# Formation Automatique et Informatique Industrielle

Master 1 S2

Matière : Systèmes Embarqués et Systèmes Temps Réel SE-STR

Par : ATOUI Hamza

# Plan du cours

- Politiques d'ordonnancement:
  - Partie 1 (Tâches périodiques indépendantes à contrainte stricte):
    - **Rate Monotonic Scheduling (RM).**
    - **Deadline Monotonic Scheduling (DM).**
    - **Earliest Deadline First Scheduling (EDF).**
    - **Least Laxity First Scheduling(Least slack time scheduling -LST-) (LLF).**

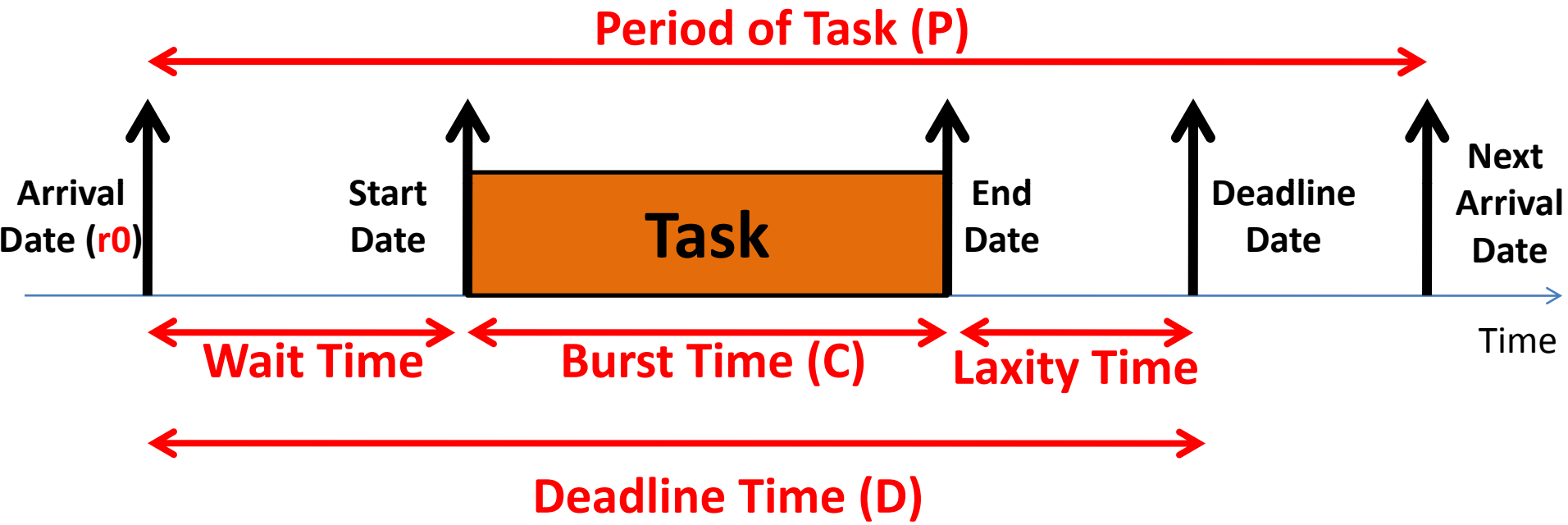
# Tâches périodiques indépendantes

- Hypothèses et modèle :
  - Nombre de CPU : 1.
  - Classe de système : Hard/Soft.
  - Modèle : Task(r0, C, D, P).
  - Priorité : Statique/Dynamique.
  - Le temps d'allocation du CPU au tâche choisi : Préemptif.
  - Temps de charge du noyau : Négligeable.

# Le TICK d'un système

- C'est le battement du cœur du système, dont certain services du noyau OS/RTOS utilisent pour la synchronisation et le calcul du temps.
- Sur l'axe matérielle, le TICK système se génère par un TIMER à intervalle régulier sous interruption.
- D'une façon générale la valeur de TICK est le GCD($C_1$, $C_2$, $C_3$, …) (Greatest Common Divisor ) avec $C_i$ le Burst Time de la tâche « i » .

# Modèle de tâche



Politiques d’ordonnancement

# RATE MONOTONIC SCHEDULING (RM)

# Rate Monotonic Scheduling (RM)

- Rate Monotonic fait partie de la famille des algorithmes à **priorité fixe**.
- Le principe consiste à **affecter** aux tâches des **priorités** qui sont **inversement proportionnelles à leurs périodes**.
- Les tâches à ordonnancer doivent être à échéance sur requête Task(r0, C, D=P).

# Rate Monotonic Scheduling (RM)

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par RM.
- **Etape 1** : Test de faisabilité/ordonnancement.
- Si l'étape 1 est valide on passe vers les étapes ci-après.
- **Etape 2** : Calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(P1, P2,…).
- **Etape 3** : Détermination de priorité des différentes tâches par : on va associer la plus haute **priorité** à la tâche de la **petite période**.
- **Etape 4** : de tracer les chronogrammes par:
  - De signaler sur long de chronogramme de chaque tâche : {r↑, P↕}.
  - D'ordonnancer les tâches selon l'algorithme dans le diapo suivant.
  - De tracer le chrono **Gantt Chart (GC)**.

# Rate Monotonic Scheduling (RM)

- Algorithme d'ordonnancement par RM :

**À chaque TICK_SYSTEM faire**

**1- Chercher les tâches à l'état READY (chargement de ReadyQueue)**

**2- Sélectionner la tâche de la plus petite période dans le ReadyQueue.**

**3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.**

**4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.**

**Répéter les 4 étapes jusqu'à la fin de l'intervalle de simulation**

# Rate Monotonic Scheduling (RM)

- **Exemple** : soit le système temps réel a trois tâches périodiques suivantes :

| Task | r0 | C | D=P |
|---|---|---|---|
| **Task1** | **0** | **3** | **20** |
| **Task2** | **0** | **2** | **5** |
| **Task3** | **0** | **2** | **10** |

- Ordonnancer ces tâches par RM.

# Rate Monotonic Scheduling (RM)

- **Etape 1** : Test de faisabilité/ordonnancement

$$U = CH = \sum_{i=1}^{N} C_i / P_i \qquad \text{(Facteur d'utilisation/charge)}$$

$$U \leq 1 \qquad \text{(T.F)} \qquad \text{(condition nécessaire)}$$

$$U \leq N(2^{\frac{1}{N}} - 1)\ \text{(T.O)} \qquad \text{(condition suffisante)}$$

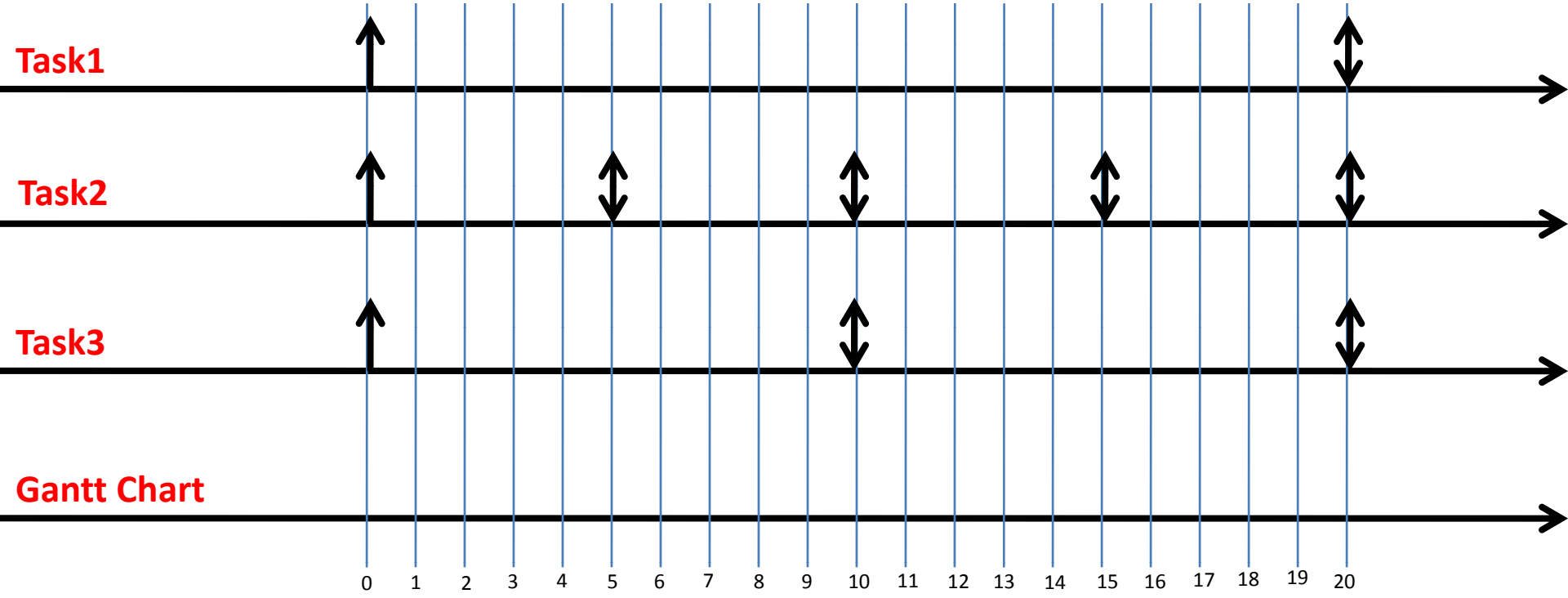
- U = 3/20 + 2/5 + 2/10 = 0.75 ≤ $3(2^{1/3}-1)$ = 0.78 ≤ 1

# Rate Monotonic Scheduling (RM)

- **Etape 2** : calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(20, 5, 10).
- HyperPeriod = 20.
- Donc, l'intervalle d'étude est [0, 20].
- **Etape 3** : Détermination de priorité.
- Task1 a une période de 20
- Task2 a une période de 5
- Task3 a une période de 10
- Donc, Task2 prioritaire que Task3 et Task3 prioritaire que Task1 (Task2 > Task3 > Task1).

