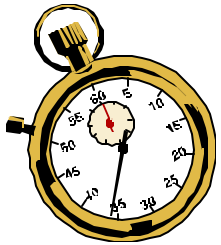
Ordonnancement temps réel

Zoubir MAMMERI

IRIT - UPS

(SITEF - 19 Octobre 2000)

Applications temps réel

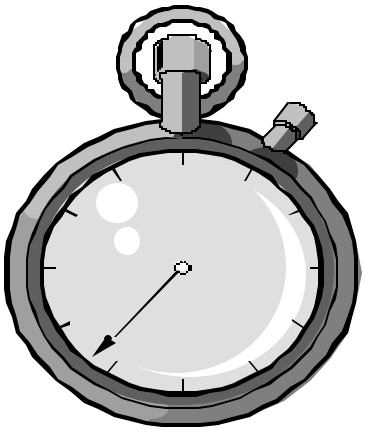


et

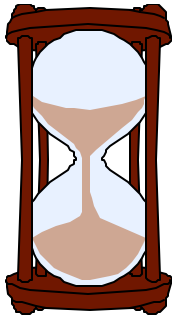
Correction logique

Correction temporelle

Temps : Critère fondamental



Top départ



Durée

Echéance

- **individuelles**
- **collectives**

Sources des contraintes de temps

- **processus physique (lois de commande)**
- **qualité de servie (qualité audio/vidéo)**
- **choix d'architecture support**
- **choix de conception**
- **autres**

Contraintes de temps

- **Objectifs : propriétés sur la manière dont se réalise une fonction (échéance, temps de réponse, gigue, ...).**

Comment spécifier les CT ?

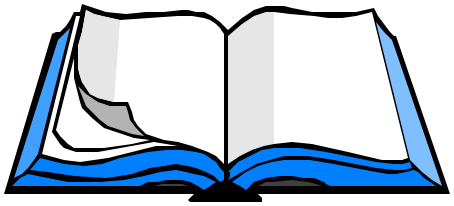
- **Moyens : ressources et gestion de ressources (processeurs, mémoire, réseaux, ...)**

Comment garantir les CT et avec quel degré ?

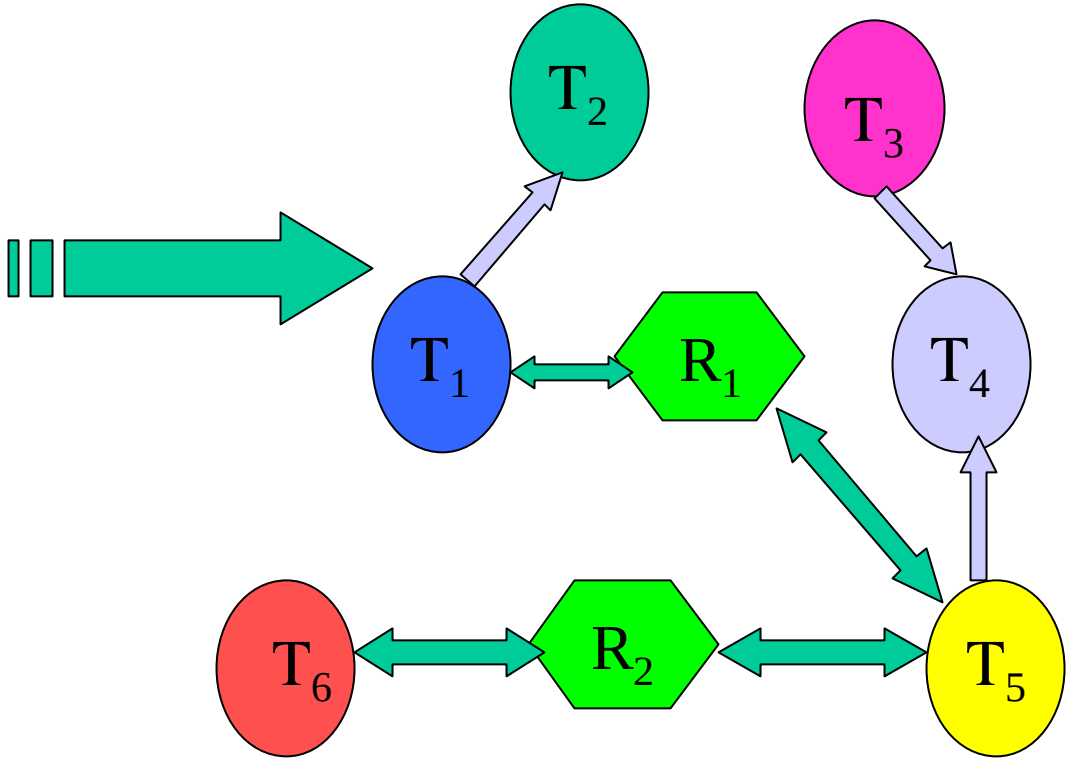
Plan

1. Modèles de tâches
2. Classes d'algorithmes d'ordonnancement
3. Principaux algorithmes d'ordonnancement de tâches indépendantes
4. Principaux algorithmes d'ordonnancement de tâches dépendantes
5. Répartition et ordonnancement
6. Ordonnancement de messages
7. Analyse d'ordonnançabilité
8. Conclusion

1. Modèles de tâches (contraintes)



Cahier des charges

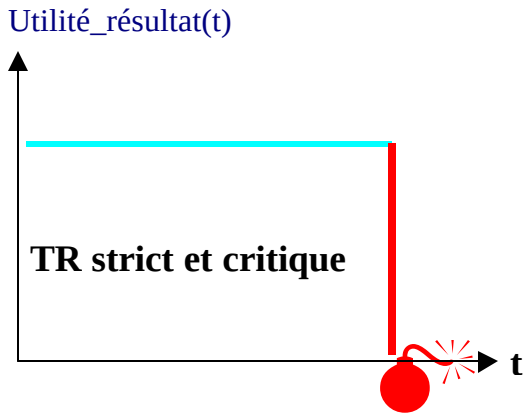


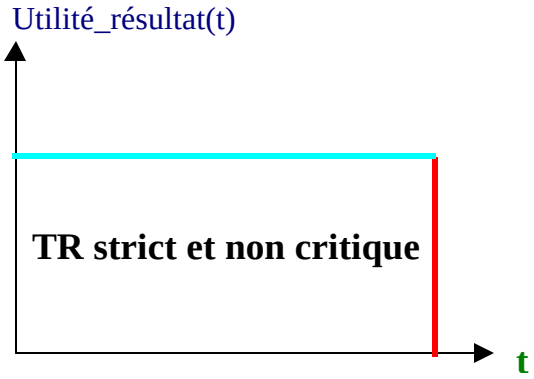
Ensemble de tâches

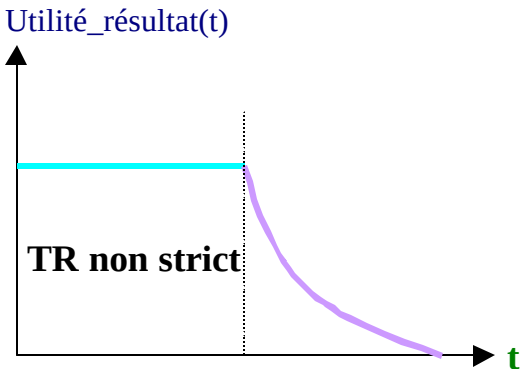
De quel temps réel s'agit-il ?

