Les Systèmes Temps Réel

Chapitre 3:
Ordonnancement des tâches apériodiques

Cours Master 1 GLSD

2019/2020

Pr. Salim BITAM

Département d'Informatique, Université de Biskra

07000 Biskra, Algérie

Notions de base

- Une tâche périodique: connaissance des instants d'arrivées des requêtes (dates de réveil)
- Un STR composé de tâches périodiques : STR dur (contraintes temporelles strictes)
- Une tâche **apériodique**: on ne connait pas les instants d'arrivées des requêtes (dates de réveil)
- On distingue entre:

- Les tâches apériodiques à contraintes relatives et,
- Les tâches apériodiques à contraintes strictes
- Objectif à atteindre pour l'ordonnancement :

1. TA à contraintes relatives : minimiser le temps de réponse
2. TA à contraintes stricts : maximiser le nombre de tâches acceptées en respectant leurs contraintes

Ordonnancement des tâches apériodiques à contraintes relatives

- On a deux grandes catégories :

1. traitement en arrière-plan
2. traitement par serveurs

Traitement en arrière-plan

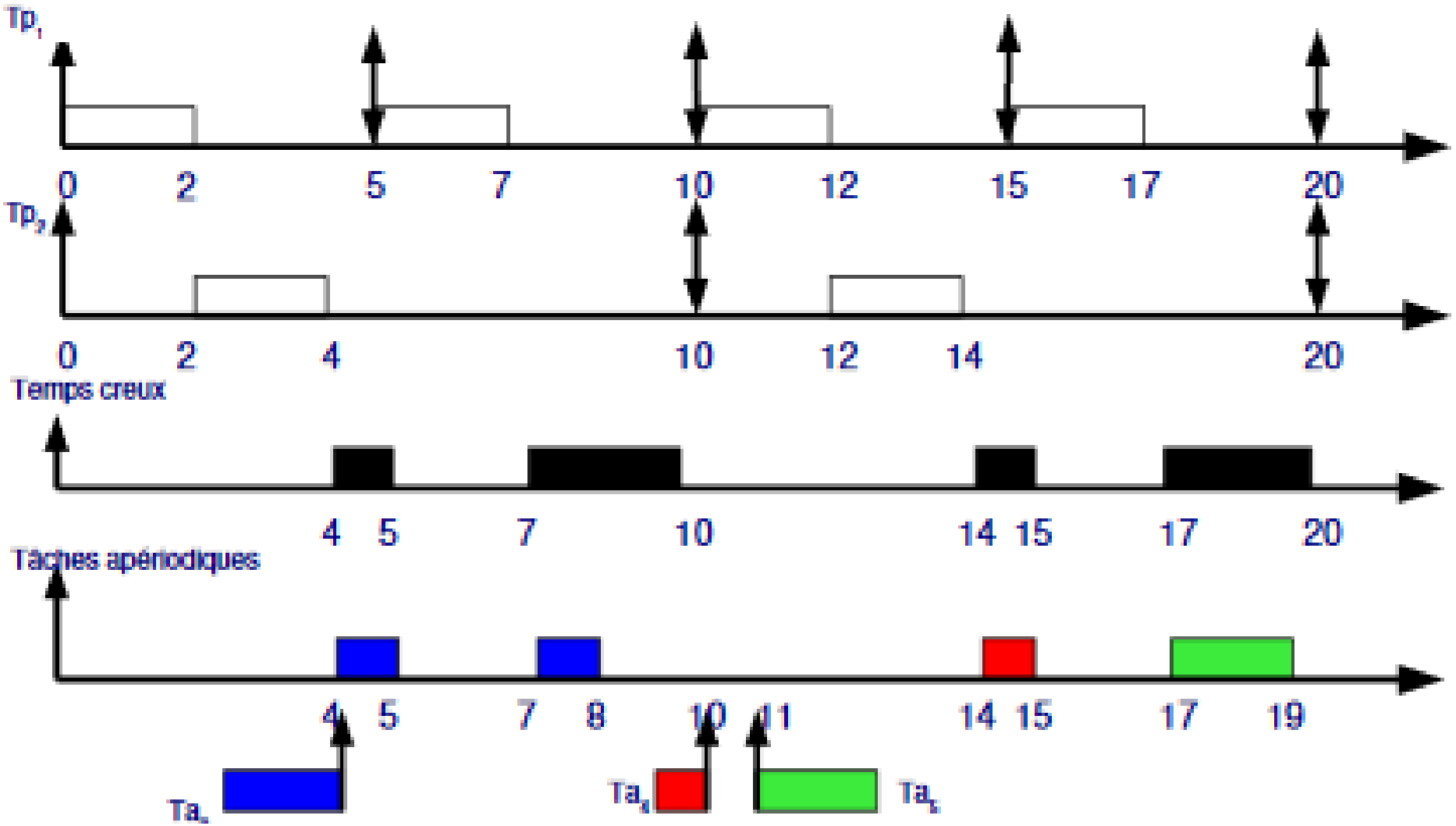
- Principe:
- Les tâches apériodiques sont ordonnancées quand le processeur est oisif
- si plusieurs tâches attendent, elles sont traitées en mode FIFO
- le plus simple, mais le moins performant