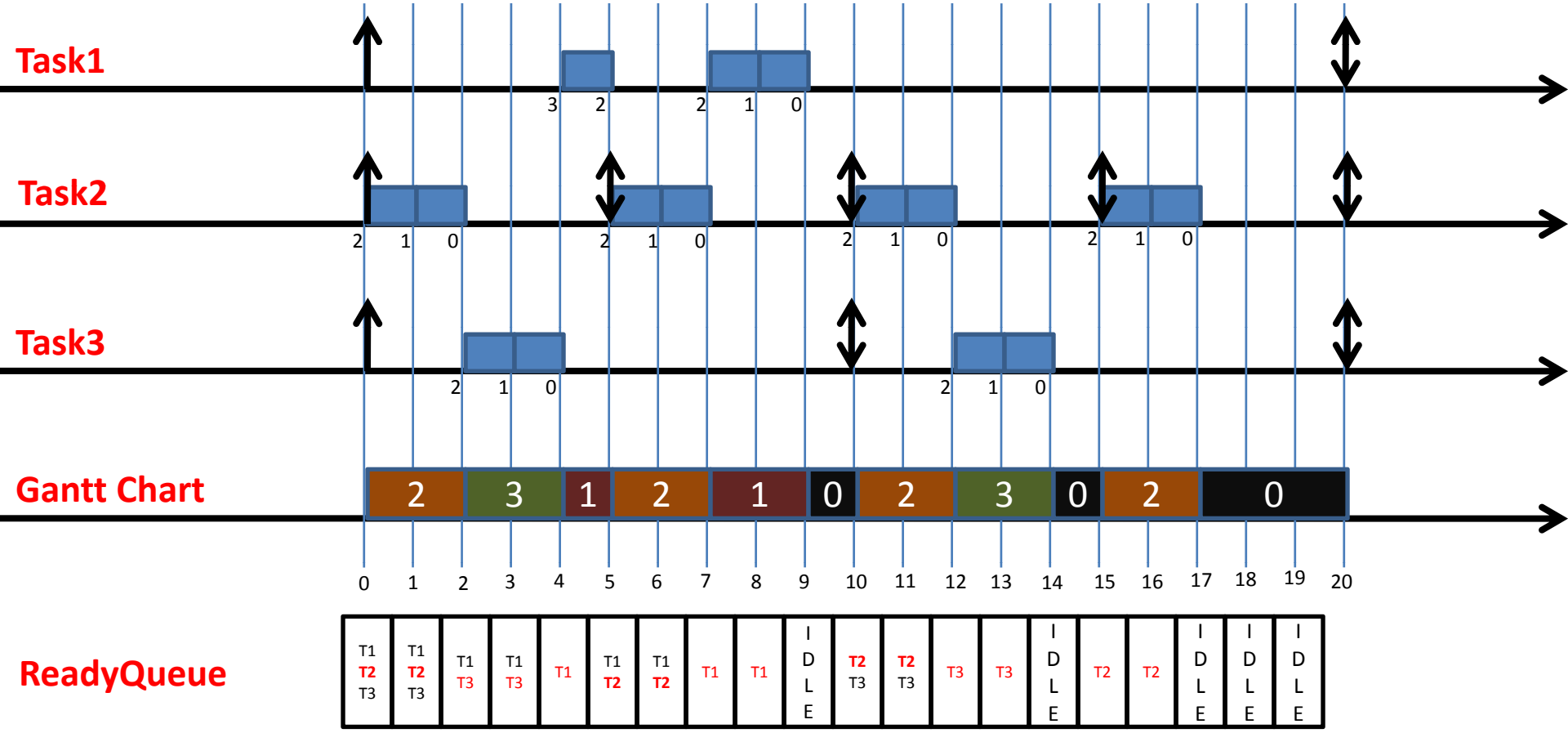
# Rate Monotonic Scheduling (RM)

- **Etape 4** : de tracer les chronogrammes.



# Rate Monotonic Scheduling (RM)

- **Etape 4** : de tracer les chronogrammes.



Politiques d’ordonnancement

# DEADLINE MONOTONIC SCHEDULING (DM)

# Deadline Monotonic Scheduling (DM)

- Deadline Monotonic est un algorithme à **priorité fixe** proche de Rate Monotonic à ceci près qu'il **affecte** à une tâche une **priorité inversement proportionnelle au délai critique**.
- Modèle de tâche : Task(r0, C, D, P) avec $D \leq P$.

# Deadline Monotonic Scheduling (DM)

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par DM.
- **Etape 1** : Test de faisabilité/ordonnancement.
- Si l'étape 1 est valide on passe vers les étapes ci-après.
- **Etape 2** : Calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(P1, P2,…).
- **Etape 3** : Détermination de priorité des différentes tâches par : on va associer la plus haute **priorité** à la tâche du **petit délai critique**.
- **Etape 4** : de tracer les chronogrammes par:
  - De signaler sur long de chronogramme de chaque tâche : {r↑, D↓, P↑}
  - D'ordonnancer les tâches selon l'algorithme dans le diapo suivant.
  - De tracer le chrono **Gantt Chart (GC)**.

# Deadline Monotonic Scheduling (DM)

- Algorithme d'ordonnancement par DM :

**À chaque TICK_SYSTEM faire**

**1- Chercher les tâches à l'état READY (chargement de ReadyQueue)**

**2- Sélectionner la tâche du délai critique le plus petit dans le ReadyQueue.**

**3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.**

**4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.**

**Répéter les 4 étapes jusqu'à la fin de l'intervalle de simulation**

# Deadline Monotonic Scheduling (DM)

- **Exemple** : soit le système temps réel a trois tâches périodiques suivantes :

| Task | r0 | C | D | P |
|---|---|---|---|---|
| **Task1** | **0** | **3** | **7** | **20** |
| **Task2** | **0** | **2** | **4** | **5** |
| **Task3** | **0** | **2** | **9** | **10** |

- Ordonnancer ces tâches par DM.

# Deadline Monotonic Scheduling (DM)

- **Etape 1** : Test de faisabilité/ordonnancement

$$U = \sum_{i=1}^{N} C_i / P_i \qquad \text{(Facteur d'utilisation)}$$

$$CH = \sum_{i=1}^{N} C_i / D_i \qquad \text{(Facteur de charge)}$$

$$U \leq 1 \quad \text{(T.F)} \qquad \text{(condition nécessaire)}$$

$$CH \leq N(2^{\frac{1}{N}} - 1) \ \text{(T.O)} \qquad \text{(condition suffisante)}$$

- U = 3/20 + 2/5 + 2/10 = 0.75 ≤ 1
- CH = 3/7 + 2/4 + 2/9 = 1.15 ≥ $3(2^{1/3}-1) = 0.78$

# Deadline Monotonic Scheduling (DM)

- Test de faisabilité par  DM satisfait.
- Test d'ordonnancement par DM non satisfait.