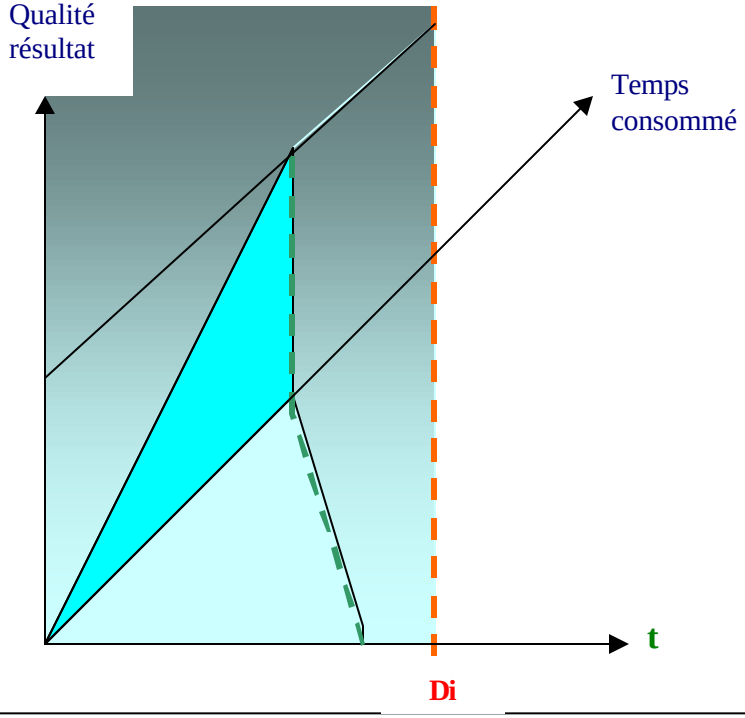
- CT strictes ou non strictes
  - Þ Validité du résultat dans le temps

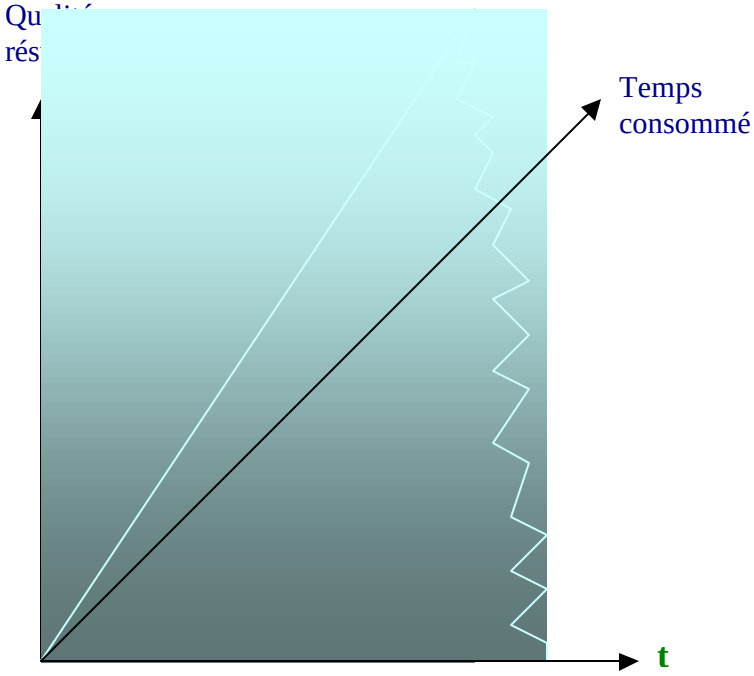
- Temps réel critique ou non
  - Þ Conséquences des fautes temporelles


Utilité_résultat(t)
TR strict et critique
t


Utilité_résultat(t)
TR strict et non critique
t


Utilité_résultat(t)
TR non strict
t


Qualité
résultat
Temps
consommé
t
Di


Temps
consommé
t

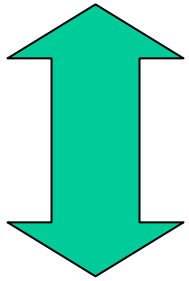
Moyens à mettre en œuvre

Ressources matérielles

Ressources logicielles

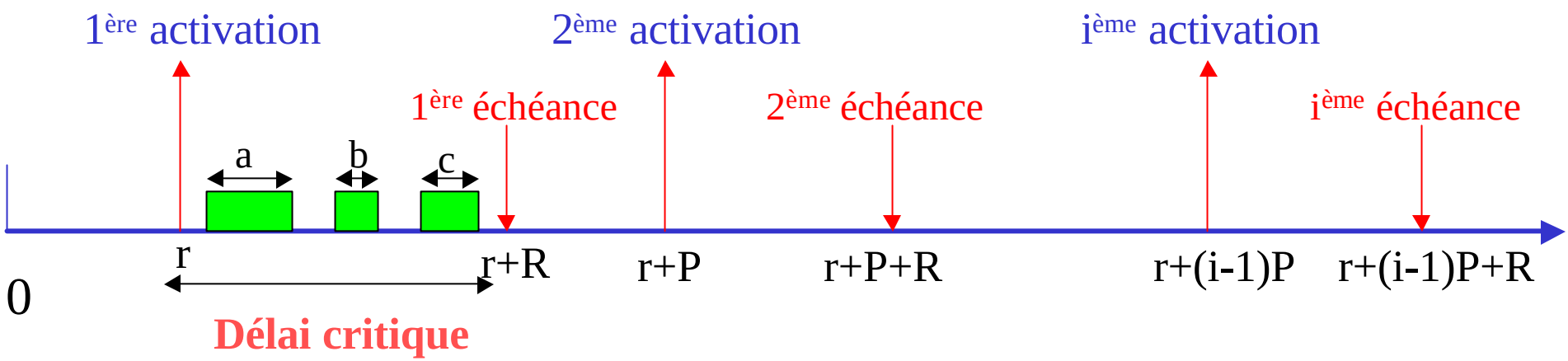
Techniques de spécification et de validation

Techniques d'ordonnancement et d 'analyse d'ordonnançabilité



Nature de l'application temps réel (« criticité »)

Modèle simple
(tâches périodiques)



r : date de première activation

C : pire durée d'exécution = Max(a) + Max(b) + Max(c)

R : délai critique

P : période

Charge-processeur d'une ATR

$\frac{C_i}{P_i}$: **Pourcentage de l'activité du processeur dédiée à la tâche T_i**

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i}$$: **Charge processeur = taux d'utilisation du processeur pour les tâches périodiques**

U > m ⇒ application non ordonnançable sur m processeurs
(Condition nécessaire, mais pas suffisante)

Modèle complet de tâche

(contraintes individuelles)

- Type de tâche (critique ou non)
- Priorité/importance de tâche
- Autorisation de réquisition du processeur
- Contraintes temporelles
  - ➔ Tâche périodique ou apériodique
  - ➔ Date au plus tôt de démarrage
  - ➔ Date au plus tard de terminaison
  - ➔ Durée maximale d'exécution
  - ➔ Durée moyenne d'exécution
  - ➔ Gigue
  - ➔ Autres
- Contraintes liées à la précision des résultats des tâches

