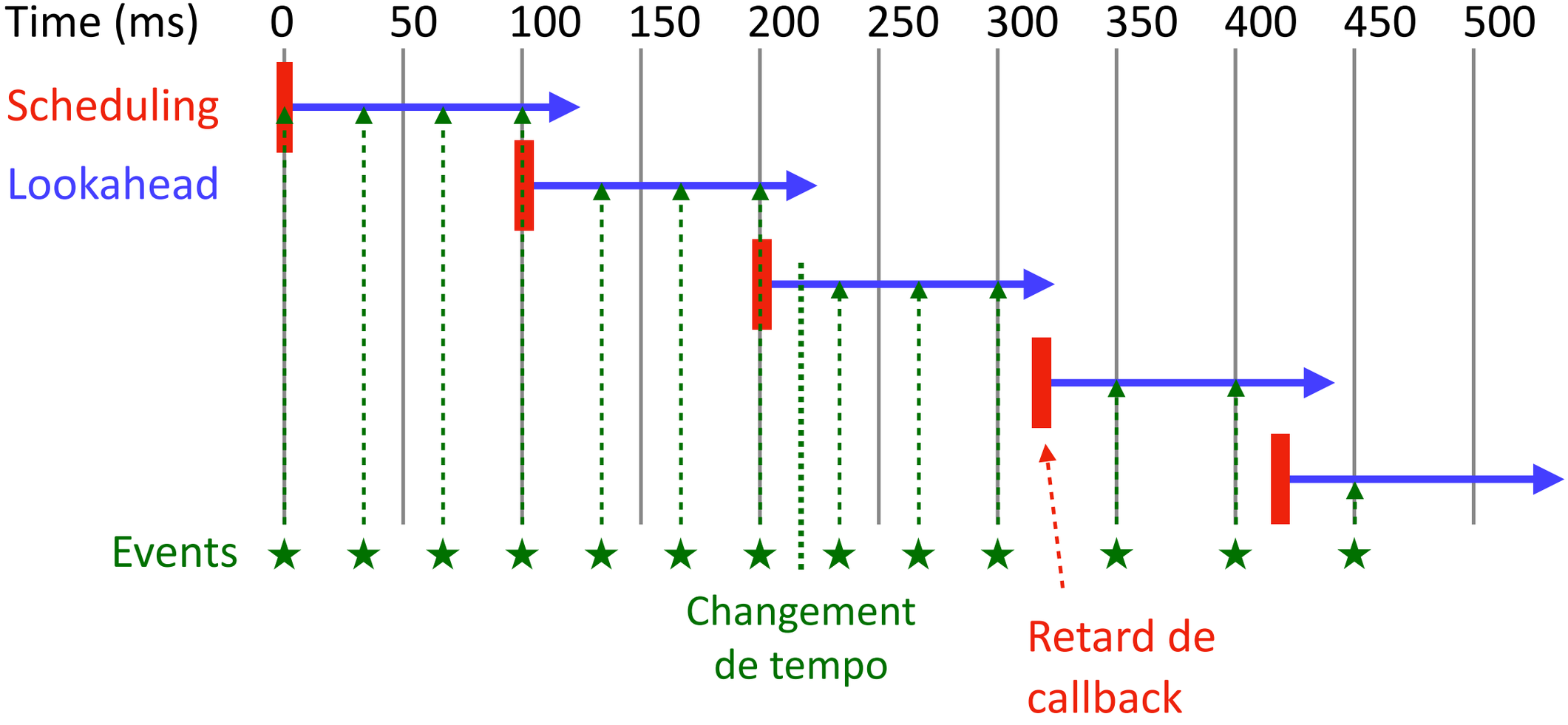
Permet d'appeler une fonction après un certains temps

**Problème :** le callback de cette méthode peut être retardée par d'autres processus qui s'exécutent sur le thread principal —> le temps donné en argument peut ne pas être respecté !

# Scheduling

Solution : avoir un temps de lookahead plus long que la période de scheduling

```
function tick() {
  *program events for the next 125ms using audio clock*
  setTimeout(tick, period * 1000);
```



# Scheduling

**Article détaillé :** https://www.html5rocks.com/en/tutorials/audio/scheduling/