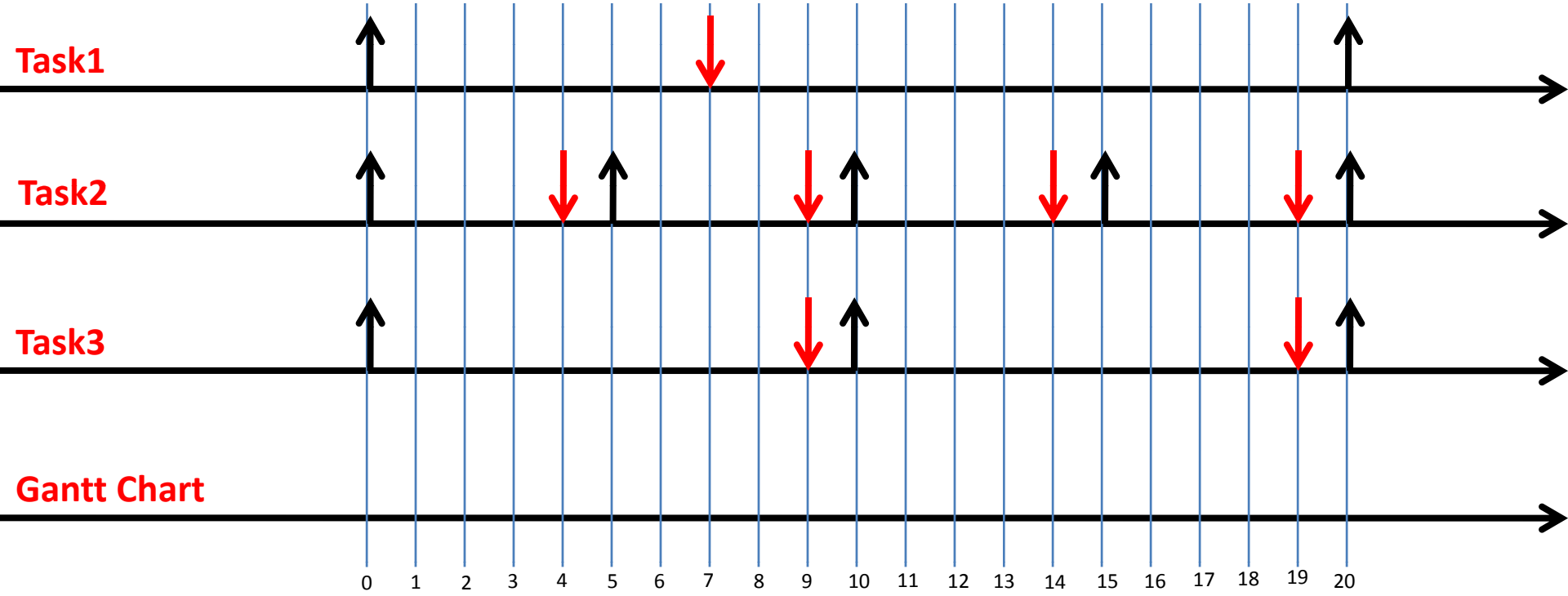
- Malgré le test d’ordonnancement non satisfait, en continuant l’étude.

# Deadline Monotonic Scheduling (DM)

- **Etape 2** : calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(20, 5, 10).
- HyperPeriod = 20.
- Donc, l'intervalle d'étude est [0, 20].
- **Etape 3** : Détermination de priorité.
- Task1 a un délai critique de 7
- Task2 a un délai critique de 4
- Task3 a un délai critique de 9
- Donc, Task2 prioritaire que Task1 et Task1 prioritaire que Task3 (Task2 > Task1 > Task3).

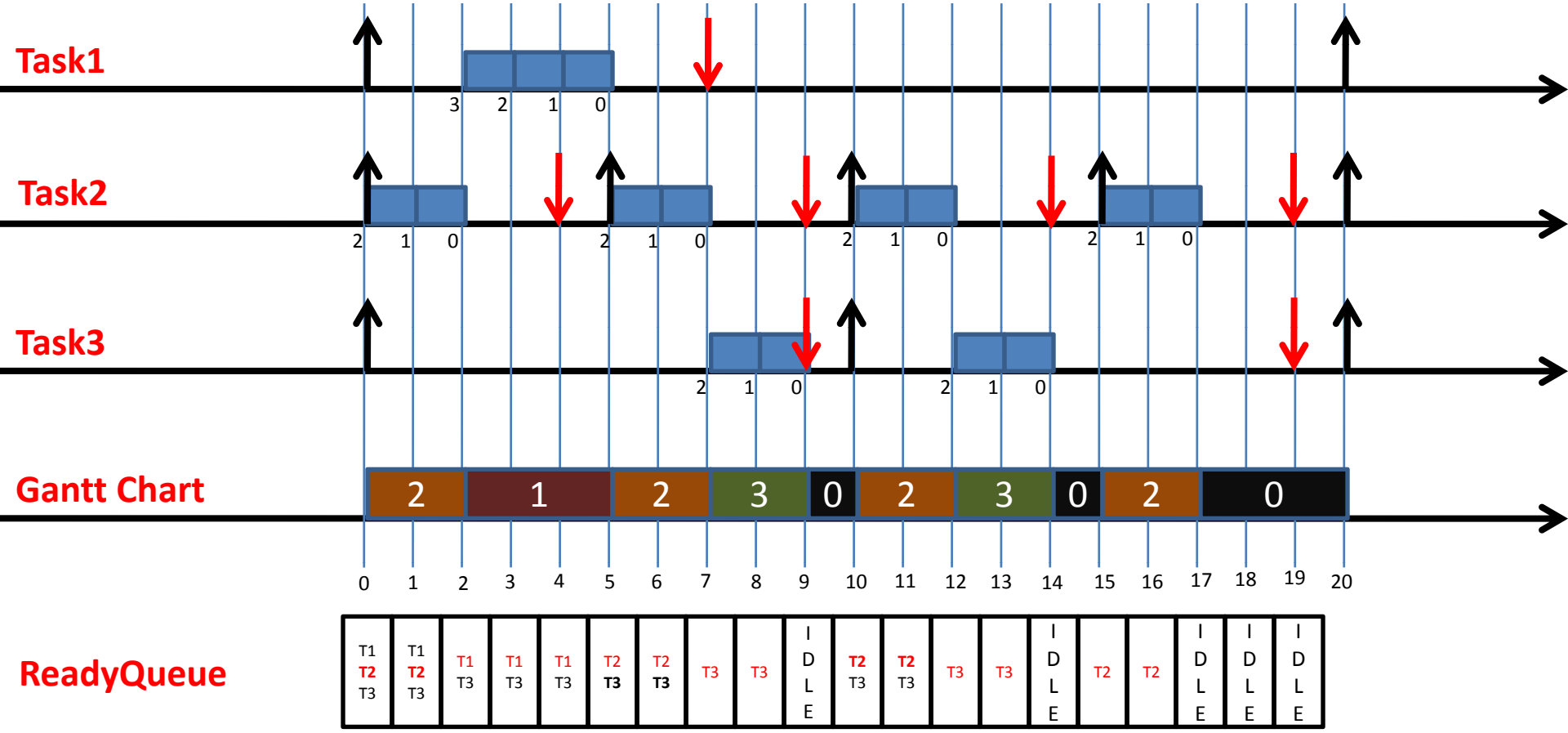
# Deadline Monotonic Scheduling (DM)

- **Etape 4** : de tracer les chronogrammes.



# Deadline Monotonic Scheduling (DM)

- **Etape 4** : de tracer les chronogrammes.



# Deadline Monotonic Scheduling (DM)

- La simulation nous montre que sur la période d'étude, les tâches sont ordonnançables !!!!!

Politiques d’ordonnancement

# EARLIEST DEADLINE FIRST SCHEDULING (EDF)

# Earliest Deadline First Scheduling (EDF)

- Earliest Deadline First est un ordonnancement à **priorité dynamique**.
- Une tâche est d'autant **plus prioritaire** que sa **date d'échéance absolue est proche de la date courante**.
- L'ordre d'exécution dépend de l'**urgence.**
- Modèle de tâche : Task(r0, C, D, P) avec $D \leq P$.

# Earliest Deadline First Scheduling (EDF)

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par EDF.
- **Etape 1** : Test de faisabilité/ordonnancement.
- Si l'étape 1 est valide on passe vers les étapes ci-après.
- **Etape 2** : Calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(P1, P2,…).
- **Etape 3** : de tracer les chronogrammes par:
  - De signaler sur long de chronogramme de chaque tâche : {r↑, D↓, P↑}
  - D'ordonnancer les tâches selon l'algorithme dans le diapo suivant.
  - De tracer le chrono **Gantt Chart (GC)**.

# Earliest Deadline First Scheduling (EDF)

- Algorithme d'ordonnancement par EDF :

**À chaque TICK_SYSTEM faire**

**1- Chercher les tâches à l'état READY (chargement de ReadyQueue)**

**2- Sélectionner la tâche de l'échéance la plus proche par rapport à la date courante dans le ReadyQueue.**

**3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.**

**4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.**

**Répéter les 4 étapes jusqu'à la fin de l'intervalle de simulation**

# Earliest Deadline First Scheduling (EDF)

- **Exemple** : soit le système temps réel a trois tâches périodiques suivantes :

| Task | r0 | C | D | P |
|---|---|---|---|---|
| **Task1** | **0** | **3** | **7** | **20** |
| **Task2** | **0** | **2** | **4** | **5** |
| **Task3** | **0** | **2** | **8** | **10** |

- Ordonnancer ces tâches par EDF.