Modèle complet de tâche
(contraintes collectives)

- **Contraintes de relations entre tâches :**
  - **Précédence**
  - **Partage de ressources**
- **Contraintes de distribution**
- **Contraintes de tolérance aux fautes**

2. Classes d'algorithmes d'ordonnancement

- **Ordonnancement = liste (partiellement) ordonnée d'accès aux ressources partagées**
- **Entités ordonnançables : tâches, threads, processus, messages, …**
- **Ordonnancement valide : Ordo qui satisfait les CT**

Deux familles d'algorithmes

Ordonnancement statique

- Connaissance statique de tous les paramètres de tâches
- Adapté à des systèmes « figés »

Beaucoup de résultats pour l'analyse de l'ordonnançabilité

Prédictibilité : Oui

Ordonnancement dynamique

- Connaissance des paramètres de tâches au moment de leur arrivée
- Tient compte des changements ou événements

Peu de résultats pour l'analyse de l'ordonnançabilité

Prédictibilité : NON

Ordonnancement hors ligne

- Calcul préalable d'une séquence ordonnancement suivie en ligne par un dispatcheur

Ordonnancement en ligne

- Elaboration de séquence d'ordonnancement à l'exécution

Préemptif	Non préemptif
Dirigé par une table statique	Dirigé par les priorités
A priorités statiques	A priorités dynamiques
Optimal	Non optimal
Stable	Instable
Monoprocesseur	Multiprocesseur
Local	Global (distribué)
Oisif retarde des décisions même si la file n'est pas vide (utile en milieu décentralisé)	Non oisif Pas de temps creux si une tâche est prête

Autres critères de classification des algorithmes

- Complexité
- Tolérance aux fautes
- Critère optimisé
  - Nombre de tâches ne respectant pas les CT
  - Longueur de l'ordonnancement
  - Equilibrage de charge entre les processeurs
  - Charge de communication
  - Autre
- Technique utilisée
  - Théorie des graphes
  - Algorithmes génétiques
  - Théorie des files d'attente
  - Autre
  - Recuit simulé
  - Réseaux de neurones
  - Logique floue

3. Principaux algorithmes d'ordonnancement de tâches indépendantes

- **A priorités statiques**

  Rate monotonic : fondé sur les périodes

  Deadline monotonic : fondé sur les délais critiques

- **A priorités dynamiques**

  Earliest deadline : fondé sur les échéances

  Least laxity : fondé sur la marge de manœuvre restante

Rate Monotonic (Liu et Layland 1973)

Principe
Priorité = F(Période)
Plus petite période Þ Plus prioritaire

RM est optimal

Condition d'ordonnançabilité (pour $P_i = D_i$) :

- **CS (théorème de la limite d'utilisation - Lui 73)**

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1) \qquad \approx \mathrm{Log}2 \approx 69\% \quad \text{qd } n \to \infty$$

- **CNS (théorème du temps de terminaison - Lehoczky, Sha et Ding 1989)**

Deadline Monotonic (Leung et Merrill 1973)

Principe
Priorité = F(Délai critique)
Plus petit délai critique Þ Plus prioritaire

DM est optimal (pour $P_i = D_i$)

Condition d'ordonnançabilité :

- CS : $$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1)$$

Earliest Deadline First (Liu et Layland 1973)

Principe

Echéance la plus proche Þ Plus prioritaire

EDF est optimal pour les systèmes de tâches indépendantes (pour $P_i = D_i$)

Condition d'ordonnançabilité pour les systèmes à échéance sur requête :

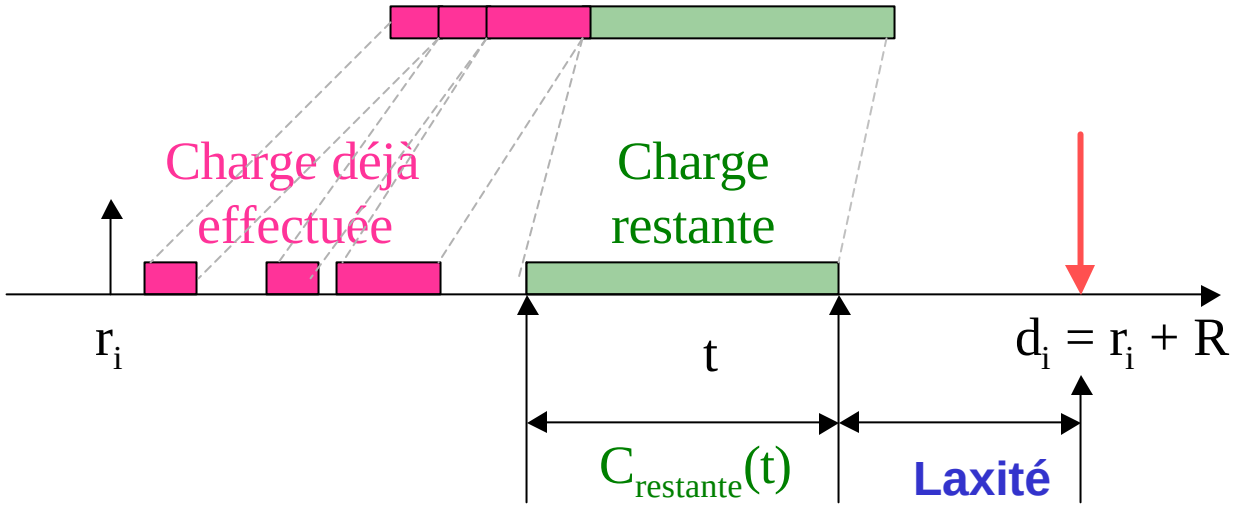
$$U \leq 1$$

Least Laxity (Sorenson 1974)

Principe

Laxité(t) = r_i + R - t - $C_{restante}$ (t)

Laxité plus petite Þ Priorité plus élevée



Même capacité d'ordonnancement que EDF

Défaut : Nombre élevé de changements de contexte

Recalcul des priorités toutes les unités de temps

Ordonnancement de tâches apériodiques indépendantes

- Cas des tâches apériodiques à contraintes strictes
  - ➔ Transformation de tâches apériodiques en tâches périodiques
  - ➔ Acceptation dans les temps creux d'une séquence rigide de tâches
- Cas des tâches apériodiques à contraintes relatives
  - ➔ Traitement d'arrière-plan ("background processing")
  - ➔ Les serveurs de tâches
    - Traitement par scrutation ("Polling")
    - Le serveur sporadique ("Sporadic Server")