Exemple:

- TP1 (r0 = 0, C=2, P=5),
- TP2 (r0 = 0, C=2, P=10),

- TA3 (r = 4, C=2),
- TA4 (r = 10, C=1),
- TA5 (r = 11, C=2)

- On vous propose d'utiliser RMA pour ordonnancer les tâches périodiques (le contexte général)



Traitement en arrière plan

Traitement par Serveur (1)

- Deux approches principales basées sur ce qu'on appelle un serveur
- Définition d'un Serveur:
- un serveur est une tâche périodique (non ordinaire: imaginaire)
- créée pour permettre l'ordonnancement des tâches apériodiques
- Le serveur est caractérisé par
- une période P (choisie par le concepteur a priori)
- Un temps d'exécution C (choisie par le concepteur a priori)
- Un premier réveil (généralement r0 = 0)

Traitement par Serveur (2)

- Le serveur est ordonnancé suivant le même algorithme que les autres tâches périodiques
- une fois actif (possède la priorité), le serveur sert les tâches apériodiques durant son temps d'exécution (C) -dans la limite de sa capacité-.
- NB. L'ordre de traitement des tâches apériodiques ne dépend pas de l'algorithme général, il dépend de l'approche de traitement par serveur
- On cite comme approches de traitement par serveur:
- Serveur par scrutation
- Serveur sporadique

Serveurs par scrutation (1)

- Principe:
- Ordonnancement des tâches apériodiques par serveur dont le principe est:
- à chaque activation du serveur (possession de priorité selon l'algorithme de contexte général –utilisé pour ordonnancer les tâches périodiques-), le serveur sert les tâches apériodiques jusqu'à épuisement de la capacité (C) ou jusqu'à ce qu'il n'y ait plus de tâches en attente

Serveurs par scrutation (2)

- si aucune tâche n'est en attente (à l'activation ou parce que
- la dernière tâche a été traitée), le serveur se suspend

immédiatement :