# Earliest Deadline First Scheduling (EDF)

- **Etape 1** : Test de faisabilité/ordonnancement

$$U = \sum_{i=1}^{N} C_i / P_i \qquad \text{(Facteur d'utilisation)}$$

$$CH = \sum_{i=1}^{N} C_i / D_i \qquad \text{(Facteur de charge)}$$

$$U \leq 1 \quad \text{(T.F)} \qquad \text{(condition nécessaire)}$$

$$CH \leq N(2^{\frac{1}{N}} - 1) \ \text{(T.O)} \qquad \text{(condition suffisante)}$$

- U = 3/20 + 2/5 + 2/10 = 0.75 ≤ 1
- CH = 3/7 + 2/4 + 2/8 = 1.18 ≥ $3(2^{1/3}-1) = 0.78$

# Earliest Deadline First Scheduling (EDF)

- Test de faisabilité par  EDF satisfait.
- Test d'ordonnancement par EDF non satisfait.

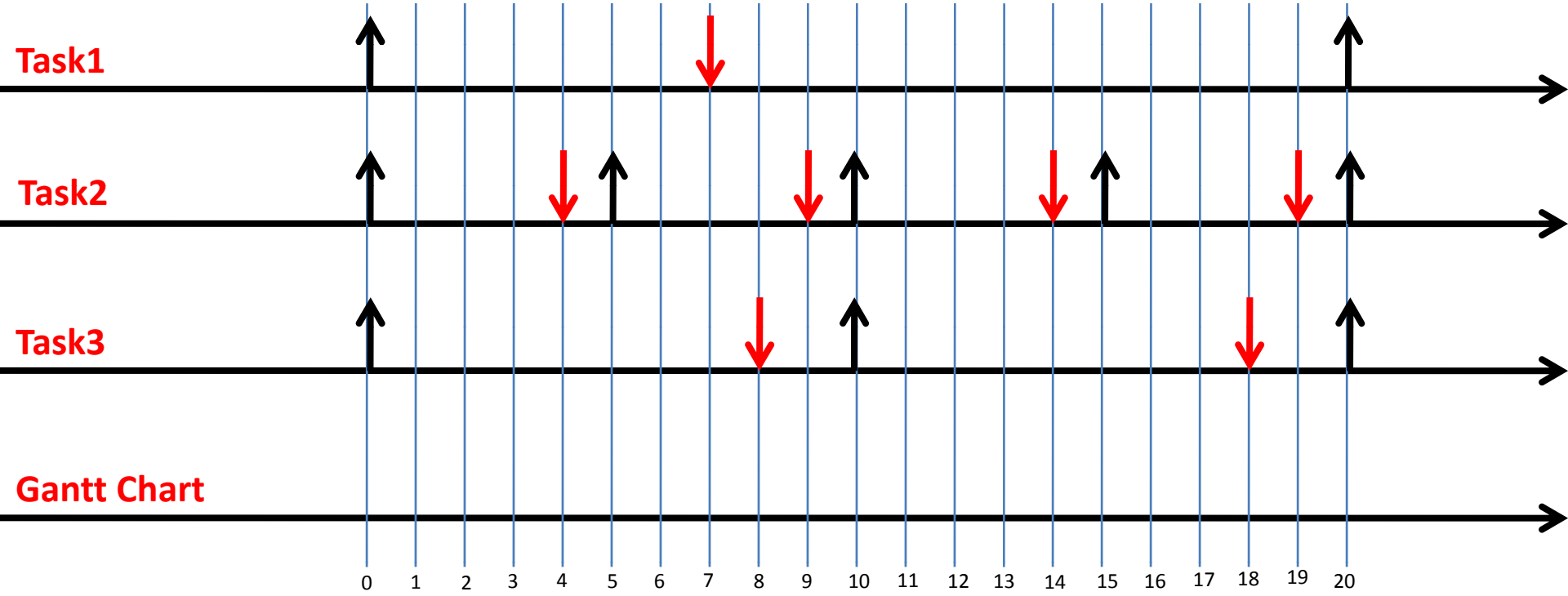
- Malgré le test d’ordonnancement non satisfait, en continuant l’étude.

# Earliest Deadline First Scheduling (EDF)

- **Etape 2** : calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(20, 5, 10).
- HyperPeriod = 20.
- Donc, l'intervalle d'étude est [0, 20].

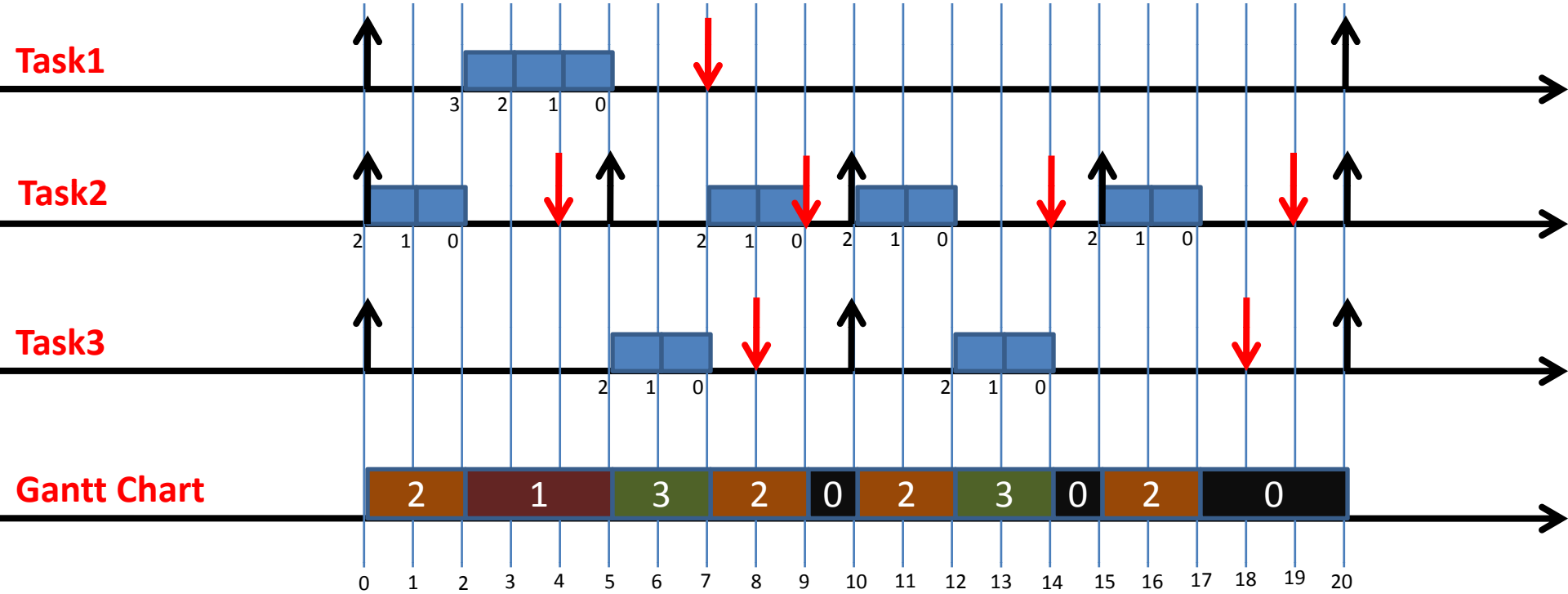
# Earliest Deadline First Scheduling (EDF)

- **Etape 3** : de tracer les chronogrammes.



# Earliest Deadline First Scheduling (EDF)

- **Etape 3** : de tracer les chronogrammes.




ReadyQueue

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T1 **T2** T3 | T1 **T2** T3 | T1 T3 | T1 T3 | T1 T3 | T2 **T3** | T2 **T3** | T2 | T2 | IDLE | **T2** T3 | **T2** T3 | T3 | T3 | IDLE | T2 | T2 | IDLE | IDLE | IDLE |

Politiques d’ordonnancement

# LEAST LAXITY FIRST SCHEDULING(LEAST SLACK TIME SCHEDULING -LST-) (LLF)

# Least Laxity First Scheduling (LLF)

- Least Laxity First est un ordonnancement à **priorité dynamique**. Les tâches sont d'autant **plus prioritaires** que leur **laxité est faible devant la date courante**.
- base sur la laxité résiduelle.
  - la priorité maximale est donnée a la tâche qui a la plus petite laxité résiduelle L(t) = D(t) – C(t).
  - L(t) : Laxité résiduelle à l'instant t.
  - D(t) : Délai critique résiduelle à l'instant t (**D(t) = D-t**).
  - C(t) : Burst Time restant à l'instant t.
- Modèle de tâche : Task(r0, C, D, P) avec $D \leq P$.

# Least Laxity First Scheduling (LLF)

- Etapes à suivre pour ordonnancer efficacement un ensemble de tâches par LLF.
- **Etape 1** : Test de faisabilité/ordonnancement.
- Si l'étape 1 est valide on passe vers les étapes ci-après.
- **Etape 2** : Calcul de l'intervalle de simulation par le calcul de HyperPeriod = LCM(P1, P2,…).
- **Etape 3** : de tracer les chronogrammes par:
  - De signaler sur long de chronogramme de chaque tâche : {r↑, D↓, P↑}
  - D'ordonnancer les tâches selon l'algorithme dans le diapo suivant.
  - De tracer le chrono **Gantt Chart (GC)**.