4. Principaux algorithmes d'ordonnancement de tâches dépendantes

- **Contraintes de précédence**

- **Partage de ressources**

Prise en compte des contraintes de précédence

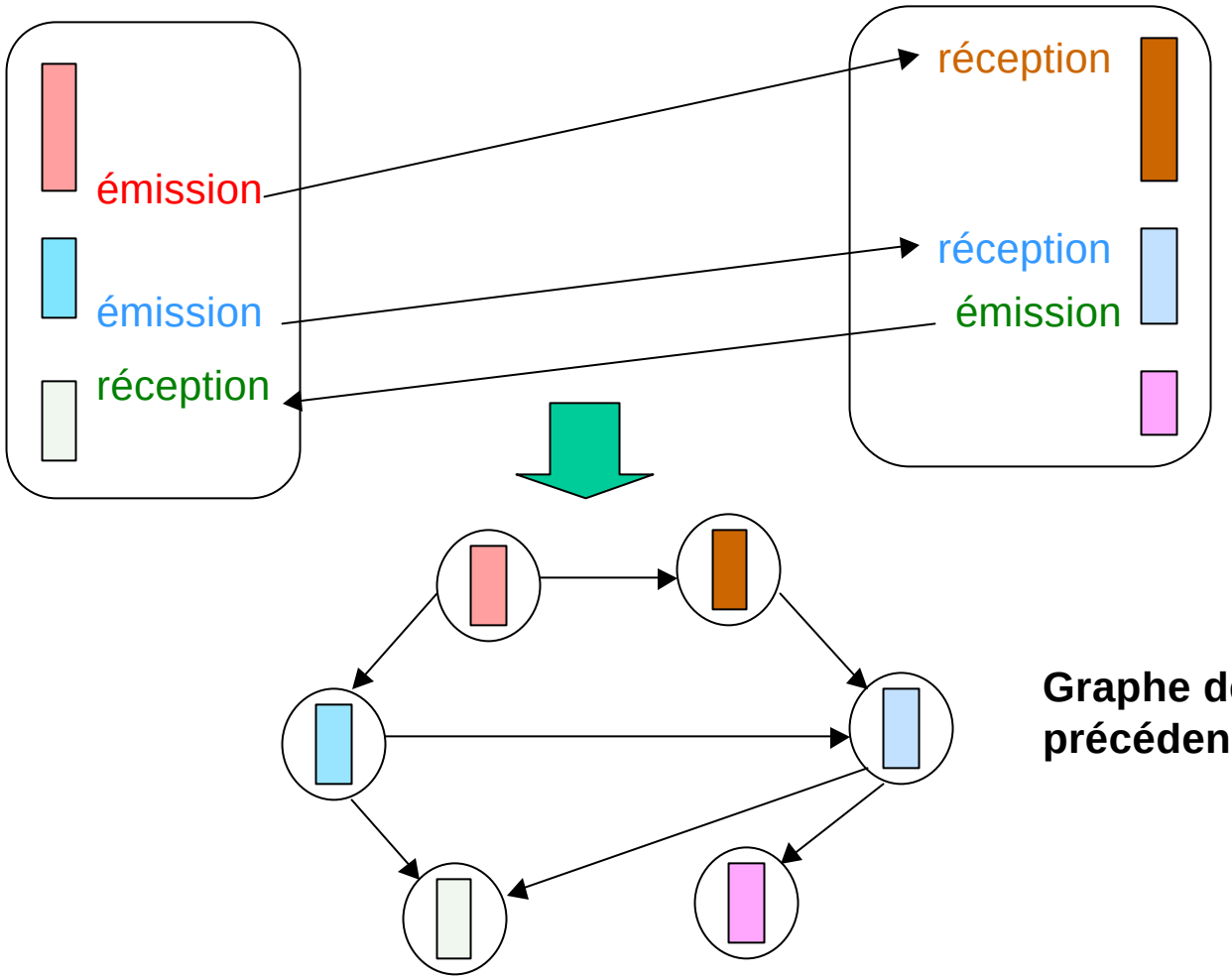
- Prise en compte de la communication
- Synchronisation

- **Principe** — Se ramener à l'ordonnancement de tâches indépendantes
- **Méthode** — Ajustement des dates de réveil et échéances

  Utilisation d'un algorithme classique : RM, DM ou EDF

  Égalité des priorités Þ utilisation du graphe de précédence

$$T_1 \longrightarrow T_2 \quad \text{Þ} \quad Prio_1 > Prio_2$$



2 tâches de même période

Graphe de précédence

- **Résultat**

Le système ajusté est ordonnançable par l'algorithme choisi

⟺ Le système initial est ordonnançable

Précédence et RM

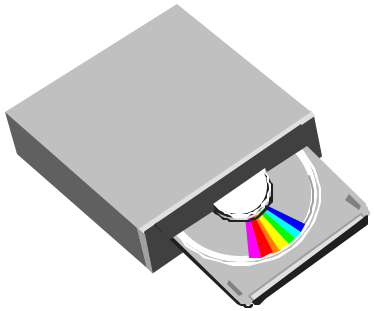
Ajustement : $r^*_i = Max\{r_i, r^*_k, T_k \mathrel{®} T_i\}$

Précédence et EDF

Ajustement : $r^*_i = Max\{r_i, Max\{r^*_k + C_k\}\}, T_k \mathrel{®} T_i$

$d^*_i = Min\{d_i, Min\{d^*_k - C_k)\}, T_k \mathrel{®} T_i$

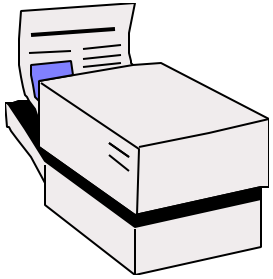
Prise en compte des contraintes de partage de ressources



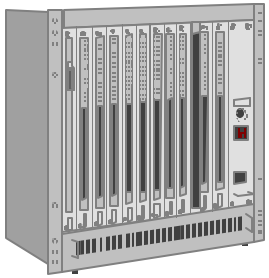
Fichiers, BD

Utilisation de ressources critiques

- **Mono ou multi exemplaires**
- **Mode lecture/écriture**
- **Demande de 1 ou n ressources**



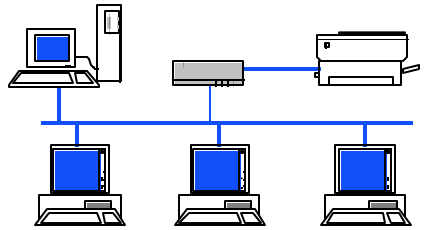
Ressource physique



E/S

Difficultés :

- **Interblocages**
- **Inversion de priorités**
- **Instabilité des algorithmes**