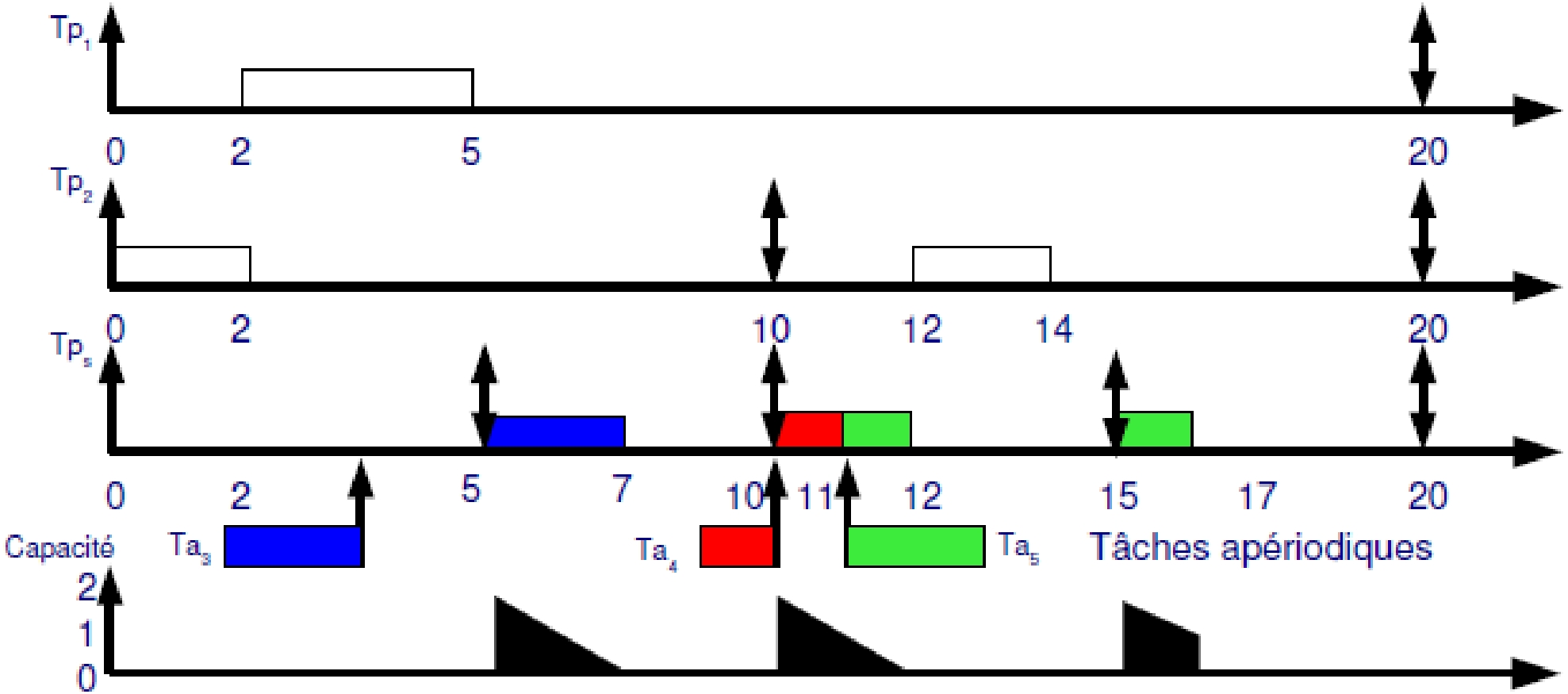
1. Il perd le processeur (donc les tâches périodiques peuvent reprendre leurs exécutions)
2. Il perd sa capacité (donc si après un instant de temps ou plus dont un tâche apériodiques se présente, cette dernière ne peut pas être exécutée, elle doit attendre une prochaine activation du serveur)

Exemple:

- TP1 (r0 = 0, C=3, P=20),
- TP2 (r0 = 0, C=2, P=10),

- TS (r0 = 0, C=2, P=5),

- TA3 (r = 4, C=2),
- TA4 (r = 10, C=1),
- TA5 (r = 11, C=2)
- On vous propose d'utiliser RMA pour ordonnancer les tâches périodiques (le contexte général)



Traitement par serveur de scrutation

Limitations de S/ Scrutation

- Perte de capacité si aucune tâche apériodique n'est présente à l'instant d'activation
- Alors, si une tâche apériodique arrive après l'instant d'activation, elle doit attendre la prochaine activation du serveur

Serveur sporadique (1)

- Principe:
- Traitement par serveur tel que le serveur par scrutation sauf que:
- si aucune tâche n'est en attente (à l'activation ou parce que
- la dernière tâche a été traitée), le serveur se suspend

immédiatement :

1. Il perd le processeur (donc les tâches périodiques peuvent reprendre leurs exécutions)
2. Il ne perd pas sa capacité (donc si après un instant de temps ou plus dont un tâche apériodiques se présente, cette dernière **est exécutée immédiatement** (elle ne doit pas attendre une prochaine activation du serveur)

Serveur sporadique (2)