# Least Laxity First Scheduling (LLF)

- Algorithme d'ordonnancement par LLF :

**À chaque TICK_SYSTEM faire**

**1- Chercher les tâches à l'état READY (chargement de ReadyQueue)**

**2- Sélectionner la tâche de la laxité résiduelle la plus petite dans le ReadyQueue.**

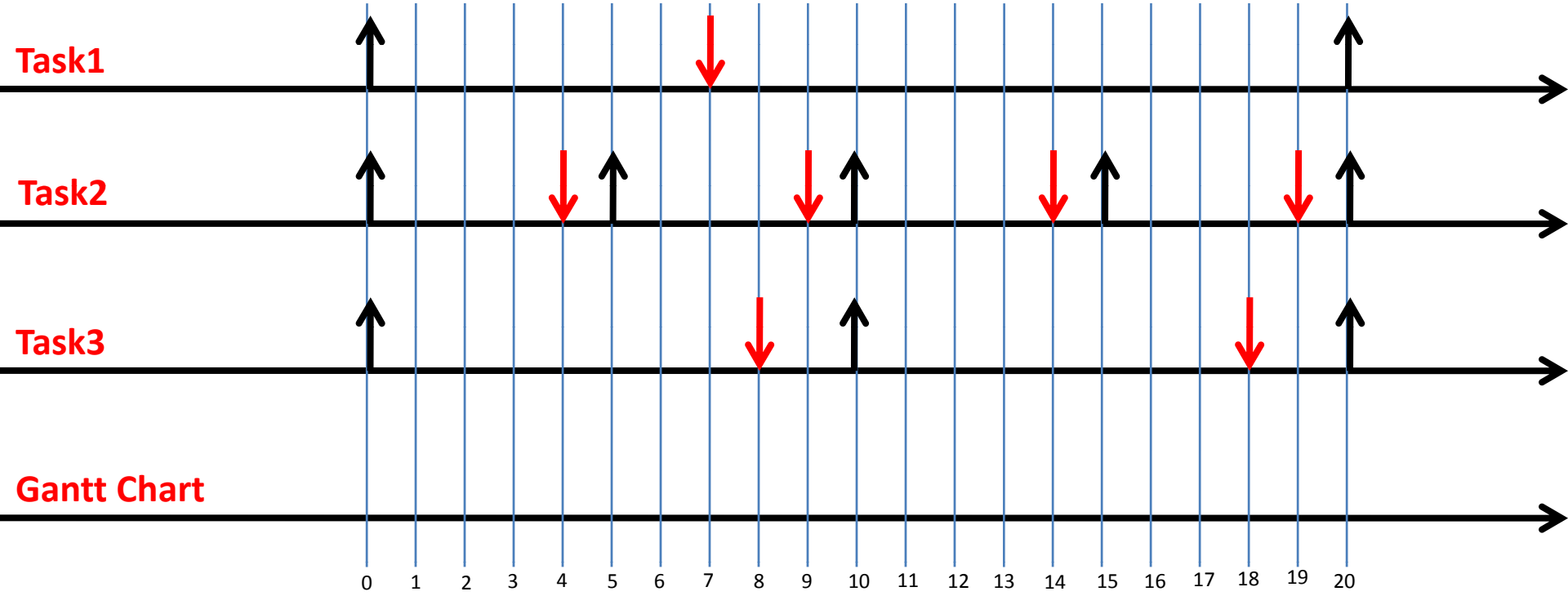
**3- Dessiner un rectangle sur le chrono de la tâche sélectionner au TICK_SYSTEM en cours.**

**4- Diminuer un TICK le BURST_TIME (C) de la tâche sélectionner.**

**Répéter les 4 étapes jusqu'à la fin de l'intervalle de simulation**

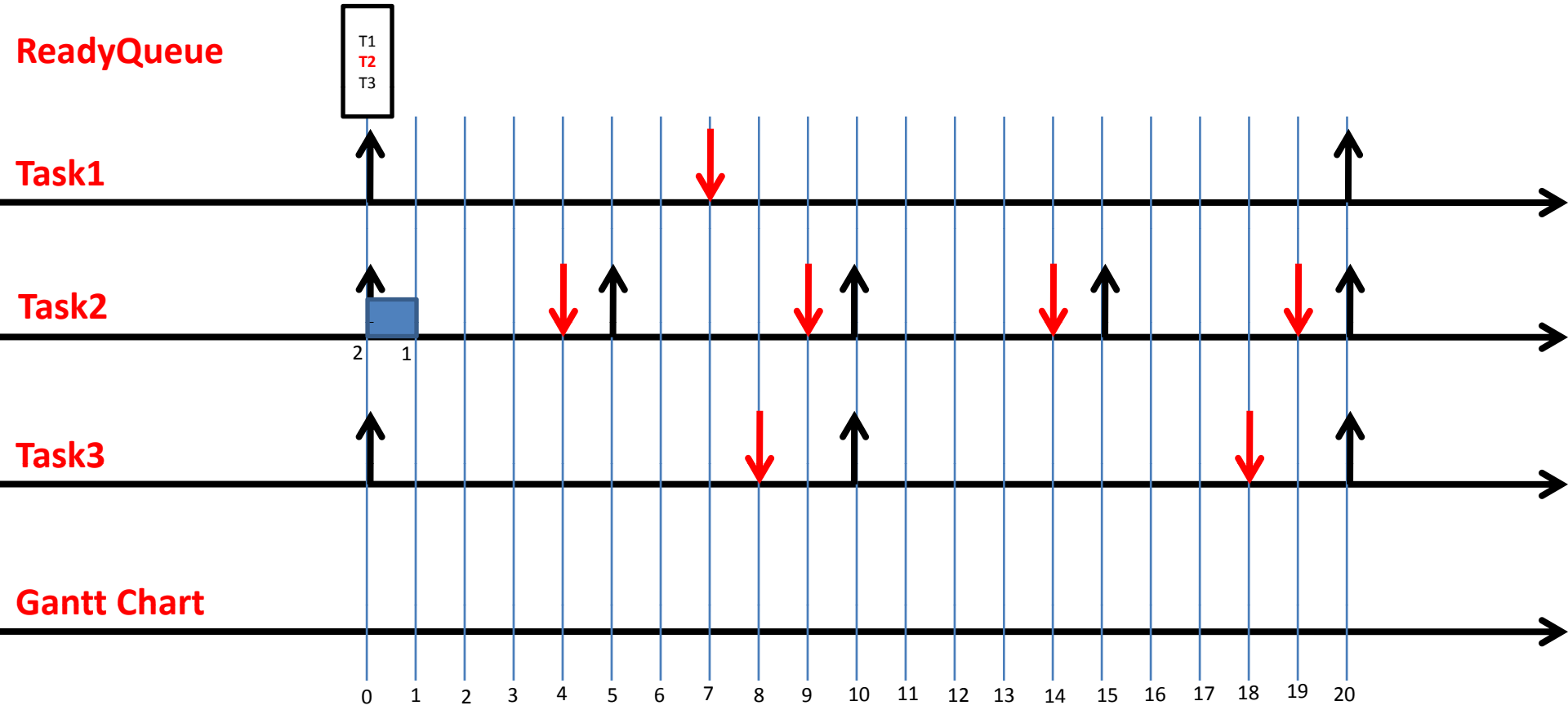
# Least Laxity First Scheduling (LLF)

- **Etape 3** : de tracer les chronogrammes.



# LLF

- **Etape 3** : de tracer les chronogrammes.