Réseau

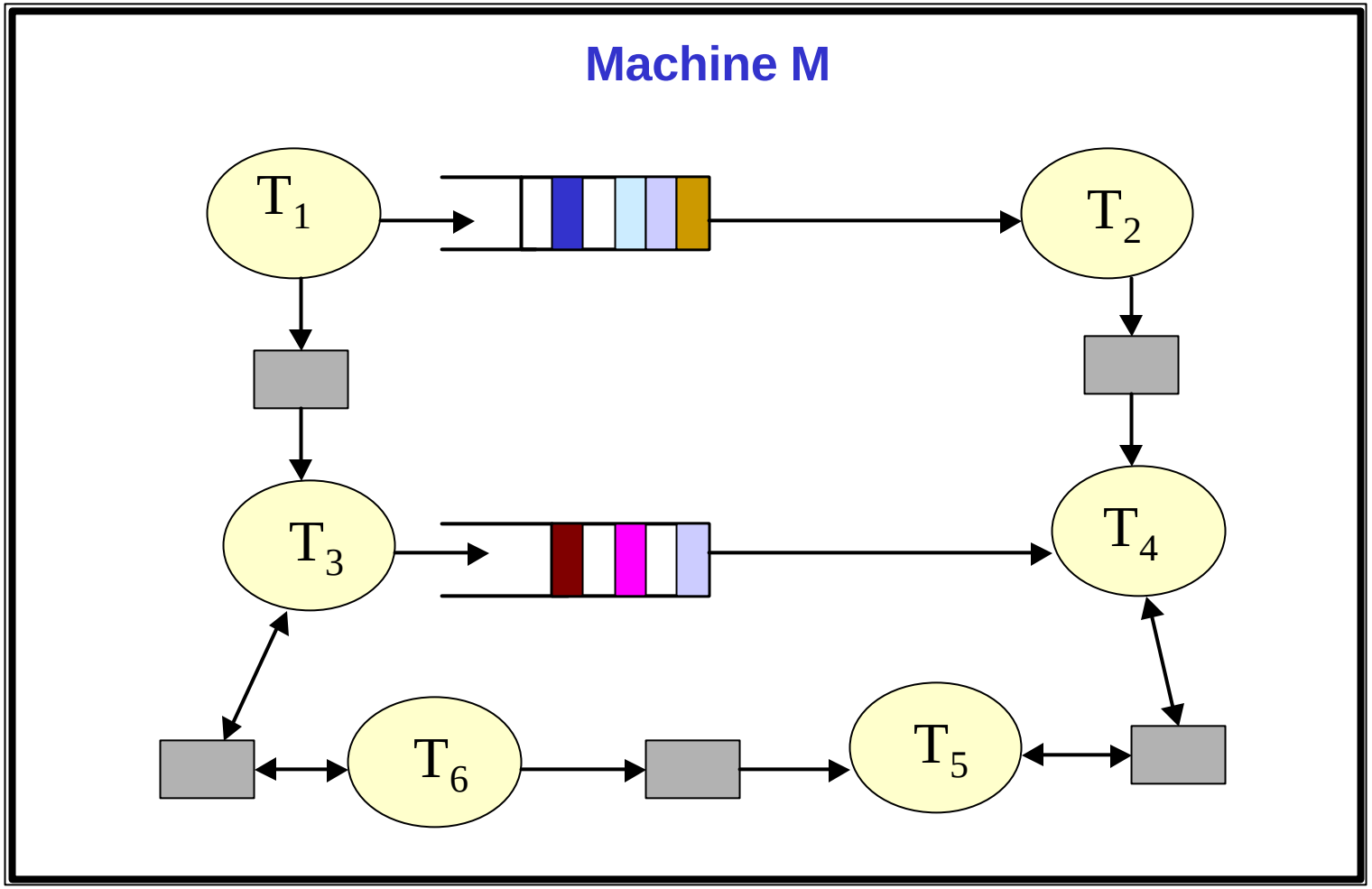
Ordonnancement : NP-difficile, pas d'algo optimal