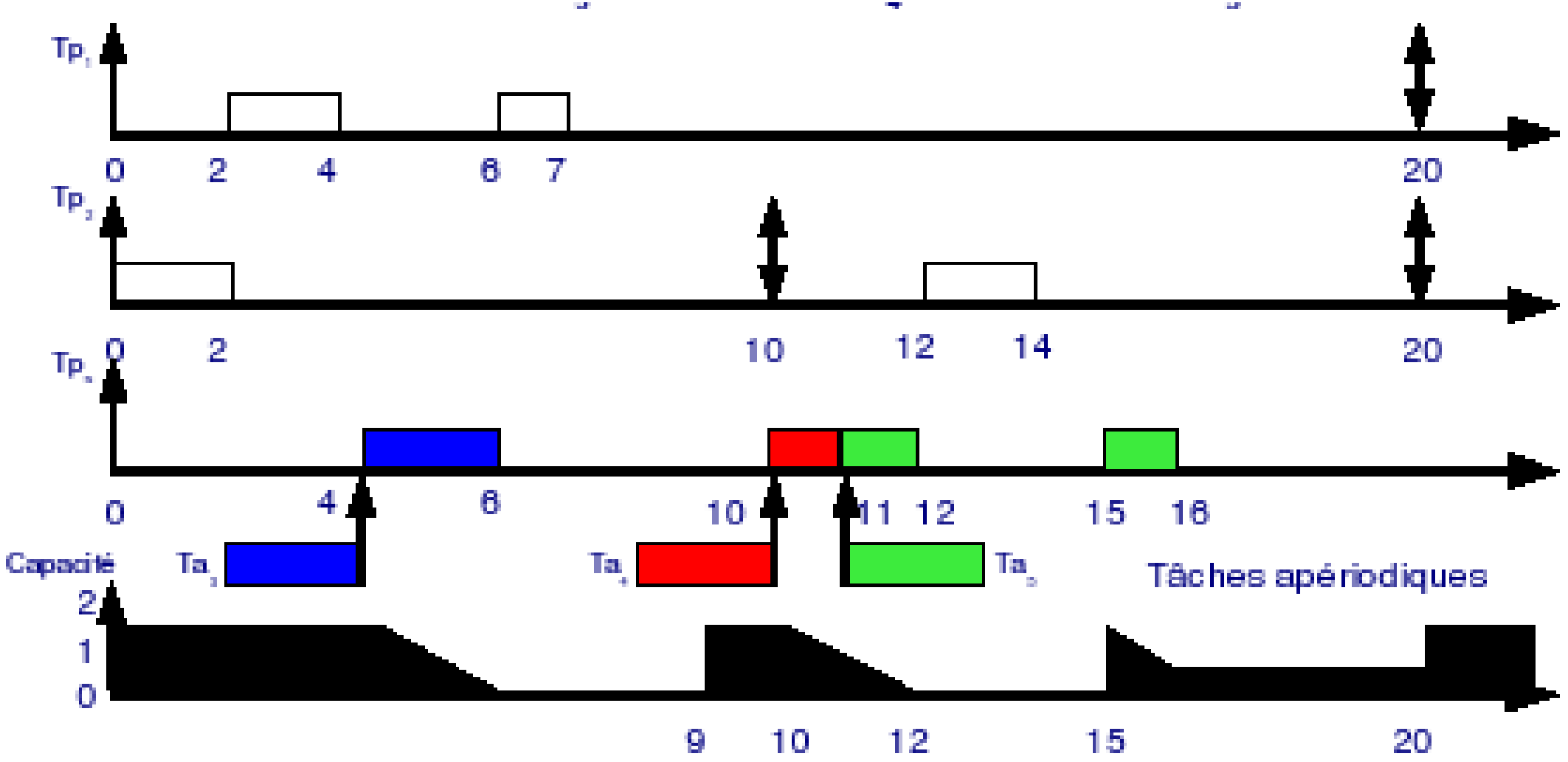
- La capacité est conservée si elle n’est pas épuisée et donc peut être utilisée même après la perte du processeur
- S/sporadique améliore le temps de réponse des tâches apériodiques sans diminuer le taux d'utilisation du processeur pour les tâches périodiques

Exemple:

- TP1 (r0 = 0, C=3, P=20),
- TP2 (r0 = 0, C=2, P=10),

- TS (r0 = 0, C=2, P=5),

- TA3 (r = 4, C=2),
- TA4 (r = 10, C=1),
- TA5 (r = 11, C=2)
- On vous propose d'utiliser RMA pour ordonnancer les tâches périodiques (le contexte général)


Tp1
0
2
4
6
7
20
Tp2
0
2
10
12
14
20
Tp3
0
4
6
10
11
12
15
16
Capacité
Ta1
Ta2
Ta3
Tâches apériodiques
2
1
0
9
10
12
15
20

Ordonnancement des tâches apériodiques à contraintes strictes (1)

- Méthode de test de garantie:
- Principe: ordonnancer les tâches en EDF.
- A chaque nouvelle tâche apériodique, faire exécuter un « test de garantie » pour:
- vérifier que toutes les contraintes temporelles seront
- Respectées de toutes les tâches périodiques et apériodiques.
- Si oui : la nouvelle tâche apériodique est acceptée
- Si non, on refuse la nouvelle tâche apériodique.