Calcul de laxité résiduelle pour (t=0) de chaque tâche:

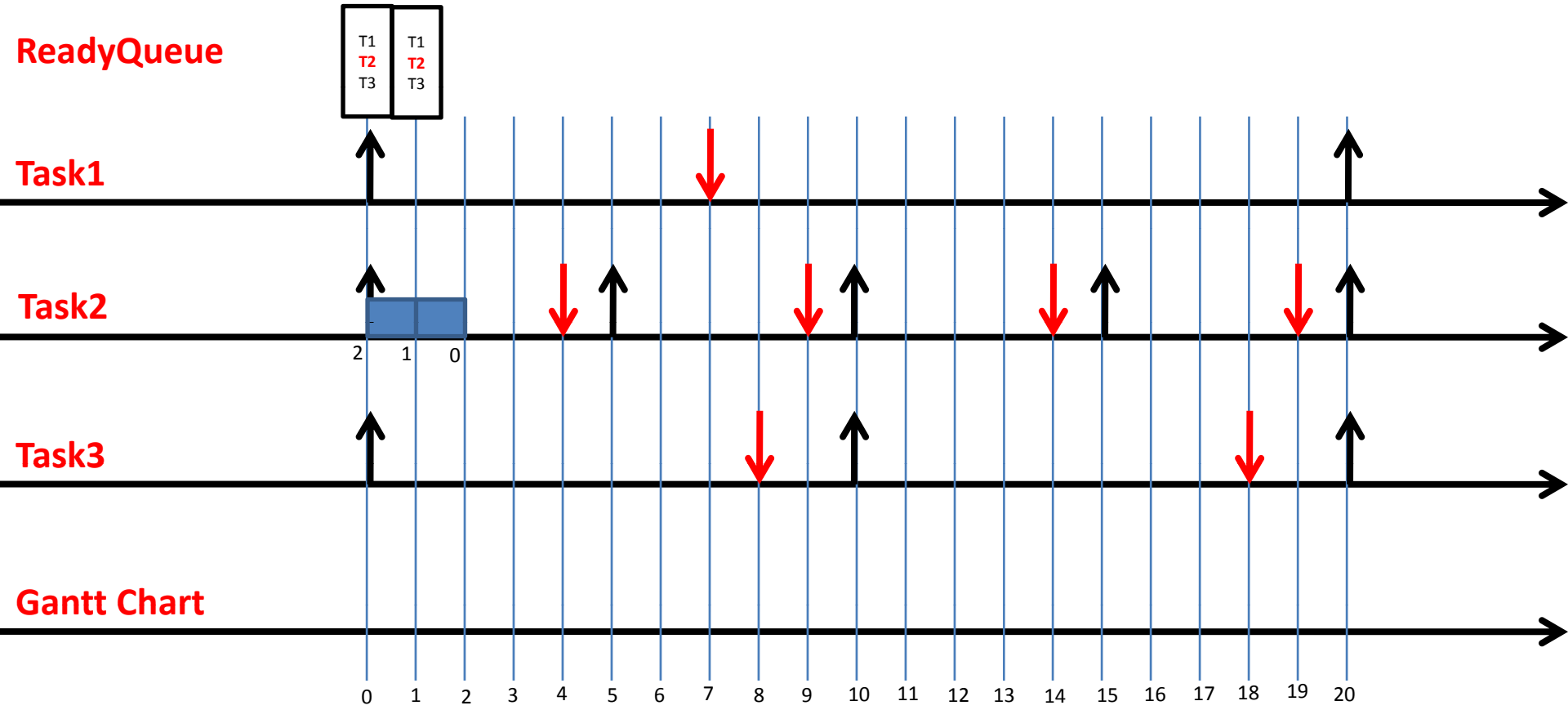
T1 : L1(0) = D1(0)-C1(0) = 7-3 = 4

T2 : L2(0) = D2(0)-C2(0) = 4-2 = 2

T3 : L3(0) = D3(0)-C3(0) = 8-2 = 6

# LLF

- **Etape 3** : de tracer les chronogrammes.




Calcul de laxité résiduelle pour (t=1) de chaque tâche:

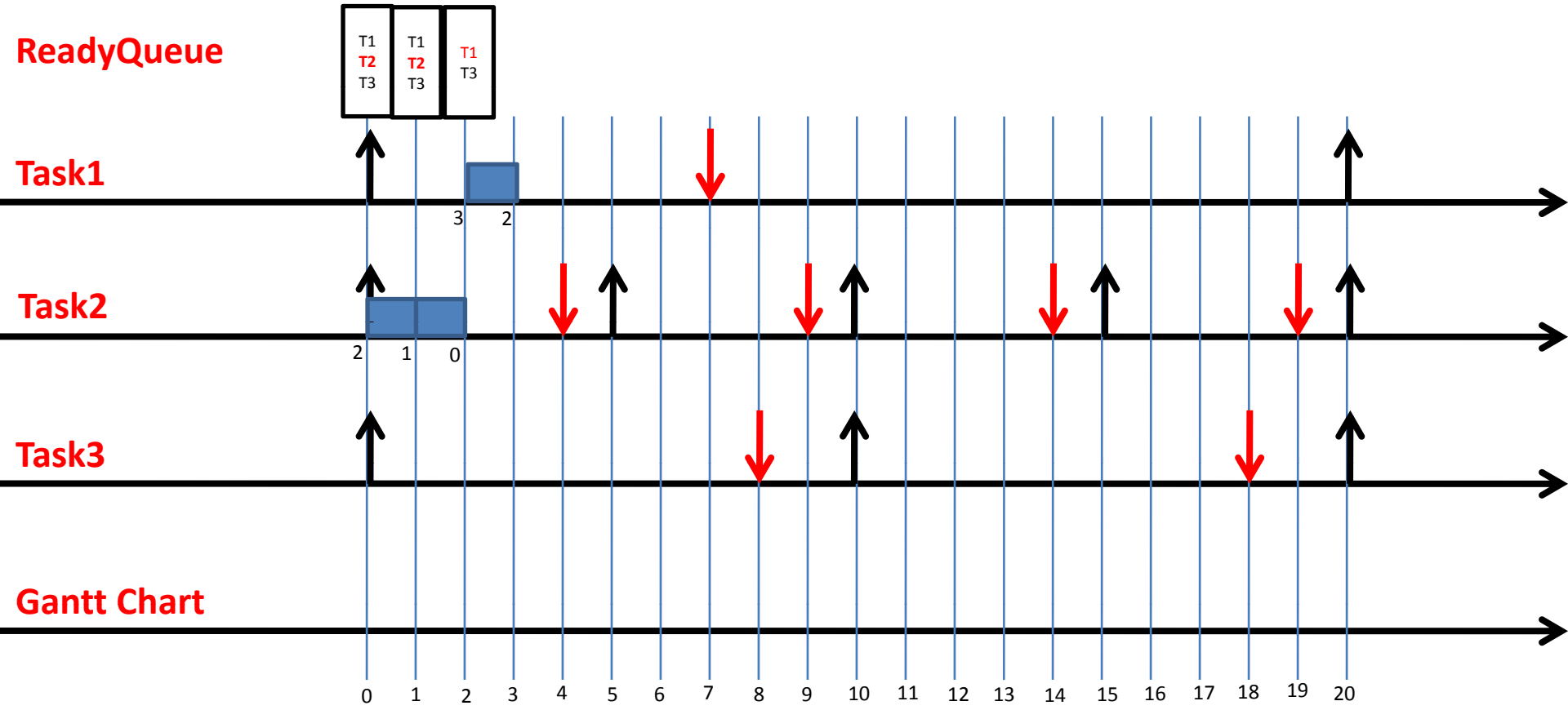
T1 : L1(1) = D1(1)-C1(1) = 6-3 = 3

T2 : L2(1) = D2(1)-C2(1) = 3-1 = 2

T3 : L3(1) = D3(1)-C3(1) = 7-2 = 5

# LLF

- **Etape 3** : de tracer les chronogrammes.