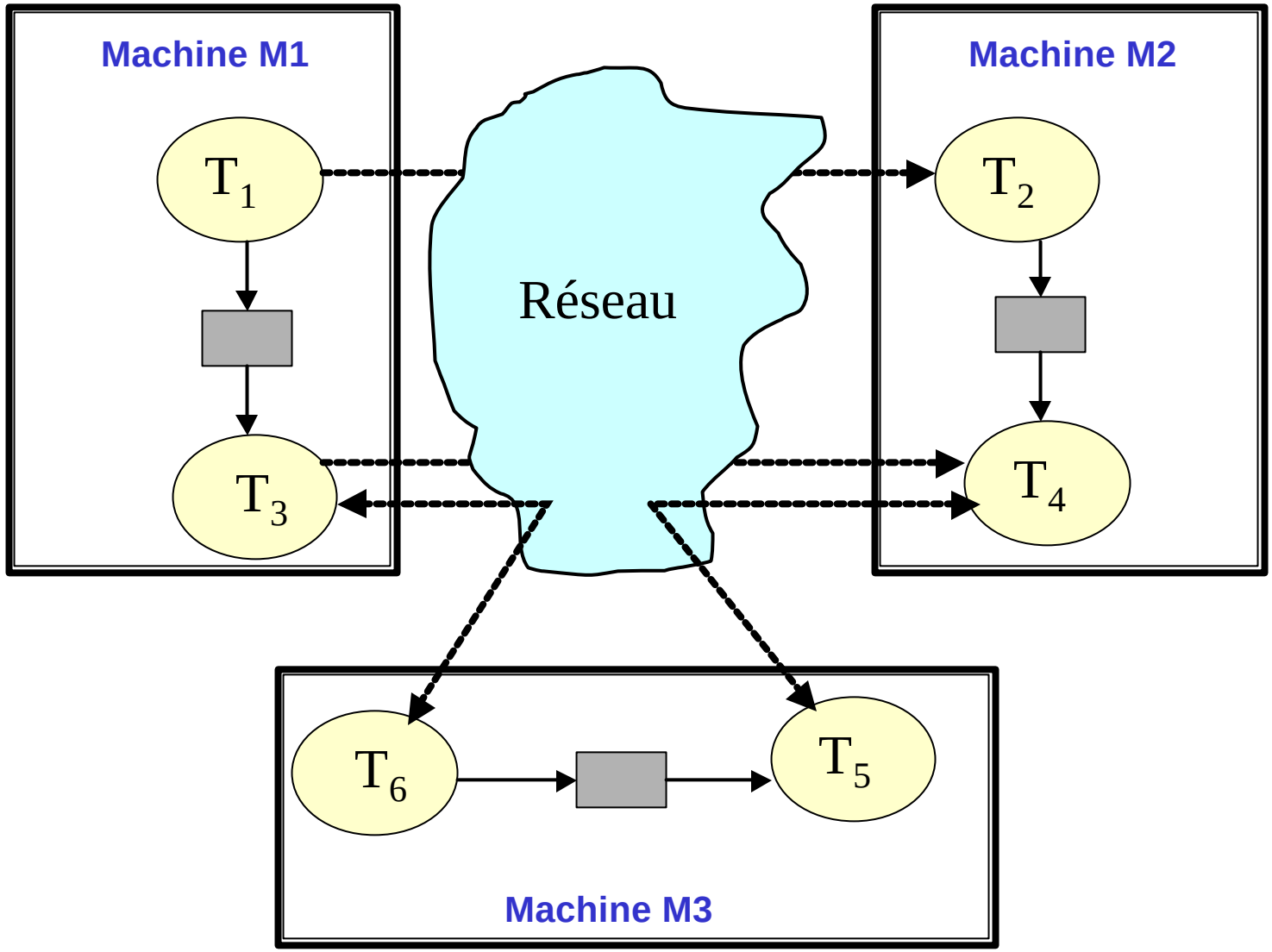
Principaux algorithmes

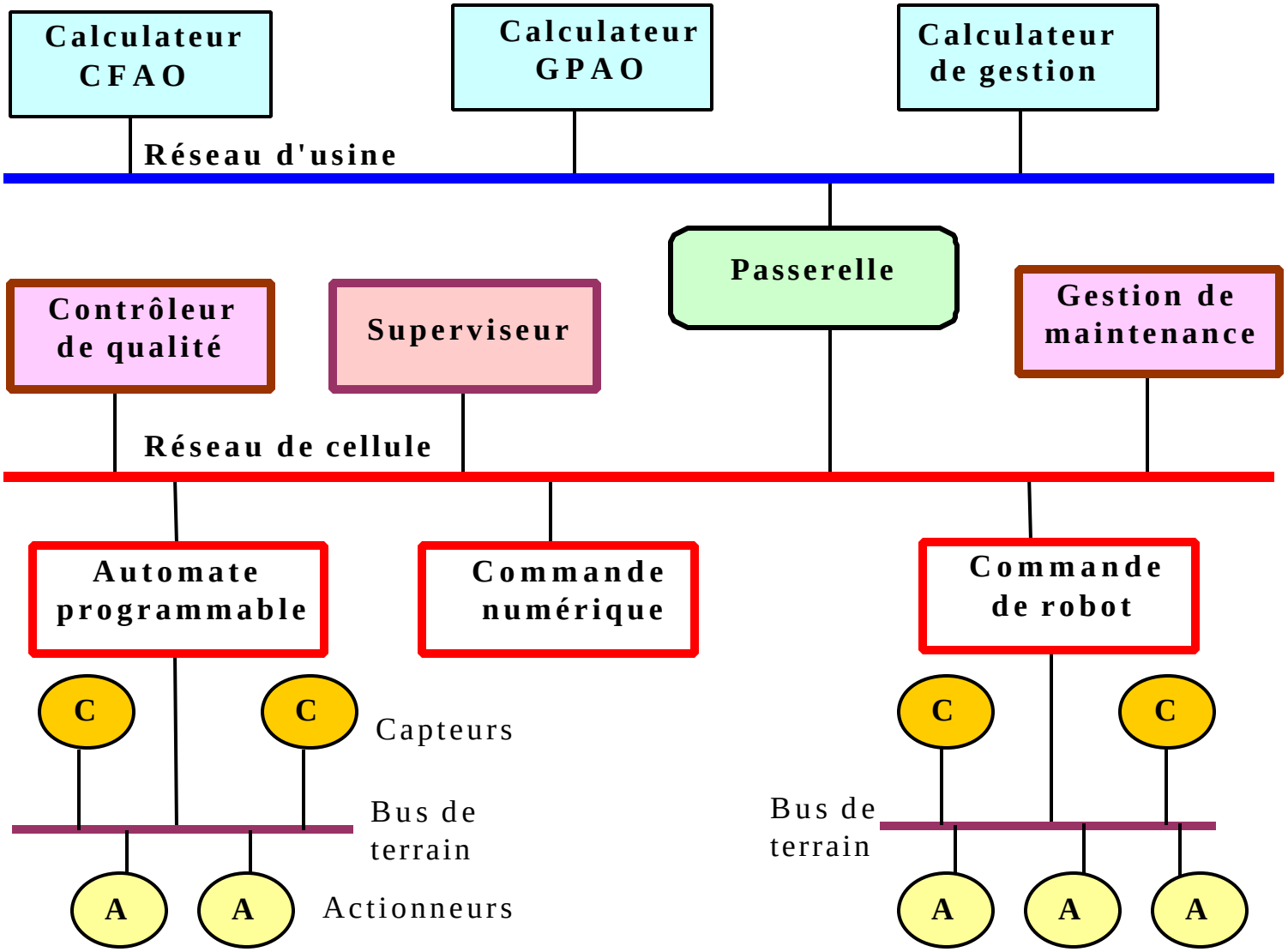
	Déclaration préalable des ressources	Prévention d'interblocage	Nombre de blocages	Calcul du de temps de blocage	Moment de blocage
Protocole à héritage de priorité (Priority Inheritance Protocol)	Non (transparent)	Non	Min(n,m)	Difficile	Au Moment de l'accès
Protocole à priorité plafond (Priority Ceiling Protocol)	Oui	Oui	1	Facile	Au Moment de l'accès
Protocole à pile de priorité (Stack Resource Policy)	Oui	Oui	1	Facile	Au moment de la préemption

n : nombre de tâches **m : nombre de ressources**

5. Répartition et ordonnancement




Machine M1
Machine M2
Réseau
T1
T2
T3
T4
T5
T6
Machine M3


Calculateur CFAO
Calculateur GPAO
Calculateur de gestion
Réseau d'usine
Passerelle
Contrôleur de qualité
Superviseur
Gestion de maintenance
Réseau de cellule
Automate programmable
Commande numérique
Commande de robot
C
C
Capteurs
Bus de terrain
A
A
Actionneurs
C
C
Bus de terrain
A
A
A

Systèmes répartis et temps réel

Problèmes à résoudre

- Nombre de machines
- Choix de réseau(x)
- Placement de tâches
  - Statique
  - dynamique
    - avec migration
    - sans migration

- Ordonnancement de tâches (local et distribué)
- Ordonnancement de messages

Cas extrêmes

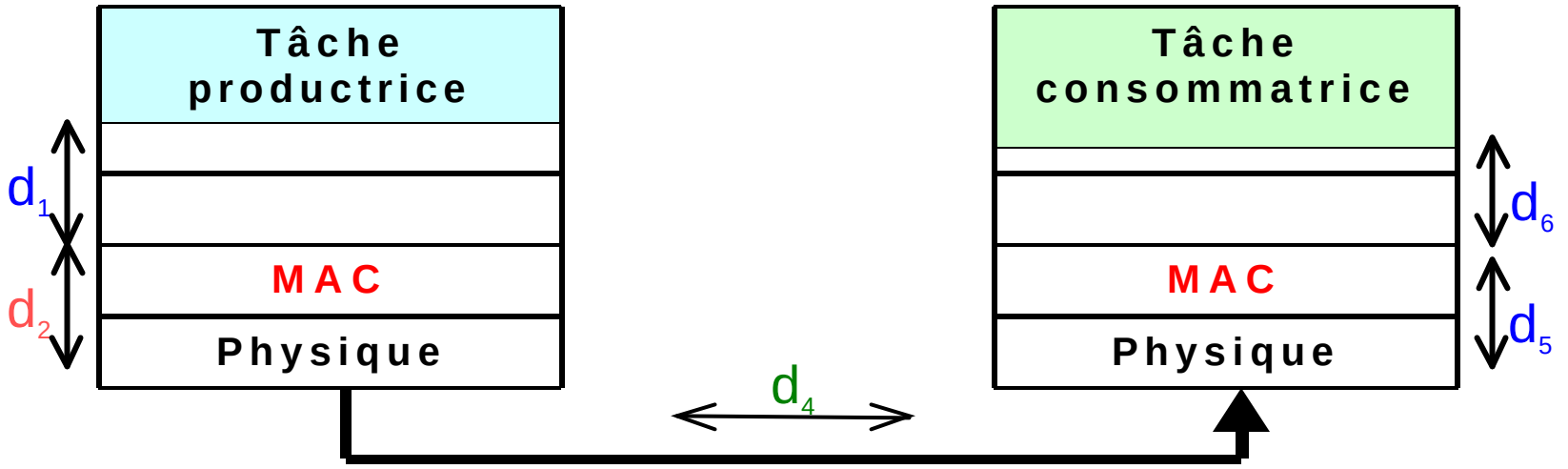
- **Tâches périodiques,**
  avec placement statique,
  sans partage de ressources distantes,
  échange de messages en fin d'exécution,
  sur un réseau avec temps de réponse borné