Ordonnancement des tâches apériodiques à contraintes strictes (2)

- NB. On favorise toujours les tâches périodiques

- On distingue deux approches:

1. acceptation dans les temps creux
2. acceptation des tâches apériodiques et ordonnancement

- conjoint

Acceptation dans les temps creux

- Principe:
- ordonnancement des tâches périodiques par EDF
- les tâches apériodiques acceptées sont ordonnancées dans les temps creux des tâches périodiques si:

A la présence de chaque tâche apériodique, on exécute le test de garantie :

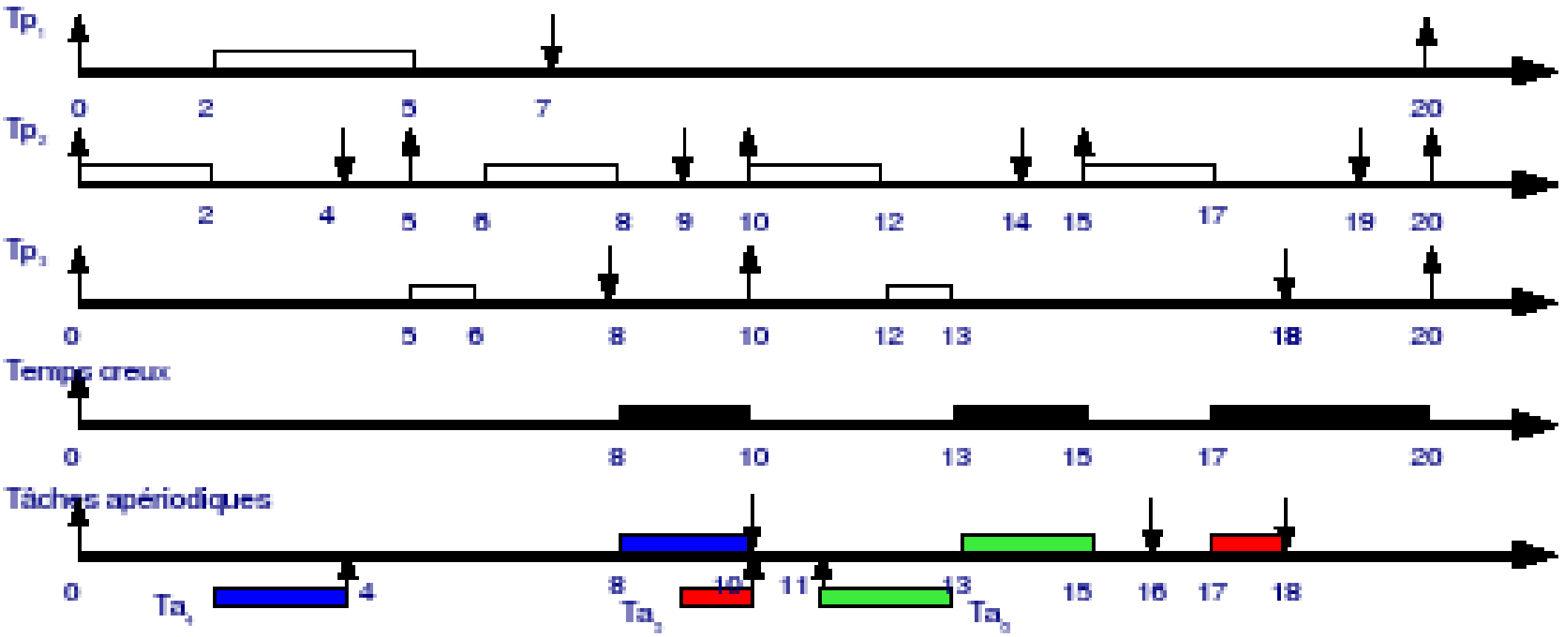
Si le test est vérifié alors:

- Tester l'existence d'un temps creux suffisant
- Si OK, la tâche est acceptée sinon elle est rejetée

Acceptation dans les temps creux: exemple

- Tp1 (r0=0, C=3, D=7, P=20),
- Tp2 (r0=0, C=2, D=4, P=5),
- Tp3 (r0=0, C=1, D=8, P=10)

- Ta4 (r=4, C=2, d=10),
- Ta5 (r=10, C=1, d=18),
- Ta6 (r=11, C=2, d=16)


Tp1
0
2
5
7
20
Tp2
2
4
5
6
8
9
10
12
14
15
17
19
20
Tp3
0
5
6
8
10
12
13
18
20
Temps creux
0
8
10
13
15
17
20
Tâches apériodiques
0
4
8
10
11
13
15
16
17
18
Ta1
Ta2
Ta3