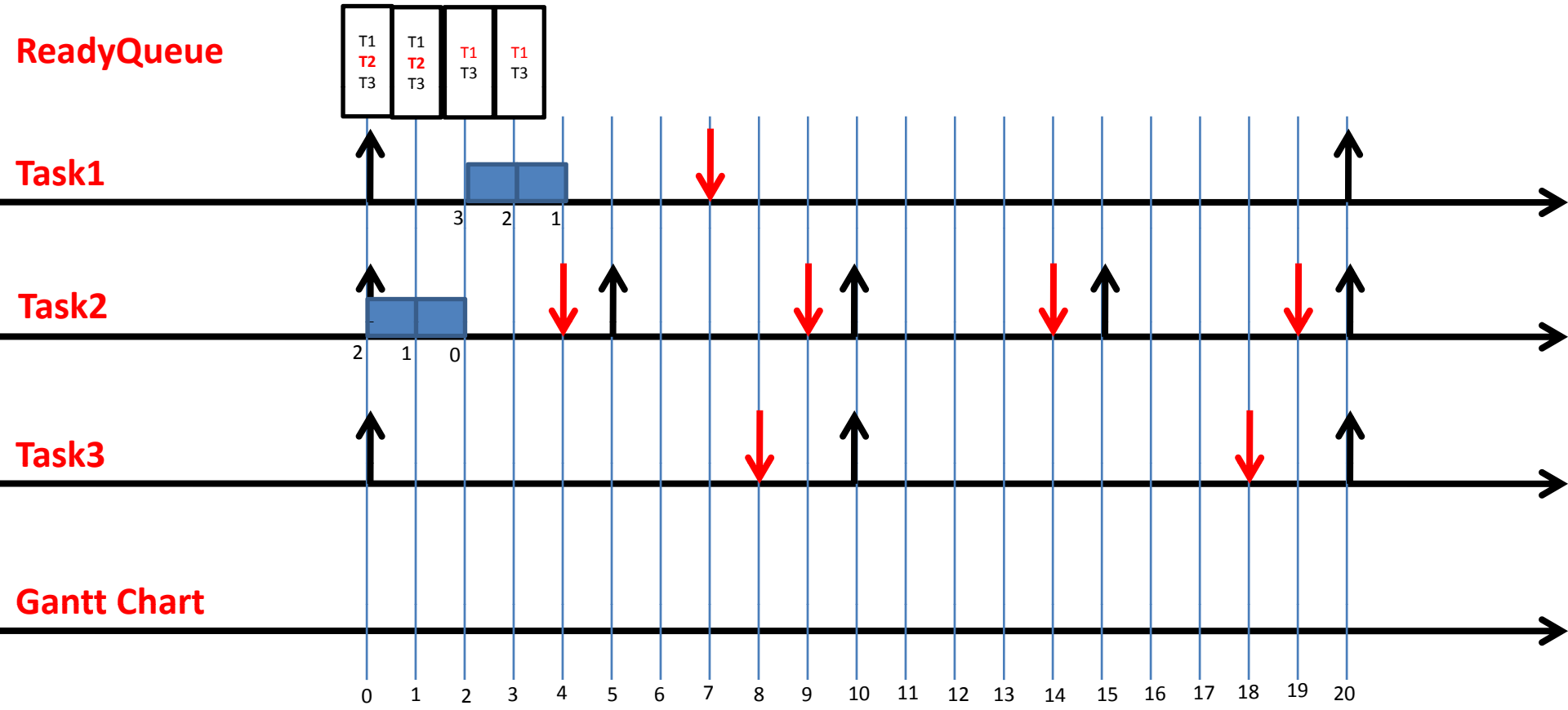

Calcul de laxité résiduelle pour (t=2) de chaque tâche:

T1 : L1(2) = D1(2)-C1(2) = 5-3 = 2

T3 : L3(2) = D3(2)-C3(2) = 6-2 = 4

# LLF

- **Etape 3** : de tracer les chronogrammes.



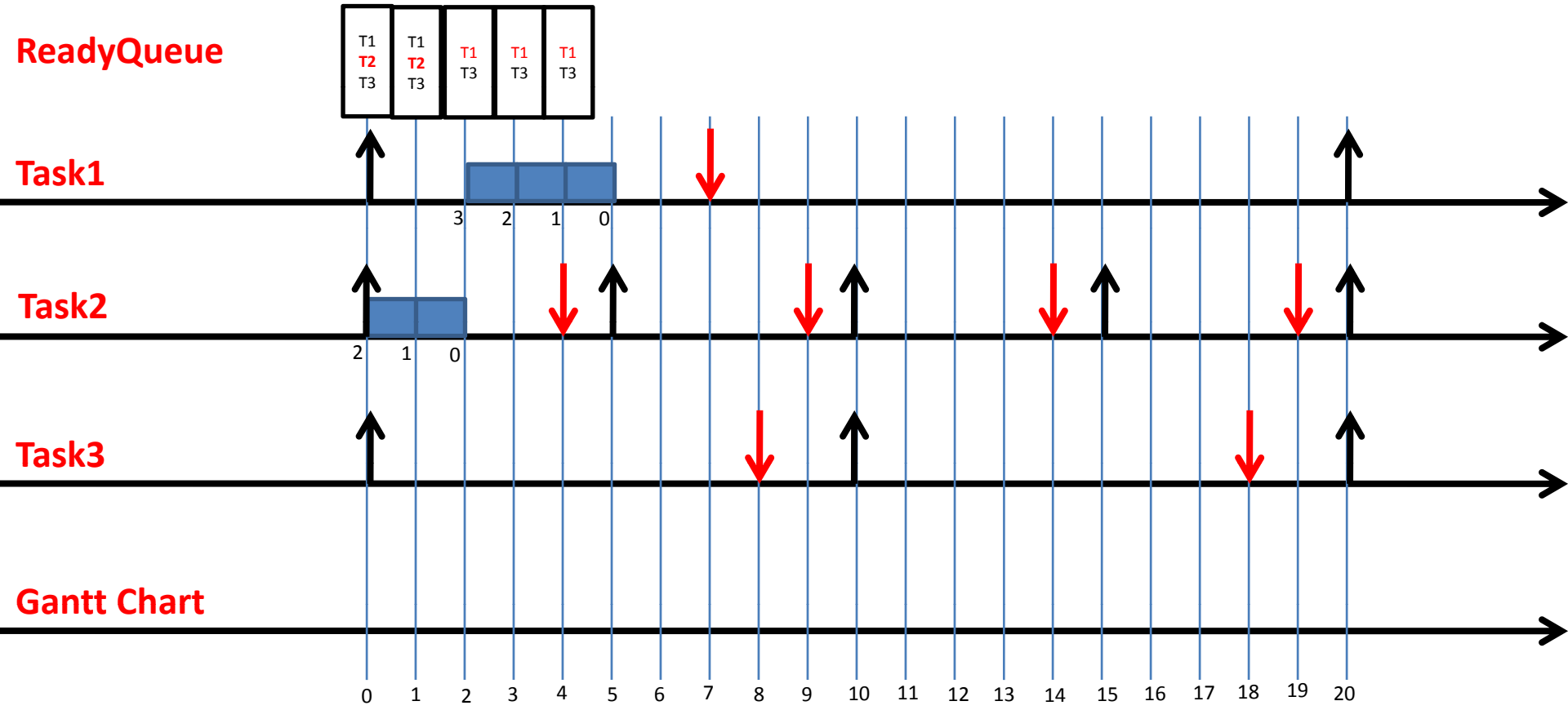

Calcul de laxité résiduelle pour (t=3) de chaque tâche:

T1 : L1(3) = D1(3)-C1(3) = 4-2 = 2

T3 : L3(3) = D3(3)-C3(3) = 5-2 = 3

# LLF

- **Etape 3** : de tracer les chronogrammes.



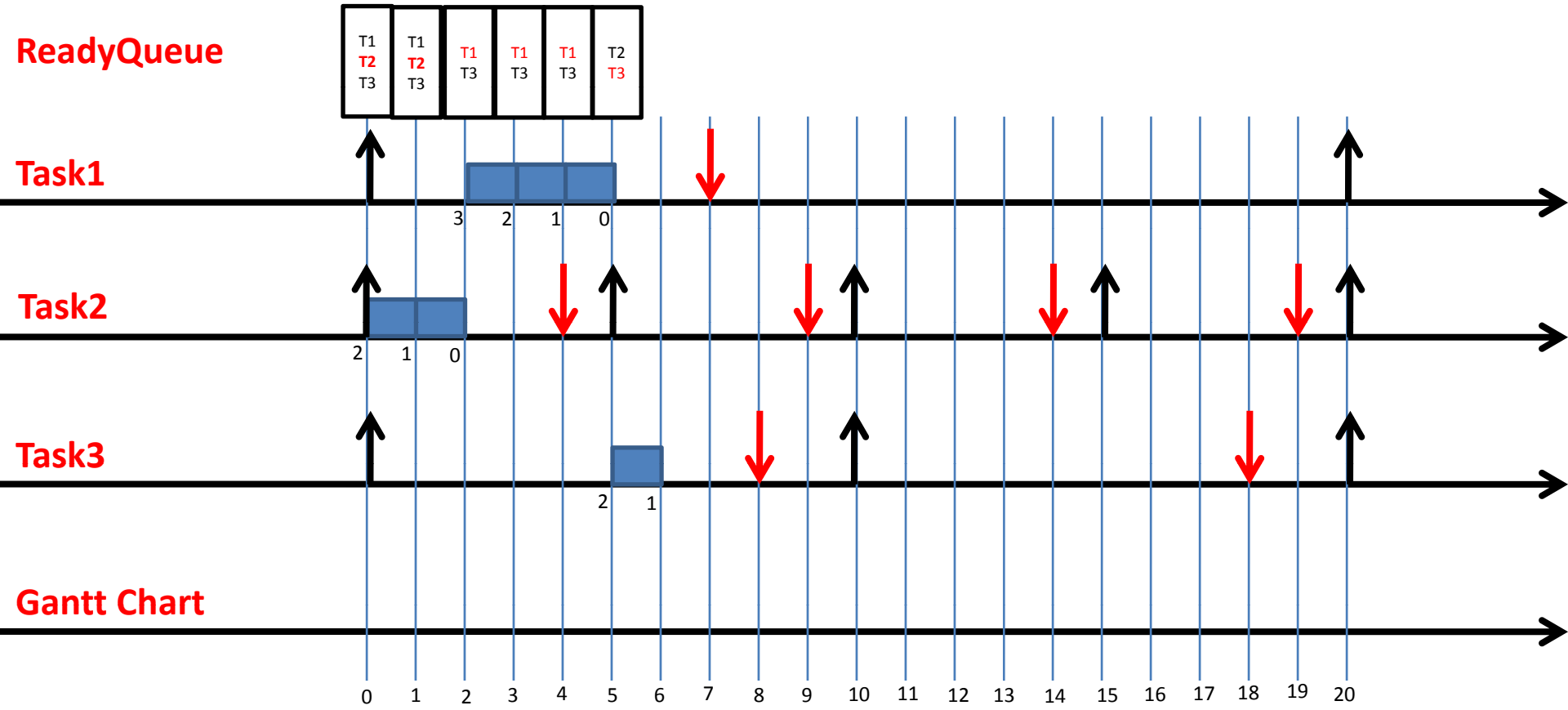

Calcul de laxité résiduelle pour (t=4) de chaque tâche:

T1 : L1(4) = D1(4)-C1(4) = 3-1 = 2 (Egalité ➔ Last Ext Task)

T3 : L3(4) = D3(4)-C3(4) = 4-2 = 2

# LLF

- **Etape 3** : de tracer les chronogrammes.