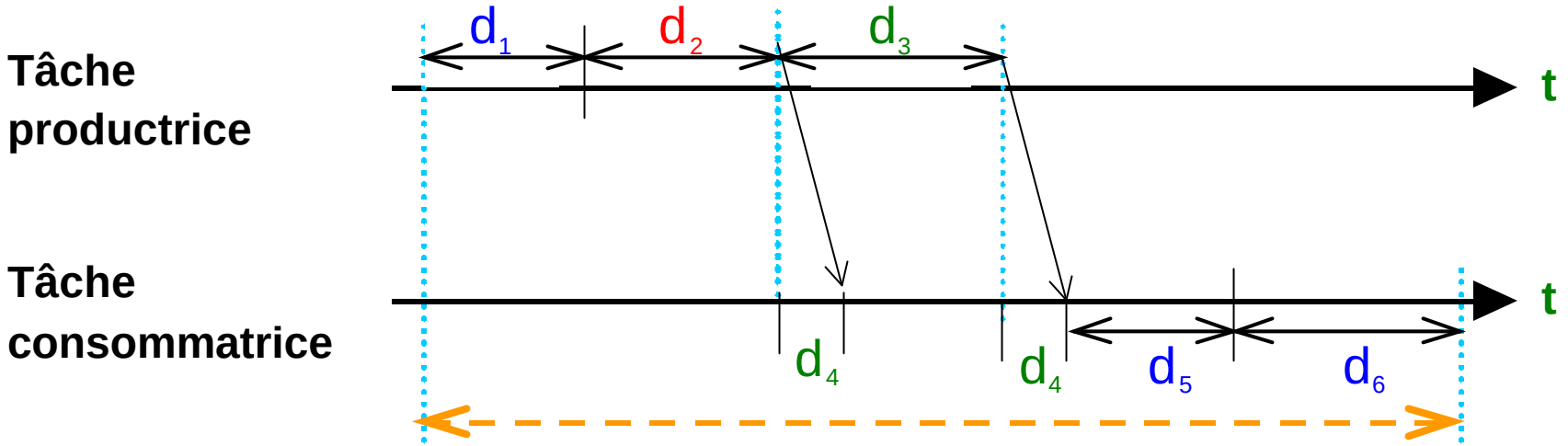
- **Les tâches arrivent à des instants inconnus,**
  partagent des ressources avec d'autres tâches distantes,
  peuvent migrer pendant leur exécution,
  fonctionnent su un réseau avec délai non borné

Modèles de trafics

- **Contraintes de temps :** **strictes critiques**
  strictes non critiques
  relatives
- **Type : périodiques, apériodiques (sporadiques, autres)**
- **Longueur**
- **Destinataires (connexions point-à-point, multipoint)**
- **Relations entre messages**
- **Qualité de service**
  - ➔ **Délai maximal de transfert, Délai moyen de transfert**
  - ➔ **Gigue maximale**
  - ➔ **Taux d'erreur maximal**
  - ➔ **Taux de perte maximal**
- **Autres critères**

Délai de transfert de message





Délai de bout-en-bout

6. Ordonnancement de messages

- **Couches hautes**
- **Couche réseau (IP, ATM, …)**
- **Couche liaison de données (MAC)**

Protocoles de niveau MAC

- **Types de contrôle d'accès**
  - **multiplexage fréquentiel ou temporel**
  - **compétition**
  - **consultation**

- **MAC représentatifs**
  - **CSMA/CD (Ethernet), CSMA/CA (CAN)**
  - **Jeton temporisé : Bus à jeton et FDDI**
  - **Boucle à jeton**
  - **Maître/esclave (FIP)**

Algorithmes d'ordonnancement de messages

- Affecter le support de manière à respecter les CT de messages
- Analogie avec l'ordonnancement de tâches
  - Quelle est la prochaine station à utiliser le médium ?
  - Quelle est la prochaine tâche à utiliser le processeur ?
- Adaptation d'algorithmes d'ordonnancement de tâches
- Beaucoup de travaux existent
  - La plupart des travaux concernent RM, EDF ou LLF
- Algorithmes pour messages périodiques et algorithmes pour messages apériodiques

7. Analyse d'ordonnançabilité

- **Primordiale pour les applications TR critiques**
- **Objectif : Eliminer les risques de fautes temporelles**
- **Calcul de divers temps**
  - ➔ **temps d'exécution**
  - ➔ **temps de changement de contexte**
  - ➔ **temps de réponse**
  - ➔ **temps de communication**
  - ➔ **temps de blocage**
- **Utilisation des conditions nécessaires et/ou suffisantes (preuve)**
- **Simulation**

  Durée de simulation : PPCM(P_i) ou Max(r_i) + 2*PPCM(P_i)
- **Coût : Le problème de l'ordonnancement est NP-difficile**

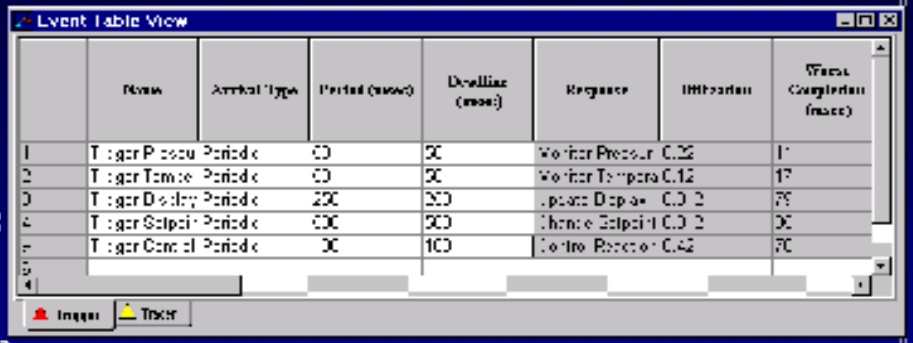
Outils pour l'analyse d'ordonnançabilité

- **PERTS (Prototyping Environment for Real-Time Systems)**
  University of Illinois
  commercialisé par Tri-pacific software
  Rapid RMA et Rapid SIM
  http://www.tripac.com/

- **TimeWiz de TimeSys Corp.**
  http://www.timesys.com/

Action Table View

- The *Action Table View* can be used to view/enter/modify values for trigger and tracer events
- Examples:
  - Arrival type
  - Period
  - Deadline
  - Response actions
  - Utilization
  - Worst completion