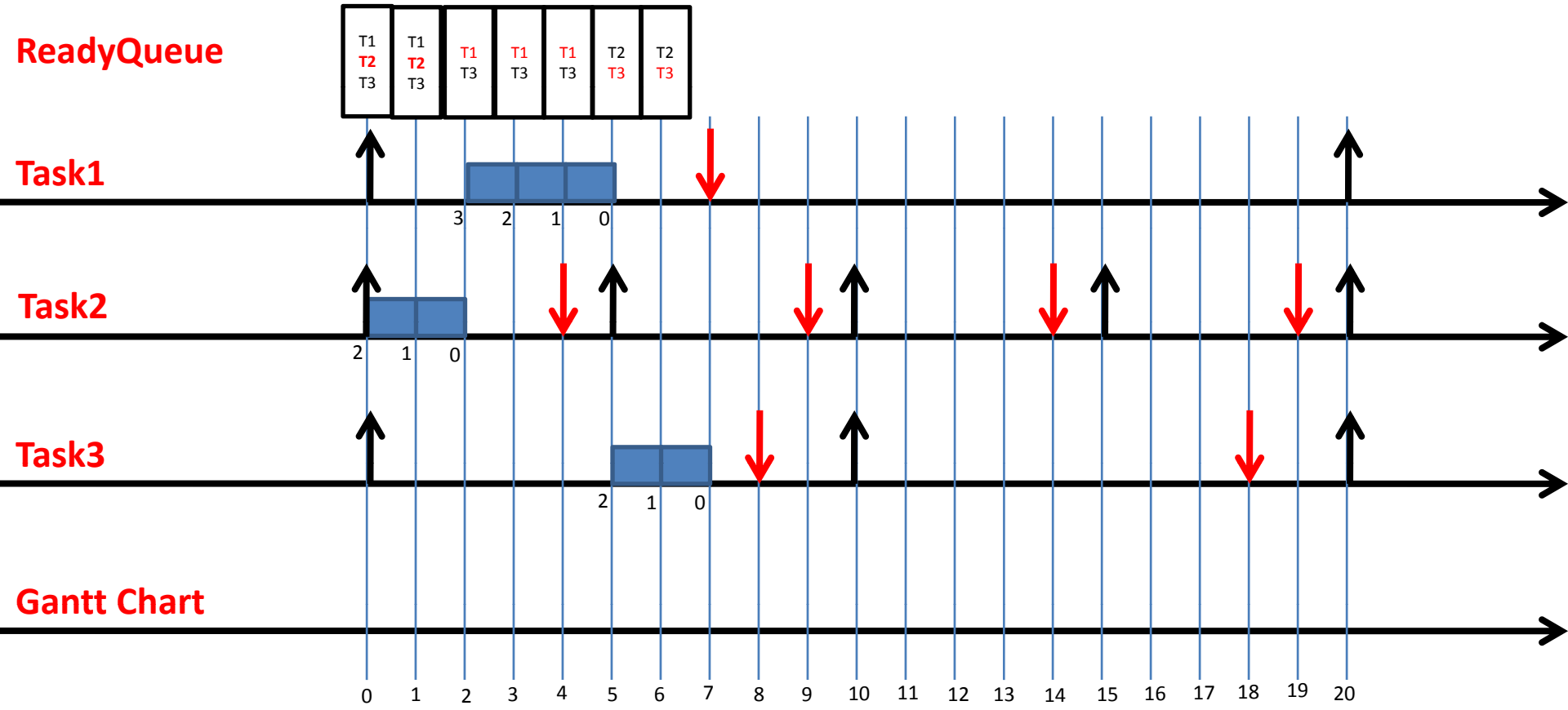

Calcul de laxité résiduelle pour (t=5) de chaque tâche:

T2 : L2(5) = D2(5)-C2(5) = 4-2 = 2

T3 : L3(5) = D3(5)-C3(5) = 3-2 = 1

# LLF

- **Etape 3** : de tracer les chronogrammes.



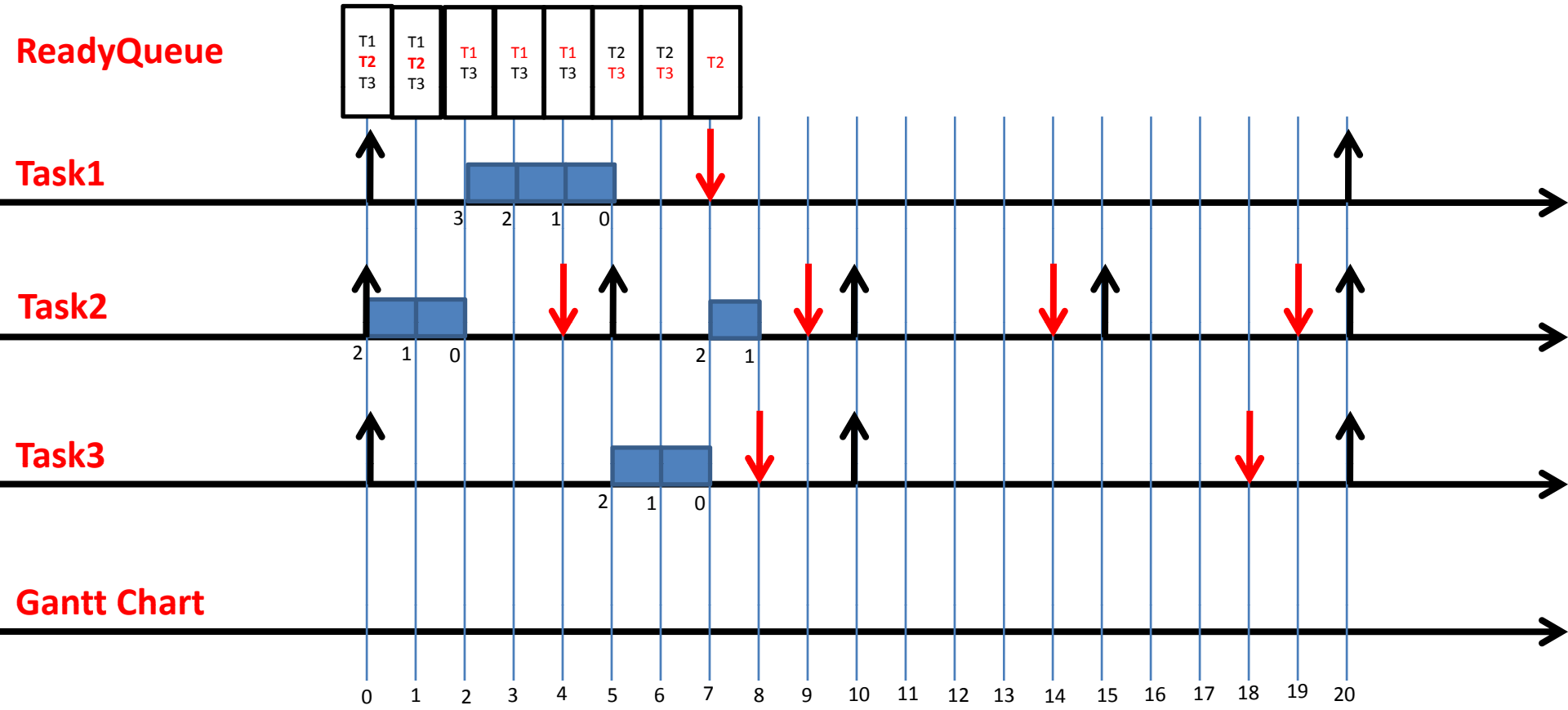

Calcul de laxité résiduelle pour (t=6) de chaque tâche:

T2 : L2(6) = D2(6)-C2(6) = 3-2 = 1

T3 : L3(6) = D3(6)-C3(6) = 2-1 = 1 (Egalité ➔ Last Ext Task)

# LLF

- **Etape 3** : de tracer les chronogrammes.




Calcul de laxité résiduelle pour (t=7) de chaque tâche:

T2 : L2(7) = D2(7)-C2(7) = 2-2 = 0

**Jusqu'à la fin**

# LLF

- **Etape 3** : de tracer les chronogrammes.