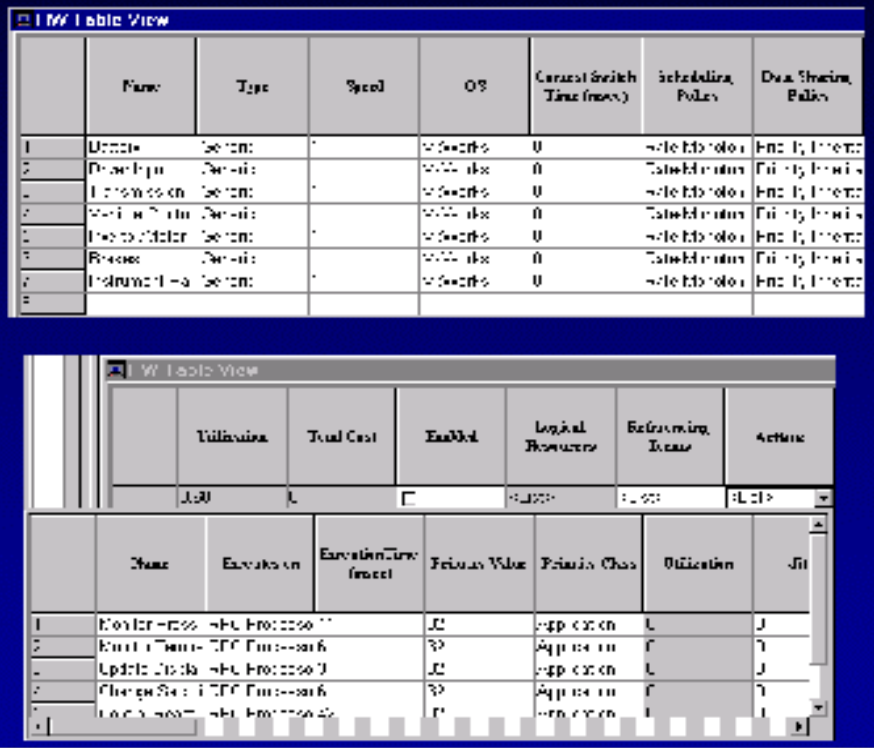


TimeSys Corporation

Real-Time... Real Solutions

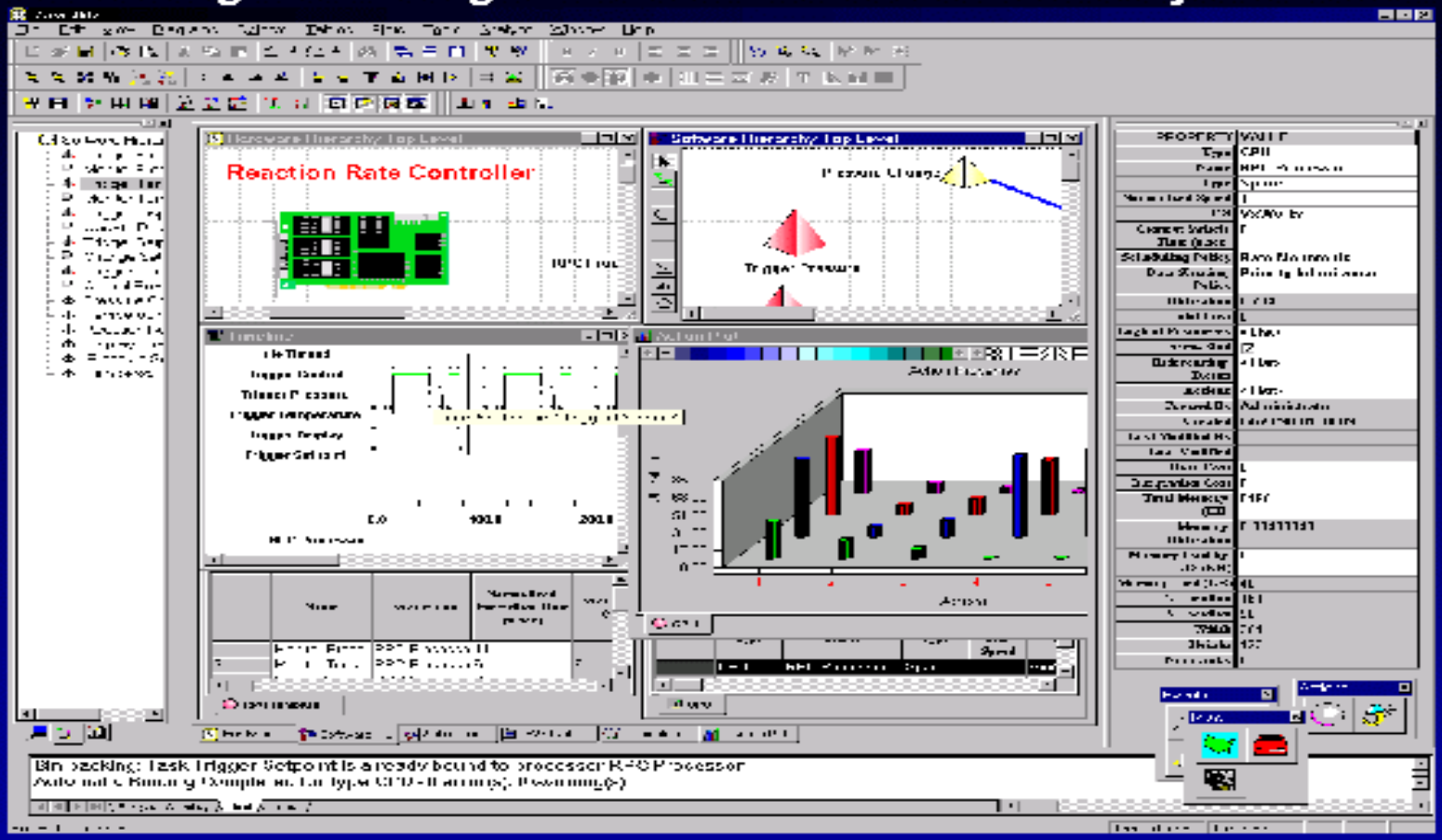
Hardware Table View

- The *Hardware Table View* is used to view/enter/modify values for different properties corresponding to groups of TimeWiz resource objects
- Examples:
  - Scheduling policy
  - Data-sharing policy
  - Context switch time
  - Logical resources
  - Utilization



TimeWiz™

An Integrated Design Environment for Real-Time Systems



TimeWiz is a unique and powerful product that lets you represent hardware and software, analyze and simulate the timing behavior of your system, model processors and networks for end-to-end performance, chart your system parameters and generate integrated system reports. All nicely integrated and affordable.

8. Conclusion

- **Variété d'algorithmes d'ordonnancement Þ**
  Des profils d'ordo selon les classes de problèmes

- **Choisir l'architecture support (problèmes de dimensionnement) et la stratégie d'ordonnancement en fonction de l'application**

- **Prouver ce que l'on peut prouver. Si on ne peut pas prouver, revoir les choix de conception ou d'architecture. Pour le reste, s'attendre à des CT**

- **Choix d'une solution optimale (existe-t-elle ?)**

- **Beaucoup reste à faire pour appréhender la prise en compte des CT dans les systèmes TR, distribués et tolérant les fautes.**

Nos travaux

- **Ordonnancement de messages**
  - ➔ **réseaux locaux**
  - ➔ **ATM**
- **Ordonnancement conjoint (tâches/messages)**
- **Ordonnancement et OO**
  - ➔ **RT-UML, SDL**
  - ➔ **RT-Corba**
  - ➔ **RT-